

**Oracle Utilities**  
**Work and Asset Management**

Administrative User Guide

Release 2.1.1.0.0

**E69061-02**

February 2016  
(updated March 2017)

Oracle Utilities Work and Asset Management Administrative User Guide

Release 2.1.1.0.0

E69061-02

February 2016

Documentation build: 2.5.2016 7:51:43 [WAM2\_1454687503000]

Copyright © 2012, 2015, Oracle and/or its affiliates. All rights reserved.

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

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

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

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

# Contents

<b>Framework Administrative User Guide.....</b>	<b>15</b>
Defining General Options.....	15
Defining Installation Options.....	15
Installation Options - Main.....	15
Installation Options - Messages.....	16
Installation Options - Algorithms.....	16
Installation Options - Accessible Modules.....	18
Installation Options - Installed Products.....	18
Support For Different Languages.....	19
User Language.....	19
Additional Topics.....	20
Defining Languages.....	20
Defining Countries.....	21
Country - Main.....	21
Country - States.....	21
Defining Currency Codes.....	22
Defining Time Zones.....	22
Designing Time Zones.....	22
Setting Up Time Zones.....	23
Setting Up Seasonal Time Shift.....	23
Defining Geographic Types.....	23
Defining Work Calendar.....	24
Defining Display Profiles.....	24
Additional Hijri Date Configuration.....	26
Defining Phone Types.....	26
Setting Up Characteristic Types & Values.....	27
There Are Four Types Of Characteristics.....	27
Searching By Characteristic Values.....	27
Characteristic Type - Main.....	28
Characteristic Type - Characteristic Entities.....	29
Setting Up Foreign Key Reference Information.....	30
Information Description Is Dynamically Derived.....	30
Navigation Information Is Dynamically Derived.....	31
Search Options.....	31
Foreign Key Reference - Main.....	31
Defining Feature Configurations.....	32
Feature Configuration - Main.....	33
Feature Configuration - Messages.....	33
Defining Master Configurations.....	33
Setting Up Master Configurations.....	34
Miscellaneous Topics.....	34
Module Configuration.....	34
Menu Item Suppression.....	35
Menu Suppression.....	35
Turn Off A Function Module.....	35
Global Context Overview.....	35
System Data Naming Convention.....	36
Base Product System Data.....	36
Implementation System Data.....	36
Caching Overview.....	37
Server Cache.....	37
Client Cache.....	38
Batch Cache.....	38
Debug Mode.....	38
System Override Date.....	39
Advanced Search Options.....	39
Defining Security & User Options.....	40
The Big Picture of Application Security.....	40

Application Security.....	40
Action Level Security.....	41
Field Level Security.....	41
Encryption and Masking.....	42
Identify the Fields to Be Masked.....	42
Create a Security Type For Each Logical Field.....	43
Create An Algorithm For Each Security Type.....	43
Create A Feature Configuration For Each Secured Element.....	44
Grant Access Rights To The User Groups.....	46
Additional Masking Information.....	47
The Base Package Controls One User, One User Group, And Many Application Services.....	47
Importing Security Configuration from an External Source.....	48
The Big Picture of Row Security.....	48
Access Groups, Data Access Roles and Users.....	48
Defining Application Services.....	49
Application Service - Main.....	49
Application Service - Application Security.....	49
Defining Security Types.....	50
Security Type - Main.....	50
Defining User Groups.....	50
User Group - Main.....	50
User Group - Application Services.....	51
User Group - Users.....	52
Defining Access Groups.....	52
Defining Data Access Roles.....	52
Data Access Role - Main.....	53
Data Access Role - Access Group.....	53
Defining Users.....	53
User Interface Tools.....	53
Defining Menu Options.....	54
Menu - Main.....	54
Menu - Menu Items.....	55
The Big Picture of System Messages.....	56
Defining System Messages.....	57
Message - Main.....	57
Message - Details.....	58
The Big Picture of Portals and Zones.....	58
There Are Three Types of Portals.....	58
Common Characteristics of All Portals.....	59
Portals Are Made Up of Zones.....	59
Configuring Zones for a Portal.....	59
Granting Access to Zones.....	60
Common Characteristics of Stand-Alone Portals.....	60
Putting Portals on Menus.....	60
Granting Access to A Portal.....	61
Custom Look and Feel Options.....	61
User Interface.....	61
UI Map Help.....	62
Setting Up Portals and Zones.....	62
Defining Zone Types.....	62
Defining Zones.....	63
Zone - Main.....	63
Zone - Portal.....	64
Additional Zone Topics.....	65
Defining Context-Sensitive Zones.....	80
Defining Portals.....	80
Portal - Main.....	80
Portal - Options.....	82
Defining Display Icons.....	82
Defining Navigation Keys.....	82
Navigation Key Types.....	83
Navigation Key vs. Navigation Option.....	83
The Flexibility of Navigation Keys.....	83
Linking to External Locations.....	83

Overriding Navigation Keys.....	84
Maintaining Navigation Keys.....	84
Defining Navigation Options.....	85
Navigation Option - Main.....	85
Navigation Option - Tree.....	87
Database Tools.....	88
Defining Table Options.....	88
Table - Main.....	88
Table - Table Field.....	90
Table - Constraints.....	91
Table - Referred by Constraints.....	92
Defining Field Options.....	92
Field - Main.....	92
Field - Tables Using Field.....	93
Defining Maintenance Object Options.....	93
Maintenance Object - Main.....	94
Maintenance Object - Options.....	94
Maintenance Object - Algorithms.....	94
Maintenance Object - Maintenance Object Tree.....	96
Defining Valid Values.....	96
Defining Lookup Options.....	98
Lookup - Main.....	98
Defining Extendable Lookups.....	99
Defining Additional Attributes.....	100
The Big Picture Of Audit Trails.....	101
Captured Information.....	101
How Auditing Works.....	101
The Audit Trail File.....	102
How To Enable Auditing.....	102
Turn On Auditing For a Table.....	102
Specify The Fields and Actions To Be Audited.....	103
Audit Queries.....	103
Audit Query by User.....	103
Audit Query by Table / Field / Key.....	104
Bundling.....	105
About Bundling.....	105
Sequencing of Objects in a Bundle.....	105
Recursive Key References.....	106
Owner Flags on Bundled Entities.....	106
Configuring Maintenance Objects for Bundling.....	106
Making Maintenance Objects Eligible for Bundling.....	106
Adding a Foreign Key Reference.....	107
Creating a Physical Business Object.....	107
Creating a Bundling Add Business Object.....	107
Adding the Current Bundle Zone.....	108
Adding a Customized Entity Search Query Zone to the Bundle Export Portal.....	108
Working with Bundles.....	109
Creating Export Bundles.....	109
Creating and Applying Import Bundles.....	110
Editing Export Bundles.....	110
Editing Import Bundles.....	110
Revision Control.....	111
About Revision Control.....	111
Turning On Revision Control.....	111
Configuring Maintenance Objects for Revision Control.....	112
Working with the Revision Control Zones.....	112
Checking Out an Object.....	113
Checking In an Object.....	113
Reverting Changes.....	113
Forcing a Check In or Restore.....	114
Deleting an Object.....	114
Restoring an Object.....	114
Working with the Revision Control Portal.....	114
Revision Control Search.....	115

Information Lifecycle Management .....	115
The Approach to Implementing Information Lifecycle Management.....	116
Batch Processes.....	116
Eligibility Algorithm.....	118
Enabling ILM for Supported Maintenance Objects.....	118
Ongoing ILM Tasks.....	119
Archived Foreign Keys.....	119
Configuration Tools.....	120
Business Objects.....	120
The Big Picture of Business Objects.....	120
What Is A Business Object?.....	120
Business Object Inheritance.....	126
Each Business Object Can Have A Different Lifecycle.....	127
BO Algorithm Execution Summary.....	133
Granting Access To Business Objects.....	134
Defining Business Objects.....	134
Business Object - Main.....	135
Business Object - Schema.....	136
Business Object - Algorithms.....	136
Business Object - Lifecycle.....	138
Business Object - Summary.....	140
Defining Status Reasons.....	141
Business Services.....	141
Service Program.....	142
Defining Business Services.....	142
Business Service - Main.....	142
Business Service - Schema.....	143
User Interface (UI) Maps.....	143
Defining UI Maps.....	145
UI Map - Main.....	145
UI Map - Schema.....	146
Ensuring Unique Element IDs for UI Maps.....	147
Maintaining Managed Content.....	147
Managed Content - Main.....	147
Managed Content - Schema.....	148
Data Areas.....	148
Defining Data Areas.....	148
Data Area - Main.....	148
Data Area - Schema.....	149
Schema Designer.....	149
Schema Viewer.....	151
Business Event Log.....	151
To Do Lists.....	152
The Big Picture of To Do Lists.....	152
To Do Entries Reference A To Do Type.....	152
To Do Entries Reference A Role.....	152
To Do Entries Can Be Rerouted (Or Suppressed) Based On Message Number.....	153
The Priority Of A To Do Entry.....	154
Working On A To Do Entry.....	154
Launching Scripts When A To Do Is Selected.....	154
To Do Entries Have Logs.....	155
How Are To Do Entries Created?.....	155
To Do Entries Created By Background Processes.....	155
To Do Entries Created By Algorithms.....	156
To Do Entries Created Manually.....	157
The Lifecycle Of A To Do Entry.....	158
Linking Additional Information To A To Do Entry.....	158
Implementing Additional To Do Entry Business Rules.....	159
To Do Entries May Be Routed Out Of The System.....	159
Periodically Purging To Do Entries.....	159
Setting Up To Do Options.....	159
Installation Options.....	159
To Do Information May Be Formatted By An Algorithm.....	160
Set Additional Information Before A To Do Is Created.....	160

Alerts.....	160
Next Assignment Algorithm.....	160
Messages.....	160
Feature Configuration.....	160
Defining To Do Roles.....	161
To Do Role - Main.....	161
To Do Role - To Do Types.....	162
Defining To Do Types.....	162
To Do Type - Main.....	162
To Do Type - Roles.....	163
To Do Type - Sort Keys.....	163
To Do Type - Drill Keys.....	163
To Do Type - Message Overrides.....	164
To Do Type - To Do Characteristics.....	164
To Do Type - Algorithms.....	164
List of System To Do Types.....	166
Implementing The To Do Entries.....	166
Background Processes.....	166
The Big Picture of Background Processes.....	166
Background Processing Concepts.....	167
Parameters Supplied To Background Processes.....	167
Override Maximum Errors in Batch Process Parameter.....	169
Extra Parameters.....	169
Processing Errors.....	169
Parallel Background Processes.....	170
Optimal Thread Count.....	170
Timed Batch Processes.....	171
How to Re-extract Information.....	171
How to Submit Batch Jobs.....	172
How to Track Batch Jobs.....	172
How to Restart Failed Jobs and Processes.....	172
Assessing Level of Service.....	172
System Background Processes.....	172
Defining Batch Controls.....	173
Batch Control - Algorithms.....	175
On-Line Batch Submission.....	175
Batch Submission Creates a Batch Run.....	175
Jobs Submitted in the Background.....	176
Email Notification.....	176
Running Multi-Threaded Processes.....	176
Batch Jobs May End in Error.....	177
Submitting Jobs in the Future.....	177
Lifecycle of a Batch Job Submission.....	177
Batch Job Submission - Main.....	177
Tracking Batch Processes.....	179
Batch Run Tree - Main.....	179
Batch Run Tree - Run Control.....	180
Service Health Check.....	180
The Big Picture of Requests.....	181
Request Type Defines Parameters.....	181
Previewing and Submitting a Request .....	182
To Do Summary Email.....	182
Exploring Request Data Relationships.....	182
Defining a New Request.....	182
Setting Up Request Types.....	183
Maintaining Requests.....	183
Defining Algorithms.....	184
The Big Picture Of Algorithms.....	184
Algorithm Type Versus Algorithm.....	184
How To Add A New Algorithm.....	185
Minimizing The Impact Of Future Upgrades.....	185
Setting Up Algorithm Types.....	185
List of Algorithm Types.....	187
Setting Up Algorithms.....	187

Defining Script Options.....	187
The Big Picture Of Scripts.....	188
Scripts Are Business Process-Oriented.....	188
A Script Is Composed Of Steps.....	188
Designing And Developing Scripts.....	188
A Script May Declare Data Areas.....	189
Designing Generic Scripts.....	189
Securing Script Execution.....	189
The Big Picture Of BPA Scripts.....	190
How To Invoke Scripts.....	190
Developing and Debugging Your BPA Scripts.....	190
Launching A Script From A Menu.....	191
Launching A Script When Starting The System.....	191
Executing A Script When A To Do Entry Is Selected.....	192
The Big Picture Of Script Eligibility Rules.....	192
Script Eligibility Rules Are Not Strictly Enforced.....	192
You Can Mark A Script As Always Eligible.....	193
You Can Mark A Script As Never Eligible.....	193
Criteria Groups versus Eligibility Criteria.....	193
Defining Logical Criteria.....	194
Examples Of Script Eligibility Rules.....	194
The Big Picture Of Server-Based Scripts.....	196
Plug-In Scripts.....	196
A Plug-In Script's API.....	197
Setting Up Plug-In Scripts.....	197
Service Scripts.....	198
A Service Script's API.....	198
Invoking Service Scripts.....	198
Debugging Server-Based Scripts.....	198
Maintaining Scripts.....	198
Script - Main.....	199
Script - Step.....	200
How To Set Up Each Step Type.....	201
Additional Topics.....	211
Script - Data Area.....	219
Script - Schema.....	219
Script - Eligibility.....	220
Merging Scripts.....	221
Script Merge.....	222
Maintaining Functions.....	225
Function - Main.....	225
Function - Send Fields.....	226
Function - Receive Fields.....	227
Attachments.....	227
Attachment Overview.....	227
Configuring Your System for Attachments.....	228
Maintaining Attachments.....	229
Adding Attachments.....	230
Application Viewer.....	230
Application Viewer Toolbar.....	230
Data Dictionary Button.....	230
Physical and Logical Buttons.....	231
Collapse Button.....	231
Attributes and Schema Button.....	231
Maintenance Object Button.....	231
Algorithm Button.....	232
Batch Control Button.....	232
To Do Type Button.....	232
Description and Code Buttons.....	232
Service XML Button.....	232
Select Service Button.....	233
Java Docs Button.....	233
Classic Button.....	233
Preferences Button.....	233



Help Button.....	233
About Button.....	233
Slider Icon.....	234
Data Dictionary.....	234
Using the Data Dictionary List Panel.....	234
Primary And Foreign Keys.....	235
Field Descriptions Shown.....	235
Using the Data Dictionary Detail Panel.....	235
Related Tables View.....	235
Table Detail View.....	236
Column Detail View.....	236
Maintenance Object Viewer.....	236
Using the Maintenance Object List Panel.....	237
Using the Maintenance Object Detail Panel.....	237
Algorithm Viewer.....	237
Using the Algorithm Viewer List Panel.....	237
Using the Algorithm Plug-In Spot Detail Panel.....	237
Using the Algorithm Type Detail Panel.....	237
Using the Algorithm Detail Panel.....	238
Batch Control Viewer.....	238
Using the Batch Control Viewer List Panel.....	238
Using the Batch Control Detail Panel.....	238
To Do Type Viewer.....	238
Using the To Do Type Viewer List Panel.....	239
Using the To Do Type Detail Panel.....	239
Service XML Viewer.....	239
Using the Service XML Viewer Overview Panel.....	239
Using the Service XML Viewer Detail Panel.....	239
Java Docs Viewer.....	240
Using the Java Docs Viewer List Panel.....	240
Using the Java Package Detail Panel.....	240
Using the Java Interface / Class Detail Panel.....	240
Application Viewer Preferences.....	241
Application Viewer Stand-Alone Operation.....	241
Stand-Alone Configuration Options.....	241
Example Application Viewer Configuration.....	242
Application Viewer Generation.....	242
Defining and Designing Reports.....	243
The Big Picture Of Reports.....	243
Integration with BI Publisher and Business Objects Enterprise.....	243
Reports Must Be Multi-Language.....	243
Requesting Reports from The System.....	243
Overview of the Data - BI Publisher.....	244
Overview of the Data - Business Objects Enterprise.....	244
How To Request Reports.....	245
Viewing Reports.....	245
Configuring The System To Enable Reports.....	245
Configuring BI Publisher Reports.....	245
Configure the System to Invoke BI Publisher Real-time.....	245
Batch Scheduling in BI Publisher.....	246
Configuring Business Objects Enterprise Reports.....	246
Configure the System to Invoke Business Objects Enterprise Real-time.....	246
Batch Scheduling in Business Objects Enterprise.....	246
Defining Reporting Options.....	247
Defining Report Definitions.....	247
Report Definition - Main.....	247
Report Definition - Labels.....	248
Report Definition - Parameters.....	248
Sample Reports Supplied with the Product.....	249
How to Use a Sample Report Provided with the System.....	249
Steps Performed at Installation Time.....	249
Subreports Used with Crystal Reports.....	249
Display Company Logo and Title.....	249
Format Report Information.....	250

Labels.....	250
How To Define A New Report.....	250
Use a Sample Report as a Starting Point.....	250
Publishing Reports in BI Publisher.....	251
BI Publisher Database Access.....	251
Verify BI Publisher User Access Rights.....	251
Publishing Reports in Business Objects Enterprise.....	251
Business Objects Enterprise Database Access.....	251
Verify Parameter Definition.....	252
Verify Business Objects Enterprise User Access Rights.....	252
Designing Your Report Definition.....	252
Designing Main Report Definition Values.....	252
Designing Characteristic Types.....	252
Designing Parameters.....	253
Designing Validation Algorithms.....	254
Designing Application Services.....	254
Designing Labels.....	254
External Messages.....	254
Incoming Messages.....	255
Inbound Web Services.....	255
Overview.....	255
Configuring Inbound Web Service Options.....	256
Deploying Web Services.....	257
Guaranteed Delivery.....	259
Outgoing Messages.....	259
Outbound Messages.....	259
Polling Outbound Messages Using OSB.....	260
Batch Message Processing.....	260
Real Time Messages.....	261
Configuring the System for Outbound Messages.....	261
Message Sender.....	262
Defining Outbound Message Types.....	265
External Systems.....	266
Managing Outbound Messages.....	267
Message Option.....	268
Web Service Adapters.....	271
Understanding Web Service Adapters.....	271
Setting Up Web Service Adapters.....	273
Sending Email.....	273
XML Application Integration.....	273
The Big Picture of XAI.....	274
XAI Architecture.....	274
XML Background Topics.....	277
Inbound Messages.....	280
Outgoing Messages.....	289
Designing Your XAI Environment.....	291
Installation.....	291
Designing XAI Inbound Services.....	293
Designing XML Schemas.....	293
Designing XSL Transformations.....	294
Designing Your Registry Options.....	294
Configuring the System for NDS Messages.....	301
Schema Editor.....	301
Opening the Schema Editor.....	302
Schema Editor Window.....	302
Validating a Schema.....	305
Registering a Service.....	305
Testing a Schema.....	305
System Wide Functions for Schema Editor.....	306
Setting Up Your XAI Environment.....	308
Message Class.....	309
XAI Envelope Handler.....	309
Setting Up Your Registry.....	309
XAI Route Type.....	318

Running Your XAI Environment.....	318
XAI Staging Control.....	318
XAI Upload Staging.....	319
Uploading XAI Staging Records.....	320
XAI Dynamic Upload.....	322
XAI Upload Exception.....	323
XAI Download Staging.....	323
XAI Download Exception.....	324
Maintaining Your XAI Environment.....	324
XAI Submission.....	324
XAI Dynamic Submission.....	325
Additional XAI Tools.....	326
Server Trace.....	328
Integrations.....	329
LDAP Integration.....	329
LDAP Integration Overview.....	330
Configuring LDAP Integration .....	331
LDAP Mapping.....	332
Oracle Identity Manager Integration.....	334
Data Synchronization.....	335
Defining Sync Requests.....	336
The Big Picture of Sync Requests.....	336
Designing Your Sync Requests.....	337
Sync Requests.....	338
Sync Request Query.....	338
Sync Request Portal.....	338
Configuration Migration Assistant (CMA).....	338
Understanding CMA.....	338
The CMA Process Flow.....	339
Migration Assumptions, Restrictions and Recommendations.....	341
CMA Configuration.....	342
Master Configuration - Migration Assistant.....	342
Migration Plans.....	342
Defining a Migration Plan.....	342
Understanding the BO Filtering Process.....	344
Migration Plans for Objects with CLOB-Embedded Links.....	344
Defining a Migration Request.....	345
Wholesale and Piecemeal Migrations.....	345
Wholesale Migrations.....	346
Piecemeal Migrations.....	346
Identifying Tables to Exclude From Migrations.....	347
Configuring Custom Objects for Migration.....	347
The CMA Execution Process.....	348
Exporting a Migration.....	349
Migration Data Set Export.....	350
Export Lifecycle.....	350
Importing and Applying a Migration.....	351
Import Step.....	351
Compare Step.....	353
Approval Step.....	355
Apply Step.....	356
Adjusting Data While Importing.....	360
Import Process Summary .....	360
Cancelling a Data Set.....	363
Additional Note Regarding Imports.....	363
Caching Considerations.....	364
Maintaining Import Data.....	364
Running Batch Jobs.....	366
CMA Reference.....	367
Framework-Provided Migration Configuration.....	367
Configuring Facts.....	369
Fact Is A Generic Entity.....	369
Fact's Business Object Controls Everything.....	369
Fact Supports A Log.....	369

<b>Oracle Utilities Work and Asset Management Administrative User Guide.....</b>	<b>371</b>
System Wide Oracle Utilities Work and Asset Management Configuration.....	371
Defining Home Page Configuration.....	371
Defining General System Master Configuration.....	372
About Access Control.....	372
About Statistics.....	372
Communication.....	373
About Communication Types.....	373
Defining Communication Types.....	373
BI Configuration Administration.....	373
Designing Approval Processing.....	374
Understanding Approval Processing Configuration.....	374
About Approval Profiles.....	375
Configuring Approval Processing.....	375
Defining an Approval Profile.....	375
Setting the Approval Profile for a Record Type.....	376
Defining the Approval Hierarchy.....	376
About No Approval Required Profiles.....	378
About No Amount Profiles.....	378
Defining To Do Types, To Do Roles, and To Do Entries for Approvals.....	378
Designing Asset Management.....	379
Understanding Asset Management Configuration.....	379
About Asset Types, Assets and Components.....	379
About Asset Location Types.....	380
About Service History Types and Service History.....	380
About Condition Assessment Questions.....	381
About Condition Assessment Groups.....	381
About Condition Assessment Classes.....	382
About Asset Inspections (Assessments).....	382
About Measurement Types and Measurements (Operational Readings) .....	386
About Measurement Unit of Measurement.....	386
About Asset and Component Configuration Types.....	386
Configuring Asset Management.....	387
Defining Measurement Unit of Measurements (UOM).....	387
Defining Measurement Types (Operational Readings).....	387
Defining Questions.....	387
Defining Service History Types.....	387
Defining Condition Assessment Groups.....	387
Defining Condition Assessment Classes.....	388
Defining Asset Location Types.....	388
Defining Out of Service Location Types.....	388
Defining Asset Types.....	389
Defining Component Types.....	389
Defining Asset Configuration Types.....	389
Designing Preventive Maintenance.....	390
Understanding Preventive Maintenance Configuration.....	390
About Preventive Maintenance Configuration.....	390
Designing Inventory Management.....	391
Understanding Inventory Configuration.....	391
About Stock Items, Stock Item Details and Storerooms.....	391
About Commodity Codes.....	393
Configuring Inventory Management.....	394
Defining Storeroom Types.....	394
Defining Commodity Codes.....	394
Designing Purchasing.....	395
Understanding Purchasing Configuration.....	395
About Buyers.....	395
About Purchase Commodities.....	395
About Capability Types.....	395
About Payment Terms.....	396
About FOB (Freight on Board or Free on Board).....	396
About Invoicing.....	396
Configuring Purchasing.....	397
Defining Buyers.....	398

Defining Purchase Commodities.....	398
Defining Capability Types.....	398
Defining Vendor Payment Terms.....	398
Defining FOB (Freight on Board or Free on Board).....	398
Defining Exchange Rates.....	398
Designing Resource Management.....	399
Understanding Resource Management Configuration.....	399
About Organization Types and Organizations.....	399
About Contact Types and Contacts.....	400
About Document Types and Documents.....	400
About Service Classes.....	400
About Service Areas.....	401
About Resource Unit of Measurement.....	401
About Business Units.....	401
About Employee Types and Employees.....	401
About Employee Unavailability Types and Employee Unavailability.....	401
About Crew Shift Types, Crew Types, Crew Shifts and Crews.....	402
About Crafts, Equipment and Miscellaneous Resources.....	402
Configuring Resource Management.....	403
Defining Resource Unit of Measurements (UOM).....	403
Defining Organization Types.....	403
Defining Service Classes.....	403
Defining Service Areas.....	403
Defining Contact Types.....	403
Defining Document Types.....	404
Defining Business Units.....	404
Defining Employee Types.....	404
Defining Employee Unavailability Types.....	404
Defining a Crew Type.....	404
Defining a Crew Shift Type.....	405
Designing Work Management.....	405
Understanding Work Management Configuration.....	405
About Planners and Maintenance Managers.....	405
About Activity Types and Activities.....	405
About Work Priority and Criticality Fields.....	406
About Completion Event Types.....	406
About Inbound Communications.....	407
About Projects and Subprojects.....	407
About Timekeeping.....	408
Configuring Work Management.....	408
Defining Maintenance Managers.....	408
Defining Planners.....	409
Defining Activity Types.....	409
Defining Completion Event Types.....	409
Defining Time Periods.....	409
Defining Labor Earning Types.....	409
Defining Overtime Types.....	409
Defining Project Categories.....	410
Designing Financials.....	410
Understanding Financials Configuration.....	410
About Accounting Calendars.....	410
About Currency.....	411
About Colors.....	411
About Cost Categories.....	411
About Cost Centers.....	411
About Expense Codes.....	412
About Factors and Factor Values.....	412
About Distribution Codes.....	412
About Tax Rate Schedules.....	412
Configuring Financials.....	412
Defining Accounting Periods.....	413
Defining Account Segments.....	414
Creating Accounting Calendars.....	414
Defining Currency.....	414

Defining Colors.....	414
Defining Cost Categories.....	415
Defining Cost Centers.....	415
Reviewing Cost Center Statistics.....	415
Deactivating Cost Centers.....	415
Defining Expense Codes.....	415
Deactivating Expense Codes.....	416
Defining Factors.....	416
Defining Distribution Codes.....	416
Defining Tax Rate Schedules.....	416
Integrations.....	417
About Integrations.....	417
Specific Product Integrations.....	417
Defining External Systems.....	418
Data Synchronization for Integrations.....	418
Understanding Synchronization Request Processing for Integrations.....	419
BI Publisher Integration.....	420
BI Publisher Integration Setup Steps.....	420
Control Table Setup Sequence.....	420
Control Data Setup.....	421
Master Data Setup.....	429
Batch Control Configuration.....	435
About Monitors.....	435
Oracle Utilities Work and Asset Management Batch Controls.....	436

# Chapter 1

---

## Framework Administrative User Guide

---

The topics in this section describe how to administer the Oracle Utilities Application Framework.

### Defining General Options

---

This section describes control tables that are used throughout your product.

### Defining Installation Options

The topics in this section describe the various installation options that control various aspects of the system.

### Installation Options - Main

Select **Admin > General > Installation Options - Framework** to define system wide installation options.

#### Description of Page

The **Environment ID** is a unique universal identifier of this instance of the system. When the system is installed, the environment id is populated with a six digit random number. While it is highly unlikely that multiple installs of the system at a given implementation would have the same environment ID, it is the obligation of the implementers to ensure that the environment ID is unique across all installed product environments.

**System Owner** will be **Customer Modification**.

The **Admin Menu Order** controls how the various control tables are grouped on Admin.

- If you choose **Alphabetical**, each control table appears under a menu item that corresponds with its first letter, using a Roman alphabet. For example, the Language control table will appear under the L menu item entry.
- If you choose **Functional**, each control table appears under a menu item that corresponds with its functional area. Note, the [menu](#) that is used when this option is chosen is the one identified with a menu type of **Admin**.

---

**NOTE:** The **Alphabetical** option only supports the Roman alphabet. For languages that do not use the Roman alphabet, the recommendation is to configure the system for the **Functional** setting.

---

**CAUTION:** In order to improve response times, installation options are cached the first time they are used after a web server is started. If you change the Admin Menu Order and you don't want to wait for the cache to rebuild, you must clear the cached information so it will be immediately rebuilt using current information. Refer to [Caching Overview](#) for information on how to clear the system login cache (this is the cache in which installation options are stored).

---

The **Language** should be set to the primary language used by the installation. Note that if multiple languages are supported, each user may define their preferred language.

The **Currency Code** is the default currency code for transactions in the product.

If your product supports effective dated characteristics on any of its objects, define the date to be used as the **Characteristic Default Date** on objects without an implicit start date. The date you enter in this field will default when new characteristics are added to these objects (and the default date can be overridden by the user).

**Active Owner** displays the owner of newly added system data (system data is data like algorithm types, zone types, To Do types, etc.). This will be **Customer Modification** unless you are working within a development region.

**Country** and **Time Zone** represent the default country and time zone that should be used throughout the application.

Turn on **Seasonal Time Shift** if your company requires seasonal time shift information to be defined. Note that this is currently only applicable to Oracle Customer Care and Billing > Interval Billing functionality.

## Installation Options - Messages

Select **Admin > General > Installation Options - Framework** and the **Messages** tab to review or enter messages that will appear throughout the application when a given event occurs.

The **Message** collection contains messages that are used in various parts of the system. For each message, define the **Installation Message Type** and **Installation Message Text**. The following table describes the **Message Types** provided by the framework product and how they are used in the system. Your specific product may have introduced additional message types.

Message Type	How The Message Is Used
Company Title for Reports	This message appears as a title line on the <a href="#">sample reports</a> provided with the system. Generally it is your company name. It is only used if you have installed reporting functionality and are using the sample reports (or have designed your reports to use this message).

## Installation Options - Algorithms

Select **Admin > General > Installation Options - Framework** and the **Algorithms** tab to review or enter the algorithms that should be evoked when a given event occurs.

The grid contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.



**CAUTION:** These algorithms are typically significant processes. The absence of an algorithm might prevent the system from operating correctly.

The following table describes each **System Event**.

<b>System Event</b>	<b>Optional / Required</b>	<b>Description</b>
<b>Validate Email Attachment</b>	Optional	Algorithms of this type are used to validate the attachments for size and total count while sending attachments using the Email service. Refer to <a href="#">Sending Email</a> for more information.  Click <a href="#">here</a> to see the algorithm types available for this system event.
<b>Geocoding Service</b>	Optional	Algorithms of this type use Oracle Locator to retrieve latitude and longitude coordinates using address information.  Click <a href="#">here</a> to see the algorithm types available for this system event.
<b>Global Context</b>	Optional	Algorithms of this type are called whenever the value of one of the global context fields is changed. Algorithms of this type are responsible for populating other global context values based on the new value of the field that was changed.  Refer to <a href="#">Global Context Overview</a> for more information.  Click <a href="#">here</a> to see the algorithm types available for this system event.
<b>Guaranteed Delivery</b>	Optional	Algorithms of this type may be called by processes that receive incoming messages that should 'guarantee delivery'. Refer to <a href="#">Guaranteed Delivery</a> for more information. The business service <b>F1-GuaranteedDelivery</b> may be used to invoke this plug-in spot.  Click <a href="#">here</a> to see the algorithm types available for this system event.
<b>Ldap Import</b>	Optional	Algorithms of this type are called for operations on users, groups, and group memberships after they have been processed.  Click <a href="#">here</a> to see the algorithm types available for this system event.
<b>Ldap Import Preprocess</b>	Optional	Algorithms of this type are called to preprocess data retrieved from LDAP.  Click <a href="#">here</a> to see the algorithm types available for this system event.
<b>Next To Do Assignment</b>	Optional	This type of algorithm is used to find the next <a href="#">To Do entry</a> a user should work on. It is called from the <a href="#">Current To Do</a> dashboard zone when the user ask for the next assignment.  Click <a href="#">here</a> to see the algorithm types available for this system event.
<b>Reporting Tool</b>	Optional	If your installation has integrated with a third party reporting tool, you may wish to allow your users to submit reports on-line using <a href="#">report submission</a> or to review <a href="#">report history</a> online. This algorithm is used by the two on-line reporting pages to properly invoke the reporting tool from within the system.  Click <a href="#">here</a> to see the algorithm types available for this system event.

System Event	Optional / Required	Description
SMS Receive Service	Optional	This type of algorithm is used to provide SMS receive service. Only one algorithm of this type should be plugged in. Click <a href="#">here</a> to see the algorithm types available for this system event.
SMS Send Service	Optional	This type of algorithm is used to provide SMS send service. If your installation uses the base algorithm that uses BPEL, you will need to create a feature configuration with the SMS Send Configuration feature type to define your Oracle BPEL server and service call details. If your installation has integrated with a third-party SMS service, you may want to override this algorithm type with your own implementation. Only one algorithm of this type should be plugged in. Click <a href="#">here</a> to see the algorithm types available for this system event.
To Do Information	Optional	We use the term To Do information to describe the basic information that appears throughout the system to describe a <a href="#">To Do entry</a> . Plug an algorithm into this spot to override the system default "To Do information". Click <a href="#">here</a> to see the algorithm types available for this system event.
To Do Pre-creation	Optional	These types of algorithms are called when a <a href="#">To Do entry</a> is being added. Click <a href="#">here</a> to see the algorithm types available for this system event.

## Installation Options - Accessible Modules

Select **Admin > General > Installation Options - Framework** and the **Accessible Modules** tab to view the list of accessible modules.

### Description of Page

This page displays the full list of the application's function modules. A **Turned Off** indication appears adjacent to a module that is not accessible based on your system's module configuration setup.

**FASTPATH:** Refer to [Module Configuration](#) for more information on function modules and how to turn off modules that are not applicable to your organization.

## Installation Options - Installed Products

Select **Admin > General > Installation Options - Framework** and the **Installed Products** tab to view a read only summary of the products that are installed in the application version that you are logged into.

### Description of Page

The **Product Name** indicates the name of the "products" that are installed. The collection should include **Framework**, an entry for your specific product and an entry for **Customer Release**.

**Release ID** shows the current release of the application that is installed. This field is used by the system to ensure that the software that executes on your application server is consistent with the release level of the database. If your implementation of the product has developed implementation-specific transactions, you can populate the Release Id for the **Customer**

**Release** entry to define the latest release of implementation-specific logic that has been applied to this environment. In order for this to work, your implementation team should populate this field as part of their upgrade scripts.

The **Release ID Suffix**, **Build Number** and **Patch Number** further describe the details of your specific product release.

The **Display** column indicates the product whose name and release information should be displayed in the title bar. Only one product sets this value to **Yes**.

**Owner** indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

**Product Type** indicates if the product is a Parallel Application. A parallel application is one that is independent of, and does not conflict with, other parallel applications. Multiple parallel applications can be installed in the same database and application server.

---

**NOTE: About Information.** The information on this tab is used to populate the information displayed in the [About](#) information for your product.

---

## Support For Different Languages

### User Language

The system provides support for multiple languages in a single environment. System users can use the system in their preferred language, as long as a translation into that language has been provided. A user sees the system in the language defined on their [user record](#). If enabled, users can use the [Switch Language](#) zone to switch to another supported language real time.

---

**NOTE:** Normally, setting up the system for another language is an implementation issue, not an administrative setup issue. However, there are several online administrative features that are used to set up a new language, and these are described here.

---

The following steps are required to support a new language:

- 1. Define a language code and indicate that it is enabled.** For details on this procedure, see [Defining Languages](#).
- 2. Copy descriptions of all language-enabled tables from an existing translation (e.g., English).** The copied values act merely as placeholders while the strings are translated into the new language. It is necessary to do this as a first step in order to create records using the new language code created in the previous step. Language-based descriptions can be copied using a supplied batch process, [FI-LANG](#). The batch copies all English labels in the system.
- 3. Apply the language pack.** If the product supplies a language pack with translations for the system metadata descriptions, follow the instructions provided with the language pack to add the translated text.
- 4. Translate additional content.** Translatable descriptions and labels for implementation data may be updated / entered in the application. First the user record must be updated to reference the new language. This may be done in one of the following ways:
  - a.** Switch to the new language using the [Switch Language](#) zone.
  - b.** If that zone is not available, navigate to the user page, assign the new language code to your User ID, sign out, and sign back in again.

Any online functions that you access will use your new language code. You can change the language code for all users who plan to use/modify the new language.

## Additional Topics

Your product may support additional uses for language. For example, in Oracle Utilities Customer Care and Billing, you can define each customer's language. This allows you to send bills and other correspondence in each customer's preferred language. For more information, navigate to the index entry **Languages, Customer Language**.

## Defining Languages

A language code exists for every language spoken by your users. The system uses this code to supply information to users in their respective language. Select **Admin > General > Language** to define a language.

### Description of Page

Enter a unique **Language Code**. If you are applying a language pack provided by the product, use the language code designed by the language pack.

Enter the **Description** for the language. Typically this should be the name of the language in that language.

Turn on **Language Enable** if the system should add a row for this language whenever a row is added in another language. For example, if you add a new currency code, the system will create language specific record for each language that has been enabled. You would only enable multiple languages if you have users who work in multiple languages. Languages that are configured as enabled, appear in the [Switch Language](#) dashboard zone. In addition, the login page for the application displays all the languages that are enabled, allowing the user to toggle the login instructions in that language.

---

**NOTE:** The list of enabled languages is captured on the server at startup time. If a new language is enabled, contact your server administrator to refresh the server in order to see the new language displayed in the login page.

---

The following two fields control how the contents of grids and search results are sorted by the Java virtual machine (JVM) on your web server:

- The **Locale** is a string containing three portions:
  - ISO language code (lower case, required)
  - ISO country code (upper case, optional)
  - Variant (optional).
- Underscores separate the various portions, and the variant can include further underscores to designate multiple variants. The specific JVM in use by your particular hardware/OS configuration constrains the available **Locales**. Validating the **Locale** against the JVM is outside the scope of this transaction. This means you are responsible for choosing valid **Locales**.

The following are examples of valid locales:

- en\_US (this is for American English)
- en\_AU (this is for Australian English)
- pt\_BR (this is for Brazilian Portuguese)
- fr\_FR\_EURO (this is for European French)
- ja\_JP (this if for Japanese)

In addition, the Java collation API can take a **Collator Strength** parameter. This parameter controls whether, for example, upper and lower-case characters are considered equivalent, or how accented characters are sorted. Valid values for collator strength are **PRIMARY**, **SECONDARY**, **TERTIARY**, and **IDENTICAL**. If you leave this field blank, Java will use its default value for the language. We'd like to stress that the impact of each value depends on the language.

Please see <http://java.sun.com/j2se/1.3/docs/guide/intl/locale.doc.html> for more information about the collator strength for your language.

**Display Order** indicates if this language is written **Left to Right** or **Right to Left**.

**Owner** indicates if this language is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a language. This information is display-only.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_LANGUAGE](#).

Note that all administrative control tables and system metadata that contain language-specific columns (e.g., a description) reference a language code.

In addition, other tables may reference the language as a specific column. For example, on the User record you indicate the preferred language of the user.

## Defining Countries

The topics in this section describe how to maintain countries.

### Country - Main

To add or review Country definitions choose **Admin > General > Country > Search**.

The **Main** page is used to customize the fields and field descriptions that will be displayed everywhere addresses are used in the system. This ensures that the all addresses conform to the customary address format and conventions of the particular country you have defined.

#### Description of Page

Enter a unique **Country** and **Description** for the country.

The address fields that appear in the **Main** page are localization options that are used to customize address formats so that they conform to address requirements around the world. By turning on an address field, you make that field available everywhere addresses for this country are used in the system. You can enter your own descriptions for the labels that suffix each switch; these labels will appear wherever addresses are maintained in the system.

---

**NOTE:** For any country where the **State** switch is checked, the valid states for the country must be entered on the **Country - State** tab. When entering address constituents on a record that captures this detail, the value for State is verified against the data in the State table. For any country where there is a component of the address that represents a "state" but your implementation does not want to populate the valid states for that country, choose a different field such as County for this constituent (and define an appropriate label). When entering address constituents on a record that captures this detail, no validation is done for the County column.

---

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_COUNTRY](#).

### Country - States

To maintain the states located in a country, choose **Admin > Country > Search** and navigate to the **State** page.

#### Description of Page

For any country where you have enabled the State switch, use the **State** collection to define the valid states in the **Country**.

- Enter the standard postal abbreviation for the **State** or province.

- Enter a **Description** for this state or province.

## Defining Currency Codes

The currency page allows you to define display options related to currency codes that are used by your system. Use **Admin > Financial > Currency** to define the currency codes in which financial information is denominated.

### Description of Page

Enter a unique **Currency** and **Description** for the currency.

Use Currency **Symbol** to define the character that prefixes currency amounts in the system (e.g., \$ for U.S. dollars).

Enter the number of **Decimals** that will appear in the notation for the currency.

---

**NOTE:** Please contact your specific product to verify whether it supports a currency with more than 2 decimals.

---

The **Currency Position** indicates whether the currency symbol should be displayed as a **Prefix** or a **Suffix** to the currency amount.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_CURRENCY\\_CD](#).

## Defining Time Zones

The following topics describe how to design and set up time zones.

## Designing Time Zones

---

**NOTE:** Oracle Utilities Customer Care and Billing - Interval Billing applications customers should consult the topic *Time Issues* (search the Help index for "time issues") for specific information relating to that product's interval billing time related functionality.

---

It is recommended that all time sensitive data is stored in the standard time (also called 'physical time') of the base time zone as defined on the installation options. This will prevent any confusion when analyzing data and will ensure that your algorithms do not have to perform any shifting of data that may be stored in different time zones.

The Time Zone entity is used to define all the time zones where your customers may operate. Each time zone should define an appropriate Time Zone Name. This is a reference to an external source that defines time zones, their relationship to Greenwich Mean Time, whether the time zone follows any shifting for summer / winter time (daylight savings time) and when this shift occurs.

When designing your time zones, the first thing to determine is the base time zone. You may choose the time zone where the company's main office resides. Once this is done you can link the time zone code to the installation option as the base time zone. Refer to [Installation Options - Main](#) for more information.

If your company does business beyond your main office's time zone, define the other time zones where you may have customers or other systems with which you exchange data. At this point, your specific product may include configuration tables to capture default time zones, for example based on a postal code or geographic location.

---

**NOTE: Date and time in business object schemas.** When defining date / time fields in a BO schema, schema attributes can be used to define whether or not data should be stored in standard time for the base time zone or if it should be stored in the standard time of another time zone (related to the data). In addition, schema attributes can be used to indicate if the display of the time should be shifted to represent the "local time". This is used to adjust for seasonal time differences. For example, if the data is stored in the appropriate time zone, but currently daylight savings time is being

---

observed, the data will be shifted and shown in the “local” time. In addition, if the data is stored in the base time zone but the data is related to a different time zone, the data will be shown in the time zone appropriate for the data (including the appropriate seasonal adjustment). Refer to Schema Tips on the business object page for more information.

---

## Setting Up Time Zones

Refer to [Designing Time Zones](#) for background information about defining time zones.

Open **Admin > General > Time Zone > Search** to define the time zones and their relation to the base time.

### Description of Page

Enter a unique **Time Zone** and **Description** for the time zone.

Select the **Time Zone Name** from the list of Olson time zone values. This value is a reference to an external definition that allows the system to know how the time zone relates to Greenwich Mean Time and information about whether the time zone shifts for summer / winter time and when.

Indicate the **Shift in Minutes** that this time zone differs from the base time zone defined on the Installation Options. This is only applicable for the *Oracle Utility Customer Care and Billing - Interval Billing* application.

Indicate the **Seasonal Time Shift** applicable for this time zone. This is only applicable for the *Oracle Utility Customer Care and Billing - Interval Billing* application.

**Default Time Zone Label** and **Shifted Time Zone Label** are used for data that is sensitive to time zones and time shifting. It indicates whether the data displayed or data to be input is related to the “standard” time or the “shifted” time. For example, on a day when clocks are turned back one hour, a time entry of 1:30 a.m. needs to be labeled as either 1:30 a.m. standard time or 1:30 a.m. daylight savings time.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_TIME\\_ZONE](#).

## Setting Up Seasonal Time Shift

---

**NOTE:** The information in this topic applies only to Oracle Utilities Customer Care and Billing - Interval Billing applications.

---

Open **Admin > General > Seasonal Time Shift > Search** to define the seasonal time shift schedule.

### Description of Page

Enter a unique **Seasonal Time Shift** code and **Description** for the seasonal time shift.

The Collection defines the **Effective Date/Time** (in standard time) that a time zone may shift in and out of standard time. If time is changed from standard time on the effective date/time, enter the **Shift in Minutes** that the time changes from standard time (usually **60**). If the time is changed back to standard time on the effective date/time, enter a **Shift in Minutes** of **0**.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_SEAS\\_TM\\_SHIFT](#).

## Defining Geographic Types

If your company uses geographic coordinates for dispatching or geographic information system integration, you need to setup a geographic (coordinate) type for each type of geographic coordinate you capture on your premises and/or service points (geographic coordinates can be defined on both premises and service points).

To define geographic types, open **Admin > Geographic > Geographic Type**.

---

**NOTE: Find a customer / premise using a geographic coordinate.** In Oracle Utilities Customer Care and Billing there are several queries that allow you to search for customers and premises by geographic type. For example, Control Central and Meter Search allow you to search using the premise or service point geographic type. Refer to the product documentation for more information.

---

### Description of Page

Enter an easily recognizable **Geographic Type** code and **Description**.

Define the algorithm used to validate the **Validation Format Algorithm**. If an algorithm is specified, the system will validate that the geographic location entered on the premise and/or service point for the geographic type is in the format as defined in the algorithm. If you require validation, you must set up this [algorithm](#) in the system.

Click [here](#) to see the algorithm types available for this plug-in spot.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_GEO\\_TYPE](#).

## Defining Work Calendar

Workday calendars are used to ensure system-calculated dates fall on a workday. Select **Admin > General > Work Calendar > Search** to define a workday calendar.

### Description of Page

The information on this transaction is used to define the days of the week on which your organization works.

Enter a unique **Work Calendar** and **Description**.

Turn on (check) the days of the week that are considered normal business days for your organization.

Use the collection to define the **Holiday Date**, **Holiday Start Date**, **Holiday End Date**, and **Holiday Name** for each company holiday. Holiday Start Date and Holiday End Date define the date and time that the holiday begins and ends. For example, your organization might begin a holiday at 5:00 p.m. on the day before the actual holiday.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_CAL\\_WORK](#).

## Defining Display Profiles

When you set up your [users](#), you reference a display profile. A user's display profile controls how dates, times, and numbers displayed. Choose **Admin > General > Display Profile > Search** to maintain display profiles.

### Description of Page

Enter a unique **Display Profile ID** and **Description** to identify the profile.

Enter a **Date Format**. This affects how users view dates and how entered dates are parsed. This is a "free format" field with some rules.

- **dd** or **d** is interpreted as the day of the month. The **d** option suppresses a leading 0.
- **MM** or **M** is interpreted as the month number. The **M** option suppresses a leading 0.
- **yyyy**, **yy**, or **y** is interpreted as the year. The year can be 4 or 2 digits. The **y** option allows entry in either 2 or 4-digit form and is displayed in 2-digit form.
- Other characters are displayed as entered. Typically, these other characters should be separators, such as "-", ".", or "/". Separators are optional; a blank space cannot be use.



Here are some examples of date formats.

Format	Displayed / entered as
<b>MM-dd-yyyy</b>	04-09-2001
<b>d/M/yyyy</b>	9/4/2001
<b>yy.MM.dd</b>	01.04.09
<b>MM-dd-y</b>	04-09-01 - in this case you could also enter the date as 04-09-2001

**NOTE:** For centuries, the default pivot for 2-digit years is **80**. Entry of a 2-digit year greater than or equal to **80** results in the year being interpreted as 19xx. Entry of a 2-digit year less than **80** results in the year being interpreted as 20xx.

In addition, the following date localization functionality is supported. Note that in every case, the date is stored in the database using the Gregorian format. The settings below result in a conversion of the date for the user interface.

- **Hijri Dates**

Entering **iiii** for the year is interpreted as a year entered and displayed in Hijri format. For example, the Gregorian date 2014–05–30 may be entered / displayed as 1435/07/30 for a user whose display profile date format is **iiii/MM/dd**. Note that this functionality relies on date mapping to be defined in the Hijri to Gregorian Date Mapping entry. Refer to [Additional Hijri Date Configuration](#) for more information.

- **Taiwanese Dates**

Entering **tttt** for the year is interpreted as a year entered and displayed in Taiwanese format where year 1911 is considered year 0000. For example, if the Gregorian date is 01-01-2005, it is displayed as 01-01-0094 for a user whose display profile date format is **dd-mm-tttt**.

- **Japanese Dates**

There are two options available for configuring Japanese Era date support. The setting **Gyy** for the year is interpreted as a year entered and displayed using an English character for the era followed by the era number. The letter ‘T’ is used for dates that fall within the *Taisho* era. The letter ‘S’ is used for dates that fall within the *Showa* era and the letter ‘H’ is used for dates that fall within the *Heisei* era. For example, for a user whose display profile date format is **Gyy/mm/dd** the Gregorian date 2008/01/01 is shown as **H20/01/01** ; the Gregorian date 1986/03/15 is shown as **S61/03/15**. The setting **GGGGyy** is interpreted as a year entered and displayed using Japanese characters for the era followed by the era number.

Japanese date limitations are as follows:

- The years 1912 through the current date are supported.
- Any functionality that displays Month and Year does not support Japanese Era dates. These dates are shown in Gregorian format.
- Graphs that display dates do not support the GGGGyy format.

Enter a **Time Format**. This is a "free format" display with some rules.

- **hh** or **h** is interpreted as the hour, 1-12; **KK** or **K** is interpreted as the hour, 0-11; **HH** or **H** is interpreted as the hour, 0-23; **kk** or **k** is interpreted as the hour, 1-24. The **h**, **K**, **H**, and **k** options suppress a leading 0.
- **mm** or **m** is interpreted as the minutes. The **m** option suppresses a leading 0.
- **ss** or **s** is interpreted as the seconds. The **s** option suppresses a leading 0.
- **a** is interpreted to mean display **am** or **pm** (only needed when the hour is entered in **hh**, **h**, **KK** or **K** formats). If an **am** or **pm** is not entered, it defaults to **am**.

Here are some examples of time formats.

Format	Displayed / entered as
hh:mma	09:34PM (can be entered as 09:34p)
hh:mm:ss	21:34:00

---

There are several options for displaying Numbers.

**Decimal Symbol** defines the separator between the integer and decimal parts of a number. Valid values are "." (a period) or "," (a comma),

**Group Symbol** defines the means to separate groups of bigger numbers. Valid values are

- "," (comma). Large numbers group by threes separated by a comma, for example 1,000,000.
- "." (period). Large numbers group by threes separated by a period, for example 1.000.000.
- **None**. Large numbers do not have any separator, for example 1000000.
- **South Asian**. This option uses a comma for its separator but will group large numbers as follows: the first comma is used for the thousands separation and numbers over 9,999 are grouped with 2 units, for example 10,00,000.
- **Space**. Large numbers group by threes separated by a space, for example 1 000 000.

**Negative Format** defines how negative values are displayed. Valid values are **-9.9**, **(9.9)**, or **9.9-**.

**Currency** values can have a different **Negative Format** from other numbers. Valid values are **-S9.9**, **(S9.9)**, or **S9.9-**, where the "S" represents the currency symbol.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_DISP\\_PROF](#).

## Additional Hijri Date Configuration

For implementations that wish to support displaying dates according to the Hijri calendar, besides appropriate configuration in the [Display Profile](#), the mapping between the Hijri dates and the Gregorian dates must be entered. This mapping is defined in a master configuration record.

Navigate using **Admin > System > Master Configuration**. You are shown a list of master configuration business objects. Find the entry for **Hijri to Gregorian Date Mapping**. Click the add icon if there is no record yet. Click the broadcast icon to view an existing record or the edit icon to modify an existing record.

The mapping record contains a collection of entries for each year in the Islamic calendar.

For each year, clicking the Expand Zone icon shows the mapping collection with the first date of each month of the Hijri calendar. The corresponding date in the Gregorian calendar should be entered for each row.

## Defining Phone Types

Phone types define the format for entering and displaying phone numbers.

To add or review phone types, choose **Admin > General > Phone Type**.

#### Description of Page

Enter a unique **Phone Type** and **Description** for each type of phone number you support.

Select an appropriate **Phone Number Format Algorithm** for each **Phone Type**. This algorithm controls the format for entry and display of phone numbers. Click [here](#) to see the algorithm types available for this plug-in spot.

Use **Phone Type Flag** to define if this type of phone number is a **Fax** number. Defining which phone type is used for facsimile transmittal is only pertinent if your product supports routing of information via fax. For example, in Oracle Utilities Customer Care and Billing, the system may be configured to fax a bill to a customer.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_PHONE\\_TYPE](#).

## Setting Up Characteristic Types & Values

If you need to introduce additional fields to objects that were delivered with minimal fields, you can add a characteristic type for each field you want to capture. Using the characteristic type approach allows you to add new fields to objects without changing the database. There are some pages in the system that support a generic list of characteristics. For those pages, no changes are required for users to view and maintain characteristics. Other pages are business object oriented pages and their maintenance and display are controlled by the UI maps defined for the business objects. In those pages the display / maintenance of the characteristics are driven by the design of the business object and its maps.

The topics in this section describe how to setup a characteristic type.

## There Are Four Types Of Characteristics

Every characteristic referenced on an object references a characteristic type. The characteristic type controls the validity of the information entered by a user when they enter the characteristic's values. For example, if you have a characteristic type on user called "skills", the information you setup on this characteristic type controls the valid values that may be specified by a user when defining another user's skills.

When you setup a characteristic type, you must classify it as one of the following categories:

- **Predefined value.** When you setup a characteristic of this type, you define the individual valid values that may be entered by a user. A good example of such a characteristic type would be one on User to define one or more predefined skills for that user. The valid values for this characteristic type would be defined in a discreet list.
- **Ad hoc value.** Characteristics of this type do not have their valid values defined in a discreet list because the possible values are infinite. Good examples of such a characteristic type would be ones used to define a user's birth date or their mother's maiden name. Optionally, you can plug-in an algorithm on such a characteristic type to validate the value entered by the user. For example, you can plug-in an algorithm on a characteristic type to ensure the value entered is a date.
- **Foreign key value.** Characteristics of this type have their valid values defined in another table. For example perhaps you want to link a user to a table where User is not already a foreign key. Valid values for this type of characteristic would be defined on the user table. Please be aware of the following in respect of characteristics of this type:
  - Before you can create a characteristic of this type, information about the table that contains the valid values must be defined on the [foreign key reference table](#).
  - The referenced table does not have to be a table within the system.
  - Not all entities that support characteristics support foreign key characteristics. Refer to the data dictionary to identify the entities that include the foreign key characteristic columns.
- **File Location.** Characteristics of this type contain a URL. The URL can point to a file or any web site. Characteristics of this type might be useful to hold references to documentation / images associated with a given entity. For example, the image of a letter sent to you by one of your customers could be referenced as a file location characteristic on a customer contact entry. When such a characteristic is defined on an entity, a button can be used to open the URL in a separate browser window. Note that for references to a file, the recommendation is to use the Attachment functionality to link documents to an object rather than a characteristic type of File Location. Refer to [Attachment Overview](#) for more information. The documentation related to file location remains for upgrade purposes.

## Searching By Characteristic Values

For certain entities in the system that have characteristics, you may search for a record linked to a given characteristic value. The search may be done in one of the following ways:

- Some base searches provide an option to search for an object by entering Characteristic Type and Characteristic Value.

- Your implementation may define a customized search for an entity by a characteristic value for a specific characteristic type using a query data explorer.
- Your implementation may require a business service to find a record via a given characteristic value. For example, maybe an upload of user information attempts to find the user via an Employee ID, defined as a characteristic.

Not all entities that support characteristics support searching by characteristics. Refer to the data dictionary to identify the characteristic collections that include the search characteristic column.

---

**CAUTION:** For ad-hoc characteristics, only the first 50 bytes are searchable. For foreign key characteristics, the search value is populated by concatenating the values of each foreign key column to a maximum of 50 bytes.

---

For the base searches that provide a generic option to search by characteristic type and value, you can restrict the characteristic types that can be used to search for an entity. For example, imagine you use a characteristic to define a "jurisdiction" associated with a To Do for reporting purposes. If your company operates within a very small number of jurisdictions, you wouldn't want to allow searching for a To Do by jurisdiction, as a large number of To Do entries would be returned.

A flag on the [characteristic type](#) allows an administrator to indicate if searching by this characteristic type is **allowed** or **not allowed**.

## Characteristic Type - Main

To define a characteristic type, open **Admin > System > Characteristic Type > Search**.

### Description of Page

Enter an easily recognizable **Characteristic Type** and **Description** for the characteristic type. **Owner** indicates if this characteristic type is owned by the base package or by your implementation (**Customer Modification**).

---

**CAUTION:** Important! If you introduce a new characteristic type, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

Use **Type of Char Value** to classify this characteristic type using one of the following options (refer to [There Are Four Types Of Characteristics](#) for more information):

- **Predefined value.** Characteristics of this type have their valid values defined in the **Characteristic Value** scroll, below. For each valid value, enter an easily recognizable **Characteristic Value** and **Description**.
- **Ad hoc value.** Characteristics of this type capture free form text. If you use this option, you can optionally define the **Validation Rule** used to validate the user-entered characteristic value. Click [here](#) to see the algorithm types available for this plug-in spot.
- **File location value.** Characteristics of this type contain a URL. The URL can point to a file or any web site. Note that for references to a file, the recommendation is to use the Attachment functionality to link documents to an object rather than a characteristic type of File Location. Refer to [Attachment Overview](#) for more information. The documentation related to file location remains for upgrade purposes.

File location characteristic values must be entered in a "non-relative" format. For example, if you want to define a characteristic value of *www.msn.com*, enter the characteristic value as `http://www.msn.com`. If you omit the `http://` prefix, the system will suffix the characteristic value to the current URL in your browser and attempt to navigate to this location when the launch button is pressed. This may or may not be the desired result.

---

### NOTE:

Due to browser security restrictions, opening URLs using the file protocol ("file://") from pages retrieved using http does not work for Internet Explorer version 7 or later, or in Firefox. If the file protocol is used, the browser either does not return properly or an error is thrown (e.g., "Access Denied", which usually results from cross site scripting features added for security reasons).

This issue has no known workaround. To comply with browser security standards, the recommendation is to move the target files to an FTP or HTTP server location to avoid protocols that are subject to browser security restrictions.

---

- **Foreign key reference.** Characteristics of this type have their valid values defined in another table. If you choose this option, you must use **FK Reference** to define the table that controls the valid values of this characteristic type. Refer to [Setting Up Foreign Key References](#) for more information.

Use the **Allow Search by Char Val** to indicate if searching for an entity by this characteristic type is **Allowed** or **Not Allowed**. Refer to [Searching by Characteristic Values](#) for more information.

The **Custom** switch is only applicable to **Predefined value** types. It indicates whether or not an implementation is allowed to add values for a characteristic type whose owner is not **Customer Modification**

- If this switch is turned on, an implementation may add characteristic values to the grid for system owned characteristic types.
- If this switch is turned off, an implementation may not add characteristic values to the grid for system owned characteristic types.

---

**NOTE:** Regardless of the value of the Custom switch, an implementation may not update or remove system owned characteristic values.

---

The **Characteristic Value** grid defines the valid values for a **Predefined value** type of characteristic.

The **Characteristic Value** is the unique identifier of the value.

**Description** is the text that is visible in the dropdowns and display when viewing this characteristic value.

**Owner** indicates if this characteristic value is owned by the system or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add characteristic values to a characteristic type. This information is display-only.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_CHAR\\_TYPE](#) in the data dictionary schema viewer.

## Characteristic Type - Characteristic Entities

To define the entities (objects) on which a given characteristic type can be defined, open **Admin > System > Characteristic Type > Search** and navigate to the **Characteristic Entities** tab.

#### Description of Page

Use the **Characteristic Entity** collection to define the entities on which the characteristic type can be used. **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

---

**NOTE:** The values for this field are customizable using the Lookup table. This field name is **CHAR\_ENTITY\_FLG**.

---

**NOTE:** For some entities in the system, the valid characteristics for a record are defined on a related "type" entity. For example, in Oracle Utilities Customer Care and Billing the meter type defines valid characteristics for meters of that type. When configuring your system, in addition to defining the appropriate entity for a characteristic type, you may also need to link the characteristic type to an appropriate entity "type". This technique is typically not followed for business object driven maintenance objects, where the business objects can be configured with the appropriate "flattened" characteristic types in the schema.

---

## Setting Up Foreign Key Reference Information

A Foreign Key Reference defines the necessary information needed to reference an entity in certain table.

You need to set up this control table if you need to validate a foreign key value against a corresponding table. For example, if a schema element is associated with an FK Reference the system validates the element's value against the corresponding table. Refer to [Configuration Tools](#) to learn more about schema-based objects. Another example is characteristics whose valid values are defined in another table (i.e., you use "foreign key reference" characteristic types). Refer to [There Are Four Types Of Characteristics](#) for a description of characteristics of this type.

A FK Reference is used not just for validation purposes. It also used to display the standard information description of the reference entity as well as provide navigation information to its maintenance transaction. Info descriptions appear throughout the UI, for example, whenever an account is displayed on a page, a description of the account appears. The product provides base product FK references for many of its entities as they are used for validation and display of elements in both fixed page user interfaces as well as portal based user interfaces.

An implementation may also see the need to define a foreign key reference. The following points describe what you should know before you can setup a foreign key reference for a table.

- The physical name of the table. Typically this is the primary table of a maintenance object.
- The program used by default to construct the referenced entity's info description. Refer to [Information Description Is Dynamically Derived](#) for more information on how this is used.
- The transaction used to maintain the referenced entity. This is where the user navigates to when using the "go to" button or hyperlink associated with the entity. Refer to [Navigation Information Is Dynamically Derived](#) for more information on how this is used.
- The name of the search page used to look for a valid entity. Refer to [Search Options](#) for more information.

## Information Description Is Dynamically Derived

Typically a FK Reference is defined for a maintenance object's primary table. In this case the system dynamically derives the standard information associated with a specific referenced entity as follows:

- Attempt to determine the business object associated with the referenced entity. Refer to the [Determine BO](#) maintenance object algorithm system event for more information. If a business object has been determined, the system lets the business object's [Information](#) plug-in, if any, format the description.
- If a business object has not been determined or the business object has no such plug-in, the system lets the maintenance object's [information](#) plug-in, if any, format the description.
- If the maintenance object has no such plug-in, the system uses the info program specified on the FK Reference to format the description.

---

**NOTE: Technical note.** The class that returns the information displayed adjacent to the referenced entity is generated specifically for use as an info routine. Please speak to your support group if you need to generate such a class.

---

**NOTE: Generic routine.** The system provides a generic information routine that returns the description of control table objects from its associated language table. By "control table" we mean a table with an associated language table that contains a **DESCR** field. Refer to [Defining Table Options](#) for more information on tables and fields. The java class is `com.splwg.base.domain.common.foreignKeyReference.DescriptionRetriever`.

---

## Navigation Information Is Dynamically Derived

Typically a FK Reference is defined for a maintenance object's primary table. In this case the system dynamically derives the actual transaction to navigate to for a given referenced entity as follows:

- Attempt to determine the business object associated with the referenced entity. Refer to the [Determine BO](#) maintenance object algorithm system event for more information. If a business object has been determined, use the maintenance portal defined as its **Portal Navigation Option** business object option.
- If a business object has not been determined or the business object defines no such option, the system uses the transaction specified on the FK Reference.

## Search Options

The product provides two main metaphors for implementing a user interface. For input fields that are foreign keys, search options are dependent on the metaphor used by the page in question.

- A [fixed maintenance page](#) user interface is a page supplied by the base product where only minor enhancements, if any, can be introduced by implementations. The foreign key reference may be used in one of two ways.
  - The based product may use an FK reference to define a base element on one of these pages. If a search is available for such elements, the FK reference's Search Navigation Key is used to implement the search.
  - Entities that support characteristics typically include a generic characteristic collection UI metaphor on these types of pages. In this metaphor, a foreign key characteristic displays a search icon if the FK Reference has configured a Search Navigation Key.
- A [portal based](#) user interface is a more flexible user interface where an implementation has more options for customizing the look and feel. The base product uses UI maps or automatic UI rendering to display input fields. Elements that are foreign keys may display a search icon if the FK reference defines a Search Zone.

---

**NOTE: Defining search zones directly.** It's possible for elements on a UI map to define a specific search zone directly in the HTML, rather than using the search zone defined on an FK reference. Refer to the UI map tips for more information on implementing searches using zones.

---

## Foreign Key Reference - Main

To setup a foreign key reference, open **Admin > Database > FK Reference > Search**.

---

**CAUTION:** Important! If you introduce a new foreign key reference, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

### Description of Page

Enter an easily recognizable **FK** (foreign key) **Reference** code and **Description** for the table.

Enter the name of the **Table** whose primary key is referenced. After selecting a **Table**, the columns in the table's primary key are displayed adjacent to **Table PK Sequence**.

Use **Navigation Option** to define the page to which the user will be transferred when they press the go to button or hyperlink associated with the referenced entity. Refer to [Navigation Information Is Dynamically Derived](#) for more information on how this is used.

The **Info Program Type** indicates whether the default program that returns the standard information description is **Java** or **Java (Converted)**, meaning it was converted into Java.



---

**NOTE:** **Java (Converted)** program types are not applicable to all products.

---

Use **Info Program Name** to enter the Java class / program name.

Refer to [Information Description Is Dynamically Derived](#) for more information on the info program is used.

---

**NOTE: View the source.** If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the [Java docs viewer](#).

---

Use **Context Menu Name** to specify the context menu that appears to the left of the value.

---

**NOTE:** Context Menu Name is not applicable to user interface elements displaying a generic collection using a foreign key characteristic type. It is only applicable for pages utilizing the foreign key compound element type for fixed page user interface and for data displayed in a portal based user interface where the foreign key reference is defined as an attribute for an element. Report parameters that reference foreign key characteristics are an example of a user interface where a context menu is not displayed even if the foreign key reference defines one.

---

Use **Search Navigation Key** to define the search page that will be opened when a user searches for valid values on a user interface that is a fixed page. Refer to [Search Options](#) for more information.

Use **Search Type** to define the default set of search criteria used by the **Search Navigation Key**'s search page.

Use **Search Zone** to define the search zone that opens when a user searches for valid values when the foreign key reference is configured as an input field on a portal based page. Refer to [Search Options](#) for more information.

Use **Search Tooltip** to define a label that describes the **Search Navigation Key**'s search page.

---

**NOTE: Search Type and Search Tooltip.** These attributes are only applicable to user interface elements utilizing the foreign key compound element type on fixed page user interfaces. Report parameters that reference foreign key characteristics are an example of a user interface where this information is not used even if the foreign key reference defines them.

---

## Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_FK\\_REF](#).

## Defining Feature Configurations

Some system features are configured by populating options on a "feature configuration". Because various options throughout the system may be controlled by settings in feature configuration, this section does not document all the disparate possible options. The topics below simply describe how to use this transaction in a generic way.

For information about specific features:

- Refer to the detailed description of each option type.
- Use the index in the online help and search for 'feature configuration' to find any specific topics describing feature options in the administration guide.

You can create options to control features that you develop for your implementation. To do this:

- Review the lookup values for the lookup field **EXT\_SYS\_TYP\_FLG**. If your new option can be logically categorized within an existing feature type, note the lookup value. If your new option warrants a new feature type, add a lookup value to this lookup field.
- Define the feature's option types. If you have identified an existing feature type to add the options to, find the lookup with the name **xxxx\_OPT\_TYP\_FLG** where **xxxx** is the lookup value of **EXT\_SYS\_TYP\_FLG** noted above. If you decided to create a new feature type (by adding a new lookup value to the **EXT\_SYS\_TYP\_FLG** lookup, you must create a new lookup with the name **xxxx\_OPT\_TYP\_FLG** where **xxxx** is the new value you defined above.



- Flush all caches.

## Feature Configuration - Main

To define your feature configuration, open **Admin > System > Feature Configuration > Search**.

### Description of Page

Enter an easily recognizable **Feature Name** code.

Indicate the **Feature Type** for this configuration. For example, if you were setting up the options for the batch scheduler, you'd select **Batch Scheduler**.

---

**NOTE: You can add new Feature Types.** Refer to the description of the page above for how you can add Feature Types to control features developed for your implementation.

---

---

**NOTE: Multiple Feature Configurations for a Feature Type.** Some Feature Types allow multiple feature configurations. The administration documentation for each feature will tell you when this is possible.

---

The **Options** grid allows you to configure the feature. To do this, select the **Option Type** and define its **Value**. Set the **Sequence** to **1** unless the option may have more than value. **Detailed Description** may display additional information on the option type.

---

**NOTE: Each option is documented elsewhere.** The administration documentation for each feature describes its options and whether an option supports multiple values.

---

---

**NOTE: You can add new options to base-package features.** Your implementation may want to add additional options to one of the base-package's feature types. For example, your implementation may have plug-in driven logic that would benefit from a new option. To do this, display the lookup field that holds the desired feature's options. The lookup field's name is **xxxx\_OPT\_TYP\_FLG** where **xxxx** is the identifier of the feature on the **EXT\_SYS\_TYP\_FLG** lookup value. For example, to add new batch scheduler options, display the lookup field **BS\_OPT\_TYP\_FLG**.

---

## Feature Configuration - Messages

If the feature exists to interface with an external system, you can use this page to define the mapping between error and warning codes in the external system and our system.

Open this page using **Admin > Feature Configuration > Search** and navigate to the **Messages** tab.

### Description of Page

For each message that may be received from an external system, define the **Feature Message Category** and **Feature Message Code** to identify the message.

A corresponding message must be defined in the [system message](#) tables. For each message identify the **Message Category** and **Message Number**. For each new message, the Message Category defaults to **90000** (because an implementation's messages should be added into this category or greater so as to avoid collisions during upgrades).

## Defining Master Configurations

A master configuration is an object that enables an implementation to define configuration for features in the system. It is an alternative to using feature configuration for defining options. A master configuration is defined using a business object. Only one master configuration may exist for a given business object.

The product provides one or more master configuration that may be used for configuration. Some examples, of base master configuration business objects are as follows

- **Hijri to Gregorian Date Mapping.** This allows an implementation that uses Hijri dates to define the mapping between those dates and Gregorian dates.
- **ILM Configuration.** For implementations that use Information Lifecycle Management, the ILM configuration record defines some parameters used by the process.
- **Migration Assistant Configuration.** For implementations that use the configuration migration assistant (CMA), the configuration record defines some parameters used by the process.

For a list of all the master configuration records provided by the product, navigate to the master configuration page in the application.

## Setting Up Master Configurations

To set up a master configuration, open **Admin > System > Master Configuration**.

The topics in this section describe the base-package zones that appear on the Master Configuration portal.

### Master Configuration

The Master Configuration List zone lists every category of master configuration.

The following functions are available:

- Click a broadcast button to open other zones that contain more information about the adjacent master configuration.
- Click the **Add/Edit** button to start a business process that updates the master configuration.

### Master Configuration Details

The Master Configuration Details zone contains display-only information about a master configuration.

This zone appears when a master configuration has been broadcast from the Master Configuration zone.

Please see the zone's help text for information about this zone's fields.

## Miscellaneous Topics

The following sections describe miscellaneous system wide topics.

## Module Configuration

The system provides the ability to simplify the user interface based on functionality areas practiced by your organization.

Menu items and other user interface elements are associated with function modules. By default, all function modules are accessible. If a function module is not applicable to your business you may turn it off. Refer to [Turn Off A Function Module](#) for more information on how to turn off a module.

If a function module is made non-accessible, i.e. turned off, its related elements are suppressed from the user interface. In addition the system may validate that related functionality is not accessed. This also means that turning off the wrong module may cause any of the following to occur:

- Menu items may not appear. Refer to [Menu Item Suppression](#) to better understand how menu item suppression works.
- Entire menus may not appear. Refer to [Menu Suppression](#) to better understand how menu suppression works.
- Tabs on pages may not appear.
- Fields may not appear.

- The system may return an error message when you attempt to use a function (indicating the function is turned off).

To correct the above situation, simply remove the module from the turned off list thus making it accessible again.

Your module configuration setup is displayed on the [installations](#) record.

## Menu Item Suppression

The following points describe how your module configuration can suppress [menu items](#).

- Menu items that are owned by the base product (as opposed to those your implementation adds) are associated with one or more function modules. If your module configuration has turned off all of the menu item's modules, the menu item is suppressed. If at least one of the modules is accessible, i.e. turned on, the menu item is not suppressed.
- If a menu line doesn't contain any accessible items, the menu line is suppressed.
- If all lines on a menu are suppressed, the menu itself (**Menu** or **Admin menu**) is suppressed in the application toolbar.

## Menu Suppression

In addition to the above Menu Item Suppression logic, the following points describe how your module configuration can suppress an entire menu.

- Menus that are owned by the base product (as opposed to those your implementation adds) are associated with one or more function modules.
- If your module configuration has turned off all of the menu's modules, the entire menu is suppressed. If at least one of the modules is accessible, i.e. turned on, the menu is not suppressed.

## Turn Off A Function Module

The base package is provided with a **Module Configuration** [Feature Configuration](#) that allows your organization to turn off base package function modules.

To turn off any of the base package function modules add a **Turned Off** option to this feature configuration referencing that module. Refer to the **MODULE\_FLG** lookup field for the complete list of the application's function modules.

Any module not referenced on this feature configuration is considered turned on, i.e. accessible. To turn on a module, simply remove its corresponding **Turned Off** option from this feature configuration.

You may view your module configuration setup on the [installation options](#) page.

---

**NOTE: Only one.** The system expects only one **Module Configuration** feature configuration to be defined.

---

## Global Context Overview

The framework web application provides each product the ability to nominate certain fields to act as a "global context" within the web application. For example, in Oracle Utilities Customer Care and Billing, the global context fields include Account ID, Person ID and Premise ID. The values of these fields may be populated as a result of searching or displaying objects that use these fields in their keys. If you navigate to the Bill page and display a bill, the global context is refreshed with the Account ID associated with that bill. The global context for Person ID and Premise ID are refreshed with data associated with that account.

The fields designated as global context for the product are defined using the lookup **F1\_UI\_CTXT\_FLDS\_FLG**.

Changing the values of the global context typically cause data displayed in zones on the dashboard to be refreshed to show information relevant to the current values of these global context fields.

When the value of one of the global context fields changes, an algorithm plugged into the [installation record](#) is responsible for populating the remaining global context values accordingly. Refer to your specific product for more information about the base algorithm that is provided for that product.

## System Data Naming Convention

There are several maintenance objects in the system that include owner flag in one or more of its tables. We refer to the data in these tables as "system data". Some examples of system data tables include Algorithm Type, Batch Control, Business Object and Script. Implementations may introduce records to the same tables. The owner flag for records created by an implementation is set to **CM** (for customer modification), however the owner flag is not part of the primary key for any of the system data tables. As a result, the base product provides the following guidelines for defining the primary key in system data tables to avoid any naming conflict.

## Base Product System Data

For any table that includes the owner flag, the base product will follow a naming convention for any new data that is owned by the base product. The primary key for records introduced by the product is prefixed with **xn-** where **xn** is the value of the owner flag. For example, if a new background process is introduced to the framework product, the batch code name is prefixed with **F1-**.

---

**NOTE:** There are some cases where the hyphen is not included. For example, portal codes omit the hyphen.

---

For most system data, the remainder of the primary key is all in capital case. An exception is schema oriented records. For business objects, business services, scripts, data areas and UI maps, the product follows the general rule of using CapitalCase after the product owner prefix. For example, **F1-AddToDoEntry** is the name of a base product business service.

---

**NOTE: Data Explorer Business Services.** For business services used to invoke a data explorer zone, it is recommended to name the Business Service the same name as the related zone rather than defining a different CapitalCase name for the business service.

---

Please note that this standard is followed for all new records introduced by the base product. However, there are base product entries in many of these system data tables that were introduced before the naming convention was adopted. That data does not follow the naming convention described above.

---

**NOTE: Schema naming conventions.** A context sensitive "Schema Tips" zone is associated with any page where a schema may be defined. The zone provides recommended naming conventions for elements within a schema along with a complete list of the XML nodes and attributes available to you when you construct a schema.

---

## Implementation System Data

When new system data is introduced for your implementation you must consider the naming convention for the primary key. The product recommends prefixing records with **CM**, which is the value of the owner flag in your environment. This is consistent with the base product naming convention. This convention allows your implementation to use the CM packaging tool in the Software Development Kit as delivered. The extract file provided with the tool selects system data records with an owner flag of **CM AND** with a **CM** prefix.

---

**NOTE:** If you choose not to follow the CM naming convention for your records and you want to use the CM packaging tool, your implementation must customize the extract file to define the appropriate selection criteria for the records to be included in the package. Refer to the Software Development Kit documentation for more information.

---

Also note that owner flag may be introduced to an existing table in a new release. When this happens, the CM packaging tool is also updated to include these new system data tables. Your implementation will have existing records in those tables that probably do not follow any naming convention. After an upgrade to such a release, if you want to include this data in the CM packaging tool, you must customize the extract file for the tables in question.

## Caching Overview

A great deal of information in the system changes infrequently. In order to avoid accessing the database every time this type of information is required by an end-user or a batch process, the system maintains a cache of static information on the web server. In addition to the web server's cache, information is also cached on each user's browser.

### Server Cache

The cache is populated the first time any user accesses a page that contains cached information. For example, consider a control table whose contents appear in a dropdown on various pages. When a user opens one of these pages, the system verifies that the list of records exists in the cache. If so, it uses the values in the cache. If not, it accesses the database to retrieve the records and saves them in the cache. In other words, the records for this control table are put into the cache the first time they are used by any user. The next user who opens one of these pages will have the records for this control table retrieved from the cache (thus obviating the database access).

The following points describe the type of data that is cached on the web server:

- **Field labels.** This portion of the cache contains the labels that prefix fields on the various pages in the system.
- **System information.** This portion of the cache contains installation and basic information about the various application services (e.g., the URL's that are associated with the various pages).
- **Menu items.** This portion of the cache contains the menu items.
- **Dropdown contents.** This portion of the cache contains the contents of the various dropdowns that appear throughout the system.
- **XSL documents.** This portion of the cache contains each page's static HTML.
- **Portal information.** This portion of the cache contains information about which zones are shown on the various pages.

The contents of the cache are cleared whenever the web server is "bounced". This means that fresh values are retrieved from the database after the application server software is restarted.

If you change the database after the cache is built and the information you changed is kept in the cache, users may continue to see the old values. If you don't want to bounce your web server, you can issue one of the following commands in your browser's URL to immediately clear the contents of the cache (a separate command exists for each type of data that is held in the cache):

- **flushAll.jsp?language=<language code>.** This command flushes every cache described below.
- **flushDropdownCache.jsp?language=<language code>.** This command flushes all objects in the dropdown contents associated with a given language (note, the **language code** is defined on the [Language](#) control table). Use this whenever you change, add or delete values to/from control tables that appear in dropdowns. Unlike the above caches, the objects in the dropdown cache are flushed automatically every 30 minutes from the time they are first built.
- **flushDropDownField.jsp?language=<language code>&key=<field>** If you want to flush the values in a specific dropdown field in the cache, specify the *<field>* using this command.
- **flushMenu.jsp.** This command flushes the menu cache. Use this command if you change menu data.
- **flushMessageCatalog.jsp.** This command flushes the field labels. Use this whenever you add or change any labels (this is typically only done by implementers who have rights to introduce new pages).
- **flushMessaging.jsp.** This command flushes messages.
- **flushNavigationInfo.jsp.** This command flushes the navigation information in the cache.

- **flushPortalMetaInfo.jsp**. This command flushes the portal information cache. Use this command whenever you change any portal zone meta-data.
- **flushSystemLoginInfo.jsp**. This command flushes the system information cache. Use this whenever you change navigation options, cached information from the installation record, or if you change an application service's URL.
- **flushUI\_XSLs.jsp**. This command flushes the XSL document cache. Use this command if you change user interface meta-data. Also use this command whenever you change or introduce new zones.

For example, assume the following:

- the web server and port on which you work is called **OU-Production:7500**
- you add a new record to a control table and you want it to be available on the appropriate transactions immediately (i.e., you cannot wait for 30 minutes)

You would issue the following command in your browser's address bar: **http://OU-Production:7500/flushDropdownCache.jsp?language=ENG**. Notice that the command replaces the typical `cis.jsp` that appears after the port number (this is because these commands are simply different JSP pages that are being executed on the web server).

## Client Cache

In addition to the web server's cache, information is also cached on each user's browser. After clearing the cache that's maintained on the web server, you must also clear the cache that's maintained on your client's browser. To do this, follow the following steps:

- Select **Tools** on your browser's menu bar
- Select **Internet Options...** on the menu that appears.
- Click the **Delete Files** button on the pop-up that appears.
- Turn on **Delete all offline content** on the subsequent pop-up that appears and then click **OK**.
- And then enter the standard URL to re-invoke the system.

---

**NOTE: Automatic refresh of the browser's cache.** Each user's cache is automatically refreshed based on the **maxAge** parameter defined in the `web.xml` document on your web server. We recommend that you set this parameter to **1** second on development / test environments and **28800** seconds (8 hours) on production environments. Please speak to system support if you need to change this value.

---

## Batch Cache

When submitting a batch job, the batch component uses a caching mechanism using a Hibernate data cache. The tables whose records are included in this cache are configured using the Caching Regime value of **Cached for Batch**. Refer to [Table - Main](#) for more information.

When starting a thread pool worker, data in tables marked as cached is loaded and cached for as long as that thread pool is running. If there is a change in cached data that should be available for the next batch job, either the thread pool worker must be restarted or alternatively, the **F1-FLUSH** (Flush all Caches) background process can be submitted. This background process receives the thread pool name as input. It will flush the data cached for batch for that thread pool.

## Debug Mode

Your implementation team can execute the system using a special mode when they are configuring the application. To enable this mode, enter **?debug=true** at the end of the URL that you use to access the application. For example, if the standard URL was **http://CD-Production:7500/cis.jsp**, you'd enter **http://CD-Production:7500/cis.jsp?debug=true** to enable configuration mode.

When in this mode certain debugging oriented tools become available right below the main toolbar.

- **Start Debug** starts a logging session. During this session the processing steps that you perform are logged. For example, the log will show the data areas that are passed in at each step and the data areas returned after the step is processed.
- **Stop Debug** stops the logging session.
- **Show Trace** opens a window that contains the logging session. All of the steps are initially collapsed.
- **Clear Trace** clears your log file.
- **Show User Log** allows you to view your own log entries. The number of "tail" entries to view may be specified in the adjacent **Log Entries** field before clicking the button. Limiting the number of entries to view allows the user to quickly and easily see only the latest log entries without having to manually scroll to the end of the log.
- Checking the **Global Debug** indication starts various tracing options.

Other parts of the system may show additional configuration oriented icons when in this mode. For example, explorer zones may provide additional tools to assist in debugging zone configuration. These icons are described in the context of where they appear.

Also, in debug mode drop down lists in data explorer and UI map zones will contain the code for each item in addition to the item's display string.

---

**NOTE: Show User Log button is secured.** An application service **FIUSERLOG** has been provided for this functionality to allow implementations to restrict user access to this button. Such restriction may be called for in production environments.

---

## System Override Date

The system provides a way to override the system date used for online operations. This feature is available if the server administrator has enabled it in the environment properties. For instructions on configuring environment properties see the *Server Administration Guide*. The system date override feature is not recommended for production environments.

Under the **General System Configuration** [Feature Configuration](#), the **System Override Date Option Type** holds the date the application will use as the global system date instead of retrieving the same from the database. This feature can be especially useful in running tests that require the system date to be progressed over a period of time.

The system override date feature is also available at the user level. This is useful when a user wants override the system date to run tests without affecting the system date for other users in the environment. In order to override the system date for the user, open the **Admin > User > Search**, page. On the **Characteristics** tab, add the **System Override Date** characteristic type with a characteristic value set to the desired date in the YYYY-MM-DD format.

If system override dates are defined at both the feature configuration level and the user level, the date set at the user level will take precedence.

## Advanced Search Options

The product supports fuzzy searching in explorer zone types using the Oracle Text CONTAINS operator.

Refer to the DBA guide for details on setting up the database to support fuzzy searching. Note that there are some implementations where fuzzy searching will not be possible. For example, it's only available for implementations using the Oracle database. Additionally, not all languages are supported. Refer to the Oracle Database documentation for more information about fuzzy searching.

For information about the particular syntax to use in the explorer zones, refer to the Zone Tips context zone available in the Dashboard on the Zone portal.



# Defining Security & User Options

---

The contents of this section describe how to maintain a user's access rights.

## The Big Picture of Application Security

The contents of this section provide background information about application security.

### Application Security

The system restricts access to its transactions as follows:

- An [application service](#) may be associated with every securable function in the system.
  - All maintenance objects define an application service that includes the basic actions available, typically **Add**, **Change**, **Delete**, and **Inquire**. The base product supplies an application service for every maintenance object.
  - For maintenance objects whose user interface page is not portal-based, the application service also controls whether the menu entry appears. If a user doesn't have access to the maintenance object's application service, the menu item that corresponds with the application service will not be visible.
  - For portal based user interfaces, each main portal defines an explicit application service with the access mode **Inquire**, allowing the user interface to be secured independently of the underlying object security. If a user doesn't have access to the portal's application service, the menu item that corresponds with the application service will not be visible. The base product supplies an application service for every portal that is accessible from the menu.
  - Menu items may define an application service. Use this technique for the following scenarios:
    - Suppress a menu item if the underlying application security for the transaction does not provide enough fine grained control. For example, imagine your implementation creates a special BPA script to add a To Do Entry and would like users to use the special BPA rather than the base supplied Add dialogue for To Do Entry. The underlying security settings for To Do Entry should grant Add access to these users given that the special BPA will still add a record. To suppress the base Add dialogue, link a special application service and access mode for the base supplied menu item for To Do Entry Add. Then define a menu entry for the new special BPA for adding.
    - Suppress the add option if a user does not have add security for the object. By default the product does not suppress the add function if a user does not have add access to the object. Rather, the user is prevented from adding the record at the back-end. If your implementation would like to suppress the icon, link the object's application service and the Add access mode to the Add menu item.
- Zones define an application service
  - For zones linked to a portal, if a user doesn't have access to the zone's application service, the zone will not be visible on the portal. In most cases the zone defines the same application service as its portal. In special cases, such as the zones on the Dashboard, the product supplies separate application services for each zone allowing implementations to determine at a more granular level which users should have access to which zones.
  - For query zones that are configured on a multi-query zone, if a user doesn't have access to the zone's application service, the zone will not be visible in the dropdown on the multi-query zone. In most cases all zones in a multi-query zone define the same application service as the multi-query zone. The product may supply a special application service for one or more zones in a multi-query zone if the functionality is special to certain markets or jurisdictions and not applicable to all implementations.
  - For zones that are used by business services to perform SQL queries, the product supplies a default application service. Security for these zones is not checked by the product as they are used for internal purposes.



- Business objects define an application service. If the business object defines a lifecycle, the application service must include access modes that correspond to each state. In addition, the standard maintenance object access modes of **Add, Change, Delete** and **Inquire** are included. The base product business objects are supplied with appropriate application services.
- Other configuration tool objects are securable but the base product typically does not supply special application services for each object. An implementation may supply custom application services and link them to the appropriate record:
  - BPA scripts may define an application service with the access mode **Execute**. The base BPA scripts are typically not configured with any application service. An implementation may define one. Note that as mentioned above, a menu item may also be configured with an application service and access mode. This allows for a BPA that is invoked via a menu entry to be secured in more than one way.
  - Business Services and Service Scripts define an application service with the access mode **Execute**. This is needed for services that may be executed from an external system, for example via an inbound web service. Base business services and service scripts are configured with a default application service, which may be overridden by an implementation.
- Users are granted access to application services via [user groups](#). For example, you may create a user group called Senior Management and give it access to senior manager-oriented pages and portals.
  - When you grant a user group access to an application service with multiple access modes, you must also define the access modes that are allowed. Often the access modes correspond to an action on a user interface. For example, you may indicate a given user group has **inquire**-only access to an application service, whereas another user group has **add, change, cancel** and **complete** access to the same service. Refer to [action level security](#) for more information.
  - If the application service has [field level security](#) enabled, you must also define the user group's security level for each secured field on the transaction.
  - And finally, you link individual [users](#) to the user groups to which they belong. When you link a user to a user group, this user inherits all of the user group's access rights.

---

**CAUTION:** Menus may be suppressed! If all menu items on a menu are suppressed, the menu is suppressed.

---

## Action Level Security

When you grant a user group access to an [application service](#), you must indicate the actions to which they have access.

- For application services that only query the database, there is a single action to which you must provide access - this is called **Inquire**.
- For application services that can modify the database, you must define the actions that the user may perform. At a minimum, most maintenance transactions support **Add, Change, and Inquire** actions. Additional actions are available depending on the application service's functions.

---

**CAUTION:** Important! If an application service supports actions that modify the database other than **Add, Change, and Delete**; you must provide the user with **Change** access in addition to the other access rights. Consider a transaction that supports special actions in addition to **Add, Change, and Inquire** (e.g., **Freeze, Complete, Cancel**). If you want to give a user access to any of these special actions, you must also give the user access to the **Inquire and Change** actions.

---

## Field Level Security

Sometimes transaction and action security is not sufficient. There are situations where you may need to restrict access based on the values of data. For example, in Oracle Utilities Customer Care and Billing you might want to prevent certain users from completing a bill for more than \$10,000. This is referred to as "field level security".

Field level security can be complex and idiosyncratic. Implementing field level security always requires some programming by your implementation group. This programming involves the introduction of the specific field-level logic into the respective application service(s).

---

**NOTE: Field level security logic is added to user exits.** Refer to the Public API chapter of the Software Development Kit Developer Guide for more information on how to introduce field-level security logic into an application service's user exits.

---

Even though the validation of a user's field-level security rights requires programming, the definition of a user's access rights is performed using the same transactions used to define transaction / action level security. This is achieved as follows:

- Create a [security type](#) for each type of field-level security.
- Define the various access levels for each security type. For example, assume you have some users who can complete bills for less than \$300, and other users who can complete bills for less than \$1,000, and still other users who can complete bills for any value. In this scenario, you'd need 3 access levels on this security type:
  - Level 1 (lowest): May authorize bills <= \$300
  - Level 2 (medium): May authorize bills <= \$1,000
  - Level 3 (highest): May authorize all bills
- Link this security type to each [application service](#) where this type of field level security is implemented. This linkage is performed on the [security type](#) transaction.
- Defining each [user group's](#) access level for each security type (this is done for each application service on which the security type is applicable).

---

**NOTE:**

Highest value grants highest security. The system expects the highest authorization level value to represent highest security level. Moreover, authorization level is an alphanumeric field so care should be taken to ensure that it's set up correctly.

---

## Encryption and Masking

"Encryption" refers to encrypting data stored in a database using an encryption key.

"Masking" refers to overwriting all or part of a decrypted field value with a masking character before it is presented to a user (or an external system) without the appropriate security access. For example, an implementation can mask the first 12 digits of a credit card number with an asterisk for users who do not have security rights to see credit card numbers. This section is describing defining masking rules that may be applied to data for display on the user interface.

**Multiple masking rules.** The system allows different masking rules to be applied to different fields. For example, a credit card number can be masked differently than a social security number. In addition, some user groups may be allowed to see certain fields unmasked.

**Masking happens after decryption.** It is obvious, but worth emphasizing, that only decrypted data can be masked. This means that if a user does not have authority to retrieve decrypted data then masking is not relevant because the data to be masked would be encrypted.

The topics in this section describe how to mask field values.

## Identify the Fields to Be Masked

Your implementation should list every field on every page and inbound web service request that requires masking

Classify each field into one of the following categories:

- An element that is retrieved by invoking a business object, a business service, or a service script
- A field that is retrieved by invoking a page service
- A field that is retrieved by invoking a search service
- An ad hoc characteristic type's value

**Primary keys cannot be masked.** A field defined as a unique identifier of a row cannot be configured for masking. Masking a field that is part of the primary key causes a problem when attempting to update the record. This restriction also applies to elements that are part of a "list" in an XML column on a maintenance object. One or more elements in the list must be defined as a primary identifier of the list. Be sure that primary key elements in the list are not ones that require masking.

**Masking applies to strings.** Only fields with a data type of String can be masked.

**List members that contain different "types".** Consider a page with a list that contains a person's phone numbers. You can set up the system so that a person's home phone has different masking rules than their work number. If your implementation has this type of requirement, the list of masked fields should contain an entry for each masking rule.

## Create a Security Type For Each Logical Field

Examine the list of masked fields and look for "logical fields".

For example, assume your list contains the following masked fields:

- The Person - Main page retrieves information using the person page service. This page contains a grid that contains the various forms of ID associated with the person. Entries in the grid with an ID Type of "Social Security Number" are subject to masking.
- The Control Central - Main page retrieves data by invoking a search service. This service shows a person's primary form of ID in the search results. If a person's primary ID is their social security number then it is subject to masking.
- The Control Central - Account Information page contains a map zone that retrieves data by invoking a service script. One of the elements in this script's schema holds the person's social security number and it is subject to masking.

In the above example, there is a single "logical field" associated with the three secured elements: the social security number.

Examine your list and define the distinct logical fields. For each one, create a security type with two authorization levels:

- **1** - Can only see the element masked
- **2** - Can only see the element unmasked

You should link all of the security types to an application service of your choosing. We recommend linking every masking-oriented security type to a single application service (e.g., **CM\_MASK**) as it makes granting access easier.

## Create An Algorithm For Each Security Type

A masking algorithm must be created for each security type.

These algorithms determine if a user has the rights to view a given field unmasked, and, if not, how the field should be masked.

The base package provides the algorithm type **F1-MASK** whose parameters are designed to handle most masking needs. This algorithm type's parameters are described below to provide the complete picture of how to control how a field is masked and who can see it unmasked:

- **Masking Character:** Enter the character used to overwrite a field's value for users who can only see masked values.
- **Number of Unmasked Characters:** If some of the field value should remain unmasked at the end of the string, enter the number of unmasked characters.

- **Unmasked Characters:** If some characters in a field value should never be masked, enter them. For example, if a phone number can contain dashes and parenthesis and you don't want these characters masked, you would enter - ( ) . .
- **Application Service:** Enter the application service that you created above.
- **Security Type:** Enter the security type that you created above.
- **Authorization Level:** Enter the authorization level that a user must have to see this field unmasked. If you followed the recommendations above, this will be 2.

## Create A Feature Configuration For Each Secured Element

Create a feature configuration with a Feature Type of **Data Masking**.

Add an option for every field that you defined in [Identify The Fields To Be Masked](#).

Each field's option value will have an **Option Type** of **Field Masking** and a **Value** that references the respective algorithm defined above. In addition, the Value will contain mnemonics that differ depending on how the field is retrieved.

---

**NOTE:** Only fields defined as strings are supported.

---

### Schema Based Object Field Masking

For data that is accessed via a schema-based object call and displayed in a UI map, the field to be masked must reference a meta-data field name in its schema definition: **field="fld\_name", alg="algorithm name"**

If the element references an mdField in the schema, that is the field used to identify the masking rule. If there is no mdField reference but only a mapField reference, that is the field used to identify the masking rule. For example, if you want to mask a credit card number, let's assume that field is defined in the schema is the following:

```
<creditCard mdField="CCNBR" mapField="EXT_ACCT_ID"/>
```

In this case, the option value should be **field="CCNBR", alg="algorithm name"**. An option value of **field="EXT\_ACCT\_ID", alg="algorithm name"** would not result in masking.

A "where" clause may also be specified. This is useful for data that resides in a list where only data of a certain type needs to be masked: **field="fld\_name", alg="algorithm name", where="fld\_name='value'"**

For example, person can have a collection of IDs and only IDs of type 'SSN' (social security number) should be masked. If the person data including its collection of person IDs is displayed on a UI map via a business object call, let's assume the collection is defined in the following way:

```
<personIds type="list" mapChild=CI_PER_ID">
  <isPrimaryId mapField="PRIM_SW"/>
  <idType mapField="ID_TYPE_CD"/>
  <personIdNumber mapField="PER_ID_NBR"/>
</personIds>
```

The option value may look like this: **field="PER\_ID\_NBR", alg="algorithm name", where="ID\_TYPE\_CD='SSN'"**

Please note the following important points for schema based masking:

- **Limitation of 'where' field** Although the main use of a 'where' clause for schema oriented elements is to mask certain elements in a list based on a 'type', it is also possible to mask a single field in the schema based on the value of another field. For example, imagine that a customer submits a registration form that defines an ID type and ID value. Although this data is not in a list, the implementation may still want to only mask the ID value if the ID type is "SSN". The framework is only able to mask an element in the schema based on a 'where' clause if the element in the 'where' clause is a "sibling" in the schema.
  - If the element to be masked is in a list, the element in the 'where' clause must be in the same list.
  - If an element to be masked maps to a real column in a table, the element in the 'where' clause must also map to a real column in the table.

- If an element to be masked maps to the CLOB of a table as a single element, the element in the 'where' clause must map to the same CLOB as a single element.
- **Multiple feature option entries for the same field.** It's possible that different schemas in the system have a similar type of data that may be masked based on different conditions. For example, imagine that an implementation has different schemas that captured or referenced person identifiers in different ways:

- One schema captures a single person ID without any corresponding "type" record and it should always be masked using Algorithm CM\_SSN\_MASK:

```
<personSSN mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
```

- One schema captures a person ID and a corresponding ID Type and it should be masked with Algorithm CM\_SSN\_MASK if the type is "SSN" and masked with algorithm CM\_FEIN\_MASK if the type is "FEIN".

```
<personIdType mapXML=BO_DATA_AREA mdField=ID_TYPE_CD/>
<personId mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
```

- One schema captures a person ID and a corresponding ID Type and it has the same masking rules as the previous schema, but a different field name is used for the ID Type code. (This scenario could happen if for example a different label is desired for ID Type on the user interface for this schema.)

```
<personIdType mapXML=BO_DATA_AREA mdField=CM_ID_TYPE/>
<personId mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
```

For this scenario, the feature options may look like this:

1. **field="PER\_ID\_NBR", alg="CM\_SSN\_MASK"**
2. **field="PER\_ID\_NBR", alg="CM\_SSN\_MASK", where="ID\_TYPE\_CD='SSN'"**
3. **field="PER\_ID\_NBR", alg="CM\_FEIN\_MASK", where="ID\_TYPE\_CD='FEIN'"**
4. **field="PER\_ID\_NBR", alg="CM\_SSN\_MASK", where="CM\_ID\_TYPE='SSN'"**
5. **field="PER\_ID\_NBR", alg="CM\_FEIN\_MASK", where="CM\_ID\_TYPE='FEIN'"**

For each schema, the system will first find whether the element applies to any masking option. It will find 5 masking options for the field PER\_ID\_NBR. Then it will determine if any sibling elements match the 'where' clause.

- If more than one sibling element matches a 'where' clause, a runtime error is issued. For example if a schema has an element that references "mdField=ID\_TYPE\_CD" and an element that references "mdField=CM\_ID\_TYPE", this is an error. Additionally, if multiple elements reference mdField=ID\_TYPE\_CD", this is an error.
- If one and only one sibling element matches a 'where' clause, the value of the element is compared to the values defined in the 'where' clause. If it finds a match on the value, the appropriate masking algorithm is applied. If no match is found (for example, the Person ID Type is "LICENSE") the element is displayed as is.
- If no sibling element matches a 'where' clause and a feature option exists with no 'where' clause (option 1 above), then the masking algorithm of the option with no 'where' clause is applied.
- **Changing the value in the 'where' clause.** If your implementation has some users that are allowed to change records where some data is masked based on a condition, it is recommended to design the user interface to reset the masked value when the value in the 'where' clause changes. For example, if a user is prevented from viewing a person's social security number, but the user is allowed to make updates to the person's record, changing the value of the Person ID Type should reset the Person ID Number. This would ensure that the user does not 'unmask' the social security number by simply changing the ID Type.

## Records Maintained Using Page Maintenance

For data that is accessed via a page maintenance service call, indicate the table name and the field name where the data resides: **table="table\_name", field="fld\_name", alg="algorithm name"**

For example if the Person record and its collection of identifiers are displayed and maintained using page maintenance, the option value should be `table="CI_PER_ID", field="PER_ID_NBR", alg="algorithm name"`

A "where" clause may also be specified: `table="table_name", field="fld_name", where="fld_name='value'", alg="algorithm name"`

This is useful for data that resides in a child table where only data of a certain type needs to be masked. For the person ID example, `table="CI_PER_ID", field="PER_ID_NBR", alg="algorithm name", where="ID_TYPE_CD='SSN'"`

## Characteristic Data

For data that is stored as a characteristic, simply indicate the characteristic type: `CHAR_TYPE_CD='char type', alg="algorithm name"`

This needs to be defined only once regardless of which characteristic entity the char type may reside on. Note that only ad-hoc characteristics are supported.

## Masking Fields in Explorer Zones or Info Strings

In explorer zones data is often retrieved using SQL directly from the database. No masking is applied automatically in this case. If there is data in the explorer zone results that should be masked, the masking must be applied by calling a business service.

Similarly, an MO Info algorithm may not use BO interaction to get data. It may access data using SQL for efficiency purposes. No masking is applied when retrieving data via SQL. To apply masking to a string prior to including it in an info string, the masking must be applied by calling a business service.

The system supplies two business services to be called to determine if masking rules apply for a specific field.

- **F1-TableFieldMask** - Mask a Table field. This business service receives a table name, field name and one or more field values. If masking applies it returns the masked value.
- **F1-SchemaFieldMask** - Mask a Schema field. This business service receives a schema name and type, XPath and field value. If masking applies it returns the masked value.

## Search Service Results

For data that is displayed via a search service call, indicate the search name and the appropriate field to mask along with the masking algorithm. For example: `search="SearchServiceName", field="PER_ID_NBR", where="ID_TYPE_CD='SSN'", alg="algorithm name"`

To find the name of the search service, launch the search in question, right click in the filter area and choose View Source. Search for ServiceName. The service name is listed there. To find the field name to mask, go back to the search window and right click on the results area and choose View Source. Look for the Widget Info section and find the field name in the SEARCH RESULTS (do not include the \$). Note, the "where" statement can only apply to fields that are also part of the search results.

## Grant Access Rights To The User Groups

For each security type, identify which users can see its data unmasked and which users can only see its data masked.

If the masked and unmasked users fit into existing user groups, no additional user groups are necessary. Otherwise, create new user groups for the masked and unmasked users.

After the user groups for each security type are defined, link each user group to the application service defined above. When a user group is linked to the application service, you will define the authorization level for each security type linked to the application service. If a user group's users should see the security type's field values unmasked, set the authorization level to 2; otherwise set it to 1. Refer to [User Group - Application Services](#) for the transaction used to link a user group to an application service.

---

**NOTE:** Flush the cache. Remember that any time you change access rights you should flush the security cache (by entering flushAll.jsp on the URL of the application) if you want the change to take effect immediately.

---

## Additional Masking Information

The following points provide additional information to assist in your masking configuration:

- If the demonstration database includes a **Data Masking** feature configuration, review the settings because it will probably contain masking rules that will match your own.
- On data input pages, a user might be able to enter or change masked data, such as a bank account number, but not be able to subsequently see what they added or changed.
- External systems can request information by performing a service call via a web service. Please keep in mind that some web service requests require data to be masked and some do not. For example, a request from an external system to synchronize person information needs the person's social security number unmasked; whereas a request from a web self service application to retrieve the same person information for display purposes needs the person's social security number masked. To implement this type of requirement, different users must be associated with each of the requests and these users must belong to separate user groups with different access rights.
- If a maintenance object (MO) contains a field that holds an XML document and a service call invokes the MO's service program directly, the system will mask individual XML elements in the field if a **Determine BO** algorithm has been plugged into the [maintenance object](#) and the element(s) in the respective BO schema have been secured as described above.

## The Base Package Controls One User, One User Group, And Many Application Services

When the system is initially installed, the following information is delivered:

- Application services for all secured transactions, maintenance objects, business objects, business services, scripts and zones in the base package.
- A user identified by the user id **SYSUSER**.
- A user group identified by the user group code **ALL\_SERVICES**. This user group is associated with all supported application services delivered with the base product. This user group is given access to all access modes for all application services (i.e., all actions on all transactions).
- The user **SYSUSER** is linked to the **ALL\_SERVICES** user group. This means that this user has access to all transactions and all actions.

---

**NOTE:** An implementation may use the User Enable setting on the user record to disable SYSUSER so that it is not available as a valid user in the application.

---

When you receive an upgrade:

- New application services are delivered for the new transactions, business objects, zones introduced in the release.
- Existing application services are updated with changes in their access modes (e.g., if a new action is added to a transaction, its application service is updated accordingly).
- The **ALL\_SERVICES** user group is updated so it can access the new / changed application services.

---

**CAUTION:** Important! You cannot change or remove the information delivered for **ALL\_SERVICES**. This information is owned by the base package. It is provided so that an "initial user" has access to the entire system and can setup user groups and users as per your organization's business requirements. It is not recommended to provide your own users with access to the **ALL\_SERVICES** user group. Rather, create user groups that are appropriate for the organization's



business requirements and define user access to these user groups. If you introduce new transactions, configure them for the appropriate custom user groups.

---

## Importing Security Configuration from an External Source

The product provides support for importing security information from an external source:

- If your organization uses Lightweight Directory Access Protocol (LDAP), you can import your existing LDAP users and groups into the system. Once imported, all user and group functions are available. You can import a user group, or a single user. You can resynchronize your LDAP users and groups at any time.

---

**FASTPATH:** For more information refer to [LDAP Integration](#).

---

- The system provides an integration with Oracle Identity Manager. When a user is created in the identity manager product, its information can automatically be interfaced to the product. Once the user is successfully created in the system, all functions are available.

---

**FASTPATH:** For more information refer to [Oracle Identity Manager Integration](#).

---

## The Big Picture of Row Security

Some products allow you to limit a user's access to specific rows. For example, in Oracle Utilities Customer Care and Billing, row level security prevents users without appropriate rights from accessing specific accounts.

By granting a user access rights to an account, you are actually granting the user access rights to the account's bills, payment, adjustments, orders, etc.

The topics in this section describe basic row level security objects.

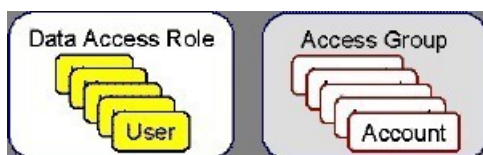
---

**FASTPATH:** Refer to your product's documentation for more information on row level security, if applicable.

---

## Access Groups, Data Access Roles and Users

We'll use an example from Oracle Utilities Customer Care and Billing to describe how access groups and roles are used to restrict access to accounts. The following diagram illustrates the objects involved with account security:



An **Access Group** defines a group of **Accounts** that have the same type of security restrictions. A **Data Access Role** defines a group of **Users** that have the same access rights (in respect of access to accounts). When you grant a data access role rights to an access group, you are giving all users in the data access role rights to all accounts in the access group.

The following points summarize the data relationships involved with account security:

- An account references a single **access group**. An **access group** may be linked to an unlimited number of **accounts**.
- A **data access role** has one or more **users** associated with it. A **user** may belong to many **data access roles**.
- A **data access role** may be linked to one or more **access group**. An **access group** may be linked to one or more **data access roles**.



If you use row level security, setting up your access roles and groups can be easy or challenging - it all depends on your organization's requirements. Refer to the product's **Administration Guide - Implementing Account Security** for several case studies. These case studies provide examples of how different requirements can be implemented using these concepts.

## Defining Application Services

An application service exists for every transaction in the system. Please refer to [Application Security](#) for a description of how application services are used when you grant user groups access rights transactions.

---

**CAUTION:** Important! If you introduce a new application service, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

### Application Service - Main

Select **Admin > Security > Application Service** to define an application service.

#### Description of Page

Enter a unique **Application Service** code and **Description** for the application service.

Indicate the application service's various **Access Modes** (i.e., actions). Refer to [Action Level Security](#) for more information about the significance of these fields.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [SC\\_APP\\_SERVICE](#).

### Application Service - Application Security

Use the Application Security portal to set up security for an application service.

Open this page using **Admin > Security > Application Service**, and then navigate to the **Application Security** tab.

This section describes the available zones on this page.

**Application Service Details zone.** The Application Service Details zone contains display-only information about the selected application service, including the Access Modes for the application service and its security type.

**User Groups with Access zone.** The User Groups with Access zone lists the user groups that have access to the application service. The following actions are available:

- Click the **Description** link to navigate to the User Group - Users page for the adjacent user group.
- Click **Deny Access** to remove the user group's access rights from the selected Application Service.
- Use the search filters to display the user groups that contain a specific user.

**User Groups without Access zone.** The User Group without Access zone lists the user groups that do not have access to the application service. The following actions are available:

- Click the **Description** link to navigate to the User Group - Users page for the user group.
- Click **Grant Access** to navigate to the User Group - Application Services page for the user group. The page is automatically positioned at the selected application service.
- Use the search filters to display the user groups that contain a specific user.

# Defining Security Types

Security types are used to define the types of [field level security](#).

---

**NOTE: Programming is required.** You cannot have field level security without introducing logic to user exits. Refer to [Field Level Security](#) for more information on how security types are used to define field level security.

---

## Security Type - Main

Select **Admin > Security > Security Type** to define your security types.

### Description of Page

Enter a unique **Security Type** and **Description**.

Use the **Authorization Level** grid to define the different authorization levels recognized for this security type. Enter an **Authorization Level Number** and its **Description**.

---

**NOTE: Programming is required.** Note that the values that you enter are not interpreted by the system itself, but by the user exit code used to implement the special security. Check with the developer of the user exit logic for the correct values. Refer to [Field Level Security](#) for more information on how security types are used to define field level security.

---

Use the **Application Services** grid to define the application service(s) to which this security type is applicable. If this application service is already associated with user groups, you must update each user group to define their respective security level. This is performed using [User Group - Application Service](#).

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_SC\\_TYPE](#).

## Defining User Groups

A user group is a group of users who have the same degree of security access. Think of a user group as a "role"; associated with a role are:

- The users who play this role
- The application services to which the role's users have access (along with the actions they can execute for each service and their field level security authorization levels).

## User Group - Main

Select **Admin > Security > User Group** to view the application services to which a user has access.

---

**CAUTION:** Application services may not be changed or removed from the **ALL\_SERVICES** user group. Refer to [The Base Package Controls One User, One User Group, And Many Application Services](#) for an explanation.

---

### Description of Page

Enter a unique **User Group** code and **Description** for the user group.

**Owner** indicates if this user group is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a user group. This information is display-only.

The **Application Services** grid displays the various application services to which users in this group have access. Please note the following in respect of this grid:

- Use the **Application Service** search to restrict the application services displayed in the grid. For example, if you only want to see application services that start with the word "field", you can enter this word and press enter.
- To add additional application services to this user group, navigate to the [User Group - Application Services](#) page and click the add icon.
- To remove or change this user group's access to an application service, click the go to button adjacent to the respective application service. This will cause you to be transferred to the [User Group - Application Services](#) tab where you should click the - icon to remove the application service from the user group.
- Note, **Owner** indicates if this user group / application service relationship is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an application service to the user group. This information is display-only.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [SC\\_USER\\_GROUP](#).

## User Group - Application Services

Select **Admin > Security > User Group** and navigate to the **Application Services** tab to maintain a user group's access rights to an application service.

---

**NOTE: Important!** When you grant a user group access rights to an application service, you are actually granting all users in the user group access rights to the application service.

---

### Description of Page

The **Application Service** scroll contains the application services to which the **User Group** has access.

---

**NOTE:** You can also use Main page to select the application service for which you wish to change the access privileges. To do this, simply click the go to button adjacent to the respective application service.

---

To add additional application services to this user group, click the + icon and specify the following:

- Enter the **Application Service ID** to which the group has access.
- Define the **Expiration Date** when the group's access to the application service expires.

Define the **Access Modes** that users in this group have to the **Application Service**. When a new application service is added, the system will default all potential **Access Modes** associate with the **Application Service**. You need only remove those modes that are not relevant for the **User Group**. Refer to [Action Level Security](#) for more information about access modes.

---

**CAUTION:** Important! If an application service supports actions that modify the database other than **Add, Change, and Delete**; you must provide the user with **Change** access in addition to the other access rights. Consider a transaction that supports actions in addition to **Add, Change, and Inquire** (e.g., **Freeze, Complete, Cancel**). If you want to give a user access to any of these additional actions, you must also give the user access to the **Inquire and Change** actions.

---

If you require additional security options, often referred to as "field level" security, then you use **Security Type Code** and assign an **Authorization Level** to each. When a new application service is added, the system will display a message indicating how many security types are associated with this application service. Use the search to define each Security Type Code and indicate the appropriate Authorization Level for this user group. Refer to [Field Level Security](#) for more information about security types.

## User Group - Users

Select **Admin > Security > User Group** and navigate to the **Users** tab to maintain the users in a user group.

### Description of Page

The scroll area contains the users who are part of this user group.

---

**NOTE:** Keep in mind that when you add a **User** to a **User Group**, you are granting this user access to all of the application services defined on the **Application Services** tab.

---

The following fields are included for each user:

- Enter the **User ID** of the user.
- Use **Expiration Date** to define when the user's membership in the group expires.
- **Owner** will be **Customer Modification**.

---

**NOTE:** You can also add a user to a user group using [User - Main](#).

---

## Defining Access Groups

---

**FASTPATH:** Refer to [The Big Picture of Row Security](#) for a description of how access groups are use to restrict access to specific objects.

---

Access groups control which groups of users (referred to as Data Access Roles) have rights to accounts (or other objects) associated with the access group. Select **Admin > Security > Access Group** to define your access groups.

### Description of Page

Enter a unique **Access Group** code and **Description** for the data access group.

Use the **Data Access Role** collection to define the data access roles whose users have access to the access group's accounts (or other objects). Keep in mind that when you add a **Data Access Role** to an **Access Group**, you are granting all users who belong to this role access to all of the accounts (or other objects) linked to the access groups. Refer to [Access Groups, Data Access Roles and Users](#) for more information.

---

**NOTE:** You can also use [Data Access Role - Access Group](#) to maintain a data access role's access groups.

---

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_ACC\\_GRP](#).

## Defining Data Access Roles

---

**FASTPATH:** Refer to [The Big Picture of Row Security](#) for a description of how access groups are use to restrict access to specific objects.

---

The data access role transaction is used to define two things:

- The users who belong to the data access role.
- The access groups whose accounts (or other objects) may be accessed by these users.

## Data Access Role - Main

Select **Admin > Security > Data Access Role** to define the users who belong to a data access role.

### Description of Page

Enter a unique **Data Access Role** code and **Description** for the data access role.

The scroll area contains the **Users** who belong to this role. A user's data access roles play a part in determining the accounts (or other objects) whose data they can access. Refer to [Access Groups, Data Access Roles and Users](#) for more information.

To add additional users to this data access role, press the add button, **+**, and specify the following:

- Enter the **User ID**. Keep in mind that when you add a **User** to a **Data Access Role**, you are granting this user access to all of the accounts (or other objects) linked to the data access role's access groups.
- Use **Expiration Date** to define when the user's membership in this data access role expires.

---

**NOTE:** Also maintained on the user page. You can also use [User - Access Security](#) to maintain a user's membership in data access roles.

---

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL\\_DAR](#).

## Data Access Role - Access Group

Select **Admin > Security > Data Access Role** and navigate to the **Access Groups** tab to define the access groups whose accounts (or other objects) may be accessed by the users in this data access role.

### Description of Page

Use the **Access Group** collection to define the access groups whose objects can be accessed by this role's users. Keep in mind that when you add an **Access Group** to a **Data Access Role**, you are granting all users who belong to this role access to all of the accounts (or other objects) linked to the access groups. Refer to [Access Groups, Data Access Roles and Users](#) for more information.

---

**NOTE:** You can also use [Access Group - Main](#) to maintain an access group's data access roles.

---

## Defining Users

The user maintenance transaction is used to define a user's user groups, data access roles, portal preferences, default values, and To Do roles. To access the user maintenance transaction, select **Admin > Security > User**.

The user maintenance transaction is the same transaction invoked when the user launches [Preferences](#).

## User Interface Tools

---

This section describes tools that impact many aspects of the user interface.

# Defining Menu Options

The contents of this section describe how you can add and change menus.

---

**CAUTION:** Updating menus requires technical knowledge of the system. This is an implementation and delivery issue and should not be attempted if you do not have previous experience with menus.

---

**NOTE: Security and menus.** Refer to [Application Security](#) for a discussion of how application security can prevent menu items (or an entire menu) from appearing.

---

**NOTE: Module configuration and menus.** Your [module configuration](#) can prevent menu items (or an entire menu) from appearing.

---

## Menu - Main

This transaction is used to define / change any menu in the system. Navigate to this page using **Admin > System > Menu**.

### Description of Page

Enter a meaningful, unique **Menu Name**.

**Owner** indicates if this menu line is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a menu line. This information is display-only.

The **Flush Menu** button is used to flush the cached menu items so you can see any modified or newly created menus. Refer to [Caching Overview](#) for more information.

**Menu Type** defines how the menu is used. You have the following options:

- **Admin** is one of the menus that appears in the Application Toolbar. It is a special type of menu because [admin menu](#) items can be grouped alphabetically or by functional group. Refer to the description of Admin Menu Order on [Installation Options - Framework](#) for more information about admin menu options.
- **Context** refers to a [context menu](#).
- **Main** is another menu that appears in the Application Toolbar that is simply titled [Menu](#).
- **Page Action Menu** defines buttons that appear in the [Page Title Area](#).
- **Submenu** defines a menu group that appears when an Application Toolbar menu is selected. for the Admin menu, this is only visible when it's organized functionally.
- Enter **User Menu** refers to the menu items that appear on the [user menu](#); for example, User Preferences.

**Description** provides a description of the menu. Note that this is not the text used when displaying a menu option.

**Sequence** is only enabled for the **Main** and **Admin** menu types.

The grid contains a summary of the menu's lines. Besides the standard add and delete icons available in a grid, the following information is displayed:

- **Menu Line ID** is the unique identifier of the line on the menu. This information is display-only. Before the menu line id is a Go To icon that allows a user to drill into the Menu Items for the displayed menu line.
- **Sequence** is the relative position of the line on the menu. Note, if two lines have the same **Sequence**, the system organizes the lines alphabetically (based on the **Long Label**, which is defined on the next tab).

---

**NOTE:** An implementation may override the sequence of a base product owned menu line. Also note that the sequence is defined on the menu line language table, allowing for different orders to be used for different languages (or to let the menu be sorted alphabetically in one language and in a specified order in a different one).

---

- **Navigation Option / Submenu** contains information about the line's items. If the line's item invokes a submenu, the submenu's unique identifier is displayed. If the line's item(s) invoke a transaction, the description of the first item's [navigation option](#) is displayed.
- **Long Label** is the verbiage that appears on the menu line.
- **Item Count** is the number of menu items on the line.
- **Owner** indicates if this menu line is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a menu line. This information is display-only.

---

**NOTE: Adding menu lines to base owned menus.** An implementation may choose to add custom menu lines along with its menu item (or items) to a base owned menu.

---

Refer to the description of [Menu Items](#) for how to add items to a menu line.

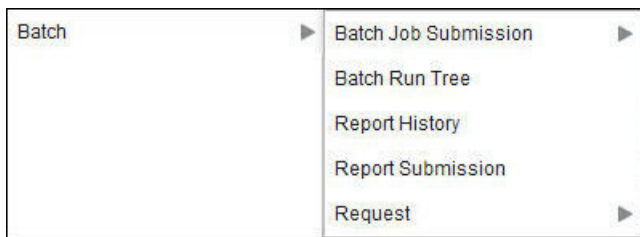
## Menu - Menu Items

Once a menu has lines (these are maintained on the main page), you use this page to maintain a menu line's items.

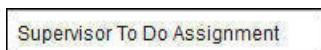
Each menu line can contain one or two menu items. The line's items control what happens when a user selects an option on the menu.

There are two types of menu lines that define a single menu item: one type causes a submenu to appear; the other type causes a transaction or script to be invoked when it's selected.

- The following is an example of a menu line with a single item that opens a submenu:



- The following is an example of a menu line with a single menu item that launches a transaction or script:



A menu line that defines two menu items is used to provide an Add option and a Search option for the same type of object. In this case each menu item defines a transaction or script to be launched. The menu is rendered with the Add and Search options displayed. The following is an example of a menu line with two menu items.



Navigate to this tab by clicking the Go To button adjacent to a menu line from the Main tab.

### Description of Page

**Menu Name** is the name of the menu on which the line appears. **Menu Line ID** is the unique identifier of the line on the menu. **Owner** indicates if this menu is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

The **Menu Line Items** scroll contains the line's menu items. The following points describe how to maintain a line's items:

- **Menu Item ID** is the system assigned unique identifier of the item.
- **Owner** indicates if this item is owned by the base product or by your implementation (**Customer Modification**).
- If the menu item should invoke a submenu:
  - Use **Sub-menu Name** to identify the menu that should appear when the line is selected
  - Use **Long Label** to define the verbiage that should appear on the menu line
  - Populate the **Override Label** to override the long label of a base product owned sub-menu.
- If the item should invoke a transaction or BPA script:
  - Use **Sequence** to define the order the item should appear in the menu line (we recommend this be set to **1** or **2** as a menu line can have a maximum of 2 menu items). The “search” menu item should be defined as sequence 1 and the “add” menu item as sequence 2 given that the label of the “search” menu item is used for the menu line’s label.
  - Use **Navigation Option** to define the transaction or script to open. Refer to [Defining Navigation Options](#) for more information.
  - For a menu line that includes two items — one for Add and one for Search, if one of the items includes configuration for **Image GIF Location and Name** , the system assumes that this represents the Add. This functionality is carry over from earlier releases where the Add function rendered in the menu with a “+” image, which also identified the item that represents the Add. If neither item includes Image configuration (because it is no longer needed for rendering the menu), the system relies on the order of the items as mentioned above. The first item is the “search” and the second item is the “add”.
  - **Image Height, Image Width and Balloon Description** are not applicable at this time.
- Use the **Long Label** to define the text to appear on the menu entry. Note that when a menu line defines two menu items, the long label on the search entry is used to build the menu entry text. The label long on the menu line that defines the Add option is information only.
- The **Override Label** is provided in case you want to override the base-package's label.
- Use **Application Service** and **Access Mode** to easily suppress a menu item for one or more users. Refer to [Application Security](#) for more information.

## The Big Picture of System Messages

All error, warning and informational messages that are displayed in the system are maintained on the message table. Every message is identified by a combination of two fields:

- **Message category number.** Think of a message category as a library of messages related to a given functional area. For example, there is a message category for billing messages and another one for payment messages.
- **Message number.** A unique number identifies each message within a category.

Every message has two components: a brief text message and a long description. On the **Main** tab, you can only maintain the brief message. If you need to update a message's long description, you must display the message on the **Details** tab.

---

**NOTE: You cannot change the product’s text.** If the message is "owned" by the product, you cannot change the product’s message or detailed description. If you want your users to see a different message or detailed description other than that supplied by the product, display the message on the **Details** tab and enter your desired verbiage in the "customer specific" fields (and flush the [cache](#)).

---



# Defining System Messages

The contents of this section describe how to maintain messages that appear throughout the system. An implementation may introduce messages used in custom processes or may choose to override the text for messages delivered by the product.

## Message - Main

Select **Admin > System > Message** to maintain a message category and its messages.

### Description of Page

To add a new message category, enter a **Message Category** number and **Description**.

---

**CAUTION:** Message category 80000 or greater must be used to define new messages introduced for a specific implementation of the system. Changes to other Message Text will be overwritten when you next upgrade. If you want to make a change to a Message, drill down on the message and specify Customer Specific Message Text. Note that even for message categories 80000 and higher, message numbers lower than 1000 are reserved for common base product messages.

---

**NOTE: Owner** indicates if this message category is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a category. This information is display-only.

---

To update a message, you must first display its **Message Category**. You can optionally start the message grid at a **Starting Message Number**.

To override the message text or detailed description of messages owned by the base product, click on the message's go to button. When clicked, the system takes you to the **Details** tab on which you can enter your implementation's override text.

The following points describe how to maintain messages owned by your implementation:

- Click the - button to delete a message.
- Click the + button to add a new message. After clicking this button, enter the following fields:
- Use **Message Number** to define the unique identifier of the message within the category.
- Use **Message Text** to define a basic message. You can use the %n notation within the message text to cause field values to be substituted into a message. For example, the message text **The %1 non-cash deposit for %2 expires on %3** will have the values of 3 fields merged into it before it is displayed to the user (%1 is the type of non-cash deposit, %2 is the name of the customer, and %3 is the expiration date of the non-cash deposit).

---

**NOTE:** The system merges whatever values are supplied to it. Therefore, if a programmer supplies a premise address as the second merge parameter in the above message, this address is merged into the message (rather than the customer's name).

---

- **Owner** indicates if this message number is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a message. This information is display-only.
- Click the go to button to enter a detailed description of the message. Clicking this button transfers you to the **Details** tab.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_MSG](#). In addition, messages are used throughout the system for error messages and other system messages.

## Message - Details

Select **Admin > System > Message** and navigate to the **Details** tab to define detailed information about a message.

---

**NOTE: Drilling in from the Main tab.** Rather than scrolling through the messages, you can display a message by clicking the respective go to button in the grid on the main tab.

---

### Description of Page

The **Message Collection** scroll contains an entry for every message in the grid on the Main tab. It's helpful to categorize messages into two categories when describing the fields on this page:

- Product messages
- Implementation-specific messages (i.e., a message added to **Message Category** 80000 or greater)

For product messages, you can use this page as follows:

- If you want to override a message, specify **Customer Specific Message Text**.
- You are limited to the same substitution values used in the original **Message Text**. For example, if the original **Message Text** is **The %1 non-cash deposit for %2 expires on %3** and %1 is the type of non-cash deposit, %2 is the name of the customer, and %3 is the expiration date of the non-cash deposit; your **Customer Specific Message Text** is limited to the same three substitution variables. However, you don't have to use any substitution variable in your message and you can use the substitution variables in whatever order you please (e.g., %3 can be referenced before %1, and %2 can be left out altogether).
- If you want to override the detailed description of an error message, specify **Customer Specific Description**. Note that the system does not present detailed descriptions when warnings are shown to users. Therefore, it doesn't make sense to enter this information for a warning message.

For implementation-specific messages, you can use this page as follows:

- **Message Text** is the same Message Text displayed on the main tab.

---

**CAUTION:** If both **Message Text** and **Customer Specific Message Text** are specified, the system will only display the **Customer Specific Message Text** in the dialog presented to the user.

---

- Use **Detailed Description** to define additional information about an error message. Note that the system does not present detailed descriptions when warnings are shown to users. Therefore, it doesn't make sense to enter this information for a warning message.

---

**CAUTION:** If both **Detailed Description** and **Customer Specific Description** are specified, the system will only display the **Customer Specific Description** in the dialog presented to the user.

---

## The Big Picture of Portals and Zones

A portal is a page that is comprised of one or more information zones. The base product pages are built using either a fixed page metaphor or using portals and zones. The contents of this section describe general information about portals and zones.

## There Are Three Types of Portals

There are three broad classes of portals:

- **Standalone Portal.** Standalone portals are separate pages where the main tab of the page is built using a portal. These pages are opened using any of the standard methods (e.g., by selecting a menu item, by selecting a favorite link, etc.). Additional tabs for a stand-alone portal may be included using tab page portals.
- **Tab Page Portals.** These types of portals cannot be attached to a menu. They simply define the zones for a tab on either a standalone portal or on a “fixed” page. Please contact customer support if you need to add portals to existing transactions.
- **Dashboard Portal.** The dashboard portal is a portal that appears in the [Dashboard Area](#) on the user’s desktop. Its zones contain tools and information that exist on the user's desktop regardless of the transaction.  
There is only one dashboard portal. This portal and several zones are delivered as part of the base-package. Your implementation can add additional zones to this portal. Please contact customer support if you need to add zones to the dashboard portal.

## Common Characteristics of All Portals

The topics that follow describe characteristics common to all types of portals.

### Portals Are Made Up of Zones

A portal is a page that contains one or more zones, and each zone contains data of some sort.

All zones reference a **Zone Type**. The zone type controls the behavior of the zone and the parameters available to configure the zone.

### Configuring Zones for a Portal

The portal includes configuration of how the zones should appear on the portal by default. This includes the following options, all of which may be overridden by an implementation.

- The order in which the zone should appear. An implementation may configure an override sequence to change the order zones on a base delivered portal.
- Whether the zone is visible on the portal. Zones delivered in the base product should be configured to be visible. But an implementation may override this if desired.
- Whether the zone should display initially collapsed or not. A zone’s data is only retrieved when it is expanded. As such, a zone may be configured to be initially collapsed when the data is not needed very often. A user can expand the zone when the information is required. Implementations may change the collapsed setting of a base product portal / zone.

---

**FASTPATH:** Refer to [Zones May Appear Initially Collapsed When a Page Opens](#) for more information.

---



---

**FASTPATH:** Refer to [Defining Portals](#) for more information about this configuration.

---

In addition, the portal includes configuration to indicate whether or not the portal should appear on a user’s portal preferences. This is typically enabled for a portal that provides disparate information where not all zones are applicable to all users or where users may wish to adjust the order of the zones. An example of a portal enabled for portal preferences is the Dashboard portal. The user can override zone oriented configuration for the portal:

- Which zones appear on that portal
- The order in which the zones appear
- Whether the zones should be initially collapsed when the portal opens.
- The refresh seconds. This is applicable to zones displaying data that changes often.

An implementation can optionally configure the system to define portal preferences on one or more "template" users. When a template user is linked to a "real" user, the real user's preferences are inherited from the "template" user and the "real" user cannot change their preferences. Some implementations opt to work this way to enforce a standard look and feel for users in the same business area.

---

**FASTPATH:** Refer to [User — Portal Preferences](#) for more information about how users configure their zones.

---

## Granting Access to Zones

An [application service](#) is associated with each zone. A user must be granted access rights to the respective application service in order to see a zone on a portal.

---

**FASTPATH:** Refer to [The Big Picture Of Application Security](#) for information about granting users access rights to an application service.

---

Please note the following with respect to zone application security:

- For most base product portals, all the zones for all the portals reference the same application service that is used to grant access to the main (stand-alone) portal for the page. In other words, if the user has access to the page, then he has access to all the zones on all portals for the page. There may be exceptions to this rule for certain portals.
- For a base product multi-query zones, typically the individual query zones and the multi-query zone reference the same application service that is used to grant access to the main (stand-alone) portal for the page. However, there may be individual query zones provided with a unique application service. This may occur when the query option is unusual and not applicable to all users or even to all implementations. If a user does not have security access to an individual query zone, that option will not be available in the dropdown.
- For base product portals that are configured to show on portal preferences, it is common that the portal contains different types of zones that may be applicable to different types of users. Typically these types of portals will deliver a unique application service for each zone so that an implementation may configure which user groups are allowed to view each zone. For these types of portals, please note the following:
  - A user's [Portal Preferences](#) page contains a row for a zone regardless of whether the user has access rights to the zone. Because of this, the system displays an indication of the user's access rights to each zone.
  - If a user's access rights to a zone are revoked, the zone will be suppressed when the user navigates to the respective portal.
  - Revoking a user's access rights does not change the user's [portal preferences](#) (i.e., a user can indicate they want to see a zone even if they don't have access to the zone - such a zone just won't appear when the respective portal appears).

---

**NOTE: If you don't need to use zone security.** When defining a zone, an application service is required. For zones that don't require special security, the product provides a "default" application service (F1-DFLT) that may be used. The expectation is that all user groups are granted access to this application service.

---

## Common Characteristics of Stand-Alone Portals

The topics that follow describe additional characteristics specific to [stand-alone portals](#).

### Putting Portals on Menus

A stand-alone portal should appear as a menu item on one of your menus. The following points provide how to do this:

- Every stand-alone portal has an associated navigation option. You can see a portal's navigation option on the [Portal - Main](#) page.

- To add a portal to a menu, you must add a [menu item](#) to the desired menu. This menu item must reference the portal's navigation option. There are two ways to add a menu item:
- If the portal's navigation option doesn't currently exist on a menu, you can press the **Add To Menu** button on the [Portal - Main](#) page. When you press this button, you will be prompted for the menu. The system will then create a menu item on this menu that references the portal's navigation option.
- You can always use the [Menu](#) page to add, change and delete menu items.

---

**NOTE: No limit.** A portal's navigation option can appear on any number of menu items (i.e., you can create several menu items that reference the same portal).

---

**NOTE: Favorite links.** Your users can set up their preferences to include the portal's navigation option on their [Favorite Links](#). This way, they can easily navigate to the portal without going through menus.

---

## Granting Access to A Portal

An [application service](#) is associated with each [stand-alone portal](#). A user must be granted access rights to the respective application service in order to see a portal. Tab portals do not have separate security access. If a user has access to the main stand-alone portal, then the user will have security access to all its tabs. However, as mentioned in [Granting Access to Zones](#), in some scenarios, the individual zones on the portal may have different security access rights depending on the functionality.

---

**FASTPATH:** Refer to [The Big Picture Of Application Security](#) for information about granting users access rights to an application service.

---

**NOTE: Automatically created.** When you add a new stand-alone portal, the system automatically creates an application service behind the scenes. You'll need to know the name of this application service as this is what you use to grant access to the portal. The name of each stand-alone portal's application service is shown on the portal transaction.

---

Please note the following in respect of how application security impacts a user's portals:

- A user's [Portal Preferences](#) page only shows the portals configured to show on user preferences and where they have security access.
- The system's menus only show portals to which a user has security access.
- Users can set up favorite links to all portals, but they must have security rights to the portal's application service in order to invoke the favorite link.

## Custom Look and Feel Options

The default look and feel of the application can be customized via feature configuration and cascading style sheets. The base product is provided with a **Custom Look And Feel** [Feature Configuration](#) type. You may want to set up a feature configuration of this type to define style sheet and UI Map help options.

## User Interface

The base product allows for the conditional inclusion of customer styles into the system style set. The custom style may override any style provided by the base product. The style sheet may also include new styles for use in customer zone definitions. Use the **Style Sheet** option on the **Custom Look And Feel** Feature Configuration to define your custom style sheet.

---

**NOTE:** Some styles cannot change if they are part of the HTML code.

---

**CAUTION:** Implementers must ensure that the customized user interface is stable and scalable. Changing font, alignment padding, border size, and other user interface parameters may cause presentation problems, like scrollbars appearing or disappearing, cursors not working as expected, and unanticipated look and feel alterations of some layouts.

---

## UI Map Help

A [tool tip](#) can be used to display additional help information to the user. This applies to section elements as well as individual elements on a map zone or UI Map. Refer to the tips context sensitive zone associated with the UI Map page for more information. The **Custom Look And Feel** Feature Configuration provides options to control the following:

- Whether UI Map Help functionality is turned on or off. By default it is turned on.
- Override the default help image with a custom image
- The location of the help image, either before or after the element.

---

**FASTPATH:** Refer to the feature configuration for a detailed description of each option.

---

## Setting Up Portals and Zones

The topics in this section describe how to set up portals and zones. Please refer to [The Big Picture of Portals and Zones](#) for background information.

## Defining Zone Types

A Zone Type represents a particular type of zone with a specific behavior. For example, a data explorer zone type is used to select data using a specific SQL statement and display the data based on parameter configuration. The zone type defines the Java Class that controls the behavior of the zone and defines the parameters that the Java Class supports. The base product supports many zone types used to build the portal / zone user interface. Implementations may introduce their own zone types.

---

**NOTE:** It is not very common for an implementation to introduce their own zone types.

---

Select **Admin > System > Zone Type** to maintain zone types.

### Description of Page

Specify an easily recognizable **Zone Type** code and **Description**. Use the **Detailed Description** to describe in detail what the zone type does.

---

**CAUTION:** When adding new zone types, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

**Owner** indicates if this zone type is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a zone type. This information is display-only.

**Java Class Name** is the Java class responsible for building the zone using the parameters defined below.

Two types of parameters are specified when defining a zone type:

- Parameter values that have a **Usage** of **Zone** are defined on the zones and control the functionality of each zone governed by the zone type. A **Usage** value of **Zone - Override Allowed** indicates that an implementation may override the parameter value for a base zone.
- Parameter values that have a **Usage** of **Zone Type** are defined directly on the zone type and control how the zone type operates (e.g., the name of the XSL template, the name of the application service). A **Usage** value of **Zone Type - Override Allowed** indicates that an implementation may override the parameter value for a base zone type.

The following points describe the fields that are defined for each parameter:

- **Sequence** defines the relative position of the parameter.
- **Parameter Name** is the name of the parameter.
- **Description** is a short description that allows you to easily identify the purpose of the parameter.
- **Comments** contain information that you must know about the parameter or its implementation. For parameters with a usage of **Zone** or **Zone — Override Allowed**, this information is visible to the user when viewing or defining this parameter for a zone of this type.
- **Usage** indicates whether the parameter value is defined in a **Zone** of this type or in the **Zone Type**. **Zone - Override Allowed** and **Zone Type - Override Allowed** indicate that override values for the parameters defined in a base zone or base zone type can be entered.
- **Required** is checked to indicate that a zone must define a value for the parameter. It is not checked if a value for the parameter is optional. This field is protected if the **Usage** is **Zone Type** or **Zone Type - Override Allowed**.
- **Parameter Value** is used to define the value of zone type parameters. This field is protected if the **Usage** is **Zone** or **Zone - Override Allowed**.
- **Owner** indicates if this parameter is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a parameter. This information is display-only.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_ZONE\\_HDL](#).

### Zone Type Parameter Comments

For the product owned zone type parameters, the parameter's detailed description provides the detail needed for properly configuring the parameter. For the Action parameters (IMPLEMENTOR\_ACTION\_n), the parameter description is abbreviated. Additional detail about configuring this parameter may be found in the [Zone Action Parameter](#) detailed information. The same details apply.

## Defining Zones

The contents of this section describe how to maintain zones.

### Zone - Main

Implementations may use the zone page to define custom zones. In addition, an implementation may override descriptions or some parameter values for base product zones.

Select **Admin > System > Zone** to create or maintain a zone.

#### Description of Page

Specify an easily recognizable **Zone** identifier and **Description**. Note that if this zone appears on a portal, this description acts as the zone title.

**Override Description** is provided if your implementation wishes to override the description of the value provided by the product.

---

**CAUTION:** Important! When introducing a new zone, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

**Owner** indicates if this zone is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a zone. This information is display-only.

**Zone Type** identifies the zone type that defines how the zone functions.

**Application Service** is the application service that is used to provide security for the zone. Refer to [Granting Access To Zones](#) for more information.

The **Width** defines if the zone occupies the **Full** width of the portal or only **Half**.

---

**NOTE:** Zones on the [dashboard portal](#) are always the width of the dashboard.

---

If the zone type supports help text, you can use **Zone Help Text** to describe the zone to the end-users. Note that for multi-query zones, if the multi-query zone has help text, that is displayed for any zone selected. If the multi-query zone does not have help text, but the selected zone has help text, the selected zone's help text is displayed. Please refer to the section on [zone help text](#) for more information on how you can use HTML and cascading style sheets to format the help text.

Use **Override Zone Help Text** to override the product provided help text for this zone.

---

**NOTE: Viewing Your Text.** You can press the **Test** button to see how the help text will look when it's displayed in the zone.

---

The grid contains the zone's parameter values. The Zone Type controls the list of parameters. The grid contains the following fields:

- **Description** describes the parameter. This is display-only. Note that if there is a detailed description on the zone type parameter, a question mark icon appears next to the parameter's description. Click the icon to see details related to the parameter, including tips on how to populate the parameter value.

---

**NOTE: Additional Details for Some Parameters.** There are several parameter types that have a lot of detail related to the possible configuration that cannot easily fit into the detailed description. Refer to [Common Zone Parameters for Additional Information](#) about these parameters.

---

- **Parameter Value** is the value for the parameter.
- Use **Override Parameter Value** to override the existing value for this parameter. This field is enabled when the related zone type parameter value is **Zone - Override Allowed**, and the zone is owned by the base product.
- **Owner** indicates if this parameter is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_ZONE](#).

## Zone - Portal

Select **Admin > System > Zone** and navigate to the **Portal** tab to define the portals on which a zone appears.

#### Description of Page

The scroll area contains the portals on which the zone appears.

To add a zone to a portal, press the + button and specify the **Portal**.



---

**NOTE: Owner** indicates if this portal / zone relationship is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

---

**NOTE:** You can also add a zone to a portal using [Portal - Main](#). Additional configuration about how the zone appears on the portal is available only on the Portal.

---

## Additional Zone Topics

The topics in this section provide additional information related to setting up your zones.

### Zone Help Text

Most zone types support a button that allows a user to see zone-specific help text, which is defined on the zone page.

You can use HTML tags in the zone help text. The following is an example of help text that contains a variety of HTML tags:

**This zone summarizes `<font color=blue><b>revenue</b></font>` in 4 periods:`<br>`**

The above would cause the word **revenue** to be bold and blue:

- `<b>` and `</b>` are the HTML tags used to indicate that the surrounded text should be bold
- `<font color=blue>` and `</font>` are the HTML tags used to indicate that the surrounded text should be blue.

The following are other useful HTML tags:

- `<br>` causes a line break in a text string. If you use `<br><br>` a blank line will appear.
- `<i>` causes the surrounded text to be italicized

Please refer to an HTML reference manual or website for more examples.

You can also use "spans" to customize the look of the contents of a text string. For example, your text string could be `<span style="font-family:Courier; font-size:large; font-weight:bold;">revenue</span>`. This would make the word "revenue" appear as large, bold, Courier text. Please refer to a Cascading Style Sheets (CSS) reference manual or website for more examples.

The following is an example of help text using a variety of HTML tags:

```
<font FACE="arial" size=2>
```

```
This zone summarizes <font color=blue><b>revenue</b></font> in 4 periods:<br>
```

```
- The <b>1st period</b> is under your control. You simply select the desired <b>Period</b>, above <i>(you may need to click the down arrow to expose the filter section)</i><br>
```

```
- The <b>2nd period</b> is the period before the 1st period<br>
```

```
- The <b>3rd period</b> is the same as the 1st period, but in the previous year<br>
```

```
- The <b>4th period</b> is the period before the 3rd period<br>
```

```
<br>
```

```
The traffic light's color is determined as follows:<br>
```

```
- The ratio of the 1st and 3rd period is calculated<br>
```

```
- If this value is between 80 and 100, <font color=orange><b>yellow</b></font> is shown<br>
```

```
- If this value is < 80, <font color=red><b>red</b></font> is shown<br>
```

```
- If this value is > 100, <font color=green><b>green</b></font> is shown<br>
```

- If the value of the 3rd period is 0, no color is shown<br>

</font>

---

**NOTE:** It is possible to associate tool tip help with individual HTML and UI map elements. For more information, see [UI Map Help](#).

---

## Common Zone Parameters

For most zone parameters, the inline help for the parameter provides the detailed information needed for configuring the parameter values. For some parameters with very detailed descriptions, the inline help is abbreviated and more detail is provided here.

## Column Parameters

Data explorer zones are used to select data to display using one or more SQL statements. For each SQL statement, the zone may configure up to 20 Columns that contain the formatting definition for displaying the output data. The parameters follow the description pattern of **Column x for SQL y**.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1-DE**)
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**)
- Info Data Explorer - Single SQL (**F1-DE-SINGLE**)

The following sections describe the various types of mnemonics.

## Source Mnemonics

This table describe the mnemonics that control how the data in a column is derived.

Mnemonic	Description	Valid Values	Comments
<b>source=</b>	Defines how the column's value is derived.	<b>SQLCOL</b>	Indicates that the source of the column's value comes from a column in the SQL statement. This type of column must also reference the <b>sqlcol=</b> mnemonic.
		<b>BO</b>	Indicates that the source of the column's value comes from a <a href="#">business object</a> . This type of column must also reference the <b>bo=</b> , <b>input=</b> and <b>output=</b> mnemonics to define how to interact with the business object.
		<b>BS</b>	Indicates that the source of the column's value comes from a <a href="#">business service</a> . This type of column must also reference the <b>bs=</b> , <b>input=</b> and <b>output=</b> mnemonics to define how to interact with the business service.
		<b>SS</b>	Indicates that the source of the column's value comes from a <a href="#">service script</a> . This type of column must also reference the <b>ss=</b> , <b>input=</b> and <b>output=</b> mnemonics to define how to interact with the service script.
		<b>FORMULA</b>	Indicates that the source of this column's value is calculated using a formula. This type of column must also reference the <b>formula=</b> mnemonic.

Mnemonic	Description	Valid Values	Comments
		SETFUNC	Indicates that the source of this column's value is calculated using a superset of values from the rows in the SQL statement. This type of column must also reference the <b>setfunc=</b> mnemonic.
		ICON	Indicates that the source of this column's value is a <a href="#">display icon</a> reference (meaning that an icon will be displayed in the column). This type of column must also reference the <b>icon=</b> mnemonic to define the icon reference.  <b>NOTE:</b> When using this source mnemonic, the <a href="#">formatting mnemonic</a> is not applicable.
		FKREF	Indicates that the source of this column's value is an <a href="#">FK reference</a> (meaning that the FK reference's context menu and information string will be displayed in the column). This type of column must also reference the <b>fkref=</b> and <b>input=</b> mnemonics to define how the FK reference is called.  <b>NOTE:</b> When using this source mnemonic, the <a href="#">formatting mnemonic</a> is not applicable.
		SPECIFIED	Indicates that the source of this column's value is specified by concatenating literals and other column values. This type of column must also reference the <b>spec=</b> mnemonic.
		MSG	Indicates that the source of this column is a message from the <a href="#">message</a> table (along with any substitution variables). This type of column must also reference the <b>msg=</b> mnemonic.
<b>sqlcol=</b>	Defines the column in the SQL statement when <b>source=SQLCOL</b> .	COLUMN_NAME	Enter the name of a column that is retrieved in the SELECT statement. Note that if the select statement uses an alias for a column, then the alias should be referenced here.
		x	Where x is an integer value that references a column by its relative position in the SELECT statement. For example, <b>sqlcol=3</b> would display the 3rd column in the SELECT statement).
<b>bo=</b>	Defines the business object to invoke when <b>source=BO</b> .  This mnemonic must be used in conjunction with the <b>input=</b> and <b>output=</b> mnemonics to define how information is sent to / received from the business object.	Cx	This means business object code is defined in an earlier column. For example, define <b>C1</b> if column 1 defines the business object.
		COLUMN_NAME	This means the business object was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'Business Object Code'	This means the business object is defined directly. For example 'F1-BundleImport'.

Mnemonic	Description	Valid Values	Comments
<b>bs=</b>	<p>Defines the business service to invoke when <b>source=BS</b>.</p> <p>This mnemonic must be used in conjunction with the <b>input=</b> and <b>output=</b> mnemonics to define how information is sent to/received from the business service.</p>	Cx	This means business service code is defined in an earlier column. For example, define <b>C1</b> if column 1 defines the business service.
		COLUMN_NAME	This means the business service was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'Business Service Code'	> This means the business service is defined directly. For example 'F1-GetCountryStates'.
<b>ss=</b>	<p>Defines the service script to invoke when <b>source=SS</b>.</p> <p>This mnemonic must be used in conjunction with the <b>input=</b> and <b>output=</b> mnemonics to define how information is sent to / received from the service script.</p>	Cx	This means service script code is defined in an earlier column. For example, define <b>C1</b> if column 1 defines the service script.
		COLUMN_NAME	This means the service script was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'Service Script Code'	This means the service script is defined directly. For example 'F1-ReturnBtn'.
<b>formula=</b>	<p>Defines the formula to use when <b>source=FORMULA</b>.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>• <b>formula=C1*.90/C2</b></li> <li>• <b>formula=(C1/C2)*100</b></li> </ul>	The formula can contain numeric constants, operators and column references.	<p>For column references, use the format Cx where x represents the column number.</p> <p>The following operators are supported: *, /, -, +, ABS, NEGATE, ROUND, FLOOR (round down), CEILING (round up), SIN, ASIN, COS, ACOS, TAN, ATAN, LOG, LOG10, EXP, EXP10, SQRT, **.</p>
<b>setfunc=</b>	<p>Defines the function to apply to the rows of a given column when <b>source=SETFUNC</b>.</p>	function(Cx)	<p>Where Cx represents a column whose rows should have the function applied and the function is one of the following</p> <ul style="list-style-type: none"> <li>• <b>MAX</b>. This derives the maximum value of all rows in the column.</li> <li>• <b>MIN</b>. This derives the minimum value of all rows in the column.</li> <li>• <b>TOT</b>. This derives the sum (total value) of all rows in the column.</li> <li>• <b>ACC</b>. This derives the cumulative total of all rows up to an including the current row.</li> </ul>
<b>input=</b>	<p>This is used to define one or more input fields and values passed to business objects, business services, service scripts, and FK references.</p> <p>The syntax is as follows: [ELEMENT_NAME=ELEMENT_REF ELEMENT_NAME=ELEMENT_REF ...]</p> <p>In other words, the list of input values is surrounded by square brackets separated by a space. Each passed value first defines the ELEMENT_NAME, which is the name of the element / field in the target. ELEMENT_REF is the value passed in. The next column indicates the possible values for ELEMENT_REF.</p>	Cx	Where Cx represents the value of a previous column. If the value to pass is in the first column, reference C1.
		COLUMN_NAME	This means the value to pass in was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'literal value'	This means a literal value within the single quotes should be passed in.
		<b>userTimeZone</b>	This means the current user's time zone should be passed in. This is typically used with the business service <b>F1-ShiftDateTime</b> to convert data in the storage time zone to the user's time zone for display.
		Examples:	<ul style="list-style-type: none"> <li>• <b>input=[USER_ID=C1]</b></li> </ul>

Mnemonic	Description	Valid Values	Comments
		<ul style="list-style-type: none"> <li>input=[USER_ID=USER_ID]</li> <li>input=[input/targetTimeZone=userTimeZone]</li> </ul>	
output=	This is used to define the name of the element retrieved from the business object, business service or service script used to populate this column.	elementName	Example: <b>output=personInfo</b>

## Formatting Mnemonics

This table describe the mnemonics that control how a column is formatted.

Mnemonic	Description	Valid Values	Comments
type=	<p>Defines how the column's value is formatted.</p> <p><b>NOTE: Icon and Foreign Key columns.</b> The <b>source=source mnemonic</b> may be used to indicate a column should be derived from an icon reference or a foreign key (FK) reference. If you use either of these sources, the <b>type=</b> mnemonic is not relevant as either an icon or a context menu / info string will appear in the column.</p>	<b>STRING</b>	Columns of this type capture a string. This is the default value.
		<b>DATE</b>	Columns of this type capture a date.
		<b>TIME</b>	Columns of this type capture a time (in database format) and will be displayed using the user's <a href="#">display profile</a> .
		<b>DATE/TIME</b>	Columns of this type capture a date and time (in database format) and will be displayed using the user's <a href="#">display profile</a> .
		<b>MONEY</b>	Columns of this type capture a monetary field. This type of column may also reference the <b>cur=</b> mnemonic. If the cur mnemonic is not specified, the currency code on the installation record is used.
label=	<p>Defines the column's override label. The label appears in the column's heading and in the zone's drag and drop area.</p> <p>If this mnemonic is not defined, the system uses the column's default label. The source of a column's default label differs depending on the column's <a href="#">source</a>. Note that some sources don't have a default value and omitting this mnemonic will result in a blank label.</p>	FIELD_NAME	Enter a valid <a href="#">field</a> name whose label should be used for the column label. This should always be the option used if multiple languages are needed.
		'text'	Defines the text directly.
cur=	<p>Defines the <a href="#">currency code</a> applied when <b>type=MONEY</b> if the installation record's currency should not be used.</p>	Cx	This means currency code value is defined in an earlier column. For example, define <b>C1</b> if column 1 defines the currency code.
		COLUMN_NAME	This means the currency code was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'Currency Code'	This means the currency code is defined directly. For example 'USD'.
dec=	<p>Defines the number of decimal places when <b>type=NUMBER</b>.</p> <p>It is optional. If provided it should be an integer. If not provided, the number of decimals will default to the number of decimal places defined on the currency code specified on the installation record.</p>	nR	<p>Where n is the number of decimal places to show. Suffixing the number of decimal places with R means that the system should round up / down.</p> <p>Simply specifying n (without an R) means that decimal places should be truncated. For example, entering</p>

Mnemonic	Description	Valid Values	Comments
			<p><b>dec=4</b> will display 4 decimal places and truncate the remainder.</p> <p><b>NOTE: Formatting only.</b> This mnemonic is only used for formatting, it does not impact the precision used for subsequent calculations. For example, if a column retrieved from the database contains 6 significant digits and <b>dec=0</b>, the column will be shown with no decimal places (truncated), however any references to the column in subsequent will use 6 decimal places. For example, if the column is referenced in a formula or set function, all 6 decimal places will be used.</p>
<b>char=</b>	This mnemonic applies special character(s) to the column's value.	'x[]x'	<p>Where x references the literal value to display and [] defines the relative position of the characters (before or after the value).</p> <p>You need only include the [] if you want to position characters in front of the value. For example, <b>char='%'</b> will place a percent sign after the value. If you want to position the word 'minutes' before a value, enter <b>char='minutes []'</b>. If you want to output a value like <b>BUDGET \$123.12 (YTD)</b>, enter <b>char='BUDGET [] (YTD)'</b>.</p>
<b>suppress=</b>	<p>This is used to indicate a column should not be displayed.</p> <p>A column would be suppressed if it's only defined for use by subsequent columns, for example, if there is a formula that derives a column using two other columns. In this scenario, the columns referenced in the formula can be suppressed.</p>	<p><b>true</b></p> <p><b>false</b></p>	<p>This is the default, meaning that you don't need to indicate <b>suppress=false</b> at all to indicate that the field should be shown.</p>
<b>suppressSearch=</b>	This is used to indicate a column should not be displayed when the zone is invoked in search mode only.	<p><b>true</b></p> <p><b>false</b></p>	<p>This is the default, meaning that you don't need to indicate <b>suppressSearch=false</b> at all to indicate that the field should be shown.</p>
<b>suppressExport=</b>	This is used to indicate a column should not be downloaded to Excel.	<p><b>true</b></p> <p><b>false</b></p>	<p>This is the default, meaning that you don't need to indicate <b>suppressExport=false</b> at all to indicate that the field should be included in a download.</p>
<b>width=</b>	This is used to override the width of a column (number of pixels). The default value is the maximum width of any cell in the column.	n	<p>Where n is a number between 0 and 999.</p> <p><b>NOTE:</b> If there is no available breaking point in the data, the column will be longer than the specified number of pixels.</p> <p>The length of the column's label (which appears in the column's heading) may also make the width wider than specified.</p>

Mnemonic	Description	Valid Values	Comments
<b>color=</b>	This is used to override the column's text color.	A valid HTML "named" color	For example <b>color=red</b> or <b>color=yellow</b> .
		A valid RGB color model combination	For example <b>color=#FF0000</b> or <b>color=#CCCCCC</b> . Note that the # is required.
<b>bgcolor=</b>	This is used to override the column's background color.	A valid HTML "named" color	Similar to the <b>color=</b> mnemonic.
		A valid RGB color model combination	Similar to the <b>color=</b> mnemonic.
<b>order=</b>	Defines the column's default sort order.	<b>ASC</b>	Indicates that the order is ascending. This is the default meaning that it is not necessary to indicate <b>order=ASC</b> .
		<b>DESC</b>	Indicates that the order is descending.

## Click Mnemonics

This table describe the mnemonics that define whether a column value may be clicked and if so, what should happen.

Mnemonic	Description	Valid Values	Comments
<b>navopt=</b>	<p>Defines the navigation option that references the target transaction or script when the user clicks a column.</p> <p>Note, this mnemonic should be used in conjunction with the <b>context=</b> mnemonic to define what information is sent to the navigation option's target transaction.</p> <p>This mnemonic is ignored if <b>type=fkref</b> because the FK reference code defines the hyperlink's destination.</p>	Cx	This means navigation option code is defined in an earlier column. For example, define <b>C1</b> if column 1 defines the navigation option.
		COLUMN_NAME	This means the navigation option was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause. Example: <b>navopt=MAIN_PORTAL</b>
		'Navigation Option Code'	This means the navigation option code is defined directly. For example <b>navopt='userMaint'</b> .
<b>context=</b>	<p>This is used to define one or more context fields and values passed to the target navigation option to go along with the <b>navopt=</b> mnemonic.</p> <p>The syntax is as follows: [FIELD_NAME=FIELD_REF FIELD_NAME=FIELD_REF ...]</p> <p>In other words, the list of input values is surrounded by square brackets separated by a space. Each passed value first defines the FIELD_NAME, which is the name of the context field in the navigation option. FIELD_REF is the value passed in. The next column indicates the possible values for FIELD_REF.</p>	Cx	Where Cx represents the value of a previous column. For example, if the value to pass is in the first column, reference <b>C1</b> .
		COLUMN_NAME	This means the value to pass in was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'literal value'	This means a literal value within the single quotes should be passed in.
<b>bpa=</b>	<p>Indicates that a BPA script should be executed with the user clicks the column and indicates the BPA to execute.</p> <p>Note, this mnemonic should be used in conjunction with the <b>tempstorage=</b> mnemonic to define the temporary storage values that will be initiated when the script is executed.</p> <p>This mnemonic is ignored if <b>type=fkref</b> because the FK reference code defines the hyperlink's destination.</p>	Cx	Indicates that the BPA script is defined in a previous column.
		COLUMN_NAME	This means the BPA script to execute was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'BPA Script Code'	This means that the BPA script to execute is defined directly.

Mnemonic	Description	Valid Values	Comments
<b>tempstorage=</b>	<p>This is used to define how temporary storage variables are initiated when the <b>bpa=</b> mnemonic is used.</p> <p>The syntax is as follows: [FIELD_NAME=FIELD_REF FIELD_NAME=FIELD_REF ...]</p> <p>In other words, the list of input values is surrounded by square brackets separated by a space. Each passed value first defines the FIELD_NAME, which is the name of the field in temporary storage. FIELD_REF is the value passed in. The next column indicates the possible values for FIELD_REF.</p>	<p>Cx</p> <hr/> <p>COLUMN_NAME</p> <hr/> <p>'literal value'</p>	<p>Where Cx represents the value of a previous column. For example, if the value to pass is in the first column, reference <b>C1</b>.</p> <hr/> <p>This means the value to pass in was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.</p> <hr/> <p>This means a literal value within the single quotes should be passed in.</p>
<b>list=</b>	<p>This is used to enable <a href="#">work list</a> capability for this column.</p> <p>You may optionally populate the <b>listdesc=</b> mnemonic to override the text that will be placed in the worklist zone.</p>	<b>true</b>	<p>Setting <b>list=true</b> will cause the work list icon to appear in the column's header. If a user clicks the column, it will populate all the rows in the output into the <a href="#">work list zone</a>.</p> <p><b>NOTE:</b> In the case of the zone type Info Data Explorer - Multiple SQLs (<b>F1-DE</b>), the output may be showing a union of the results of multiple SQL statements. In this case, if some of the SQL statements configure a given column with <b>list=true</b>, but not all, only the data in the cells for the statements that configure this mnemonic are put into the work list when the user clicks the icon.</p>
<b>listdesc=</b>	This is an optional mnemonic when using the <b>list=</b> mnemonic. It can be used to override the text that is placed in the work list zone.	Cx	Where Cx represents the value of a previous column. For example, if the text to use is in the first column, reference <b>C1</b> .
<b>listbroadcast=</b>	Indicates that the broadcast information for the column is also to be made available in the work list zone. This means that the work list can be used to broadcast information to a portal in the same manner as a data explorer.	<b>true</b>	Use this setting to turn on the feature.

## Zone Action

Most zone types provided by the product allow for one or more Zone Actions to be defined to appear in the zone header. An action can appear as a hyperlink, icon or button. The action can also be provided as an HTML string.

**NOTE:** Zone types also include parameters for actions defined at the zone type level using IMPLEMENTOR\_ACTION\_n (Action n) parameters. These are rarely used by the product zone types. The actions defined here override any actions defined on the zone type (if present). The details below apply to the zone type level actions as well.

A zone action is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
<b>type=</b>	This mnemonic defines the appearance of the action in the zone header.	<p><b>LINK</b></p> <hr/> <p><b>ICON</b></p>	<p>Indicates that the action is shown as a textual hyperlink.</p> <hr/> <p>Indicates that the action is shown as a graphical icon.</p>



Mnemonic	Description	Valid Values	Comments
		<b>BUTTON</b>	Indicates that the action is shown as an HTML button.
		<b>ASIS</b>	Indicates that the parameter will provide the HTML to be used for the action.
<b>action=</b>	This mnemonic defines the action to take when the link/icon/button is clicked. This is ignored when the type=ASIS.	<b>NAVIGATION</b>	Indicates that the action is navigation to a page.
		<b>SCRIPT</b>	Indicates that the action is to run a BPA script.
<b>navopt=</b>	Defines the navigation option to use when the <b>action=NAVIGATION</b> .	'NAV_OPT_CD'	Enter a reference to a valid navigation option in single quotes.
<b>bpa=</b>	Defines the script to run when the <b>action=SCRIPT</b> .	'SCRIPT_CD'	Enter a reference to a valid BPA script in single quotes.
<b>icon=</b>	Indicates the icon to use when <b>type=ICON</b> .	DISP_ICON_CD	Enter a reference to a valid display icon.
		'path'	Enter an explicit path to the icon, for example 'images/gotoZone.gif'.
<b>asis=</b>	This is required when the <b>type=ASIS</b> . This provides the ability to precisely define the HTML you wish to have included in the header. All valid HTML is permitted including the use of "ora" css classes and JavaScript functions.	['HTML']	
<b>label=</b>	By default, the label or tooltip will come from the navigation option or BPA script description. Use this mnemonic to override that label.	FIELD_NAME	Enter a valid field name whose label should be used. This should always be the option used if multiple languages are needed.
		'text'	Enter the text directly in single quotes.
<b>context=[target1=source1 target2=source2]</b>	This is used to pass context data when navigating to a page or executing a BPA script. The mnemonic supports passing multiple values.  In each case the target context field or BPA script variable is defined first followed by an equal sign, followed by source data defined using one of the valid values defined in the next column.  One or more values may be defined. Each context value is defined separated by spaces. The whole set of context values should be surrounded by square brackets.	FIELD_NAME	Indicates that the value should be taken from the field with this name from portal context, global context or the page data model. The mnemonic sourceLoc is used for defining the source.
		xpath	Indicates that the value should be taken from a schema field, represented by the Xpath, displayed in this zone. This is valid when the zone is displaying a UI Map.
		'constant'	Indicates that the value defined in single quotes should be passed.
<b>sourceLoc=</b>	This mnemonic defines the source of the FIELD_NAME's value in the context mnemonic.  If this mnemonic is left blank, the default behavior is as follows:	<b>G</b>	Indicates that the field's value is retrieved from the global context.
		<b>P</b>	Indicates that the field's value is retrieved from the portal context.

Mnemonic	Description	Valid Values	Comments
	<ul style="list-style-type: none"> <li>- The portal context is checked.</li> <li>- If no portal context value is found, the global context is checked.</li> <li>- If neither value is available, the field is ignored.</li> </ul>	D	Indicates that the field's value is retrieved from the page data model.
<b>class=</b>	Use this mnemonic to override the look and feel of the link / icon / button using a different CSS style.	'className1' 'className2'	Enter one or more classes in single quotes. Multiple class names may be provided.
<b>style=</b>	Use this mnemonic to override the look and feel of the action element using the indicated css style.	Standard style= format.	All allowed css style definitions may be used.

Examples:

- **type=BUTTON action=SCRIPT bpa='F1-SET-USER' context=[USER\_ID=USER\_ID] label=UPDATE\_LBL**
- **type=LINK action=NAVIGATION navopt='gotoUser' context=[USER\_ID=path(schema/userId)]**
- **type=ASIS asis=['<A class="oraLink" href="www.google.com">Search</a>']**

**NOTE:** If the zone type has actions defined and there is a desire to simply remove the zone type actions, the Zone Action can be set with the following configuration: **type=ASIS asis=[]**

## User Filters

Data explorer zones include the ability to define User filters to allow a user to enter data to restrict the zone's rows and / or columns. The filters may be defined individually using User Filter parameters 1–25. Alternatively, a UI map may be defined for capturing filters. In this case, the map's input fields must be associated with the zone's filters by specifying the **xpath=** mnemonic on the respective User Filter parameters.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1-DE**)
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**)
- Info Data Explorer - Single SQL (**F1-DE-SINGLE**)

A user filter is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
<b>name=</b>	This mnemonic is used if the zone's filter should be pre-populated with a value from global context, portal context or broadcast from another zone.	FIELD_NAME	
<b>datasource=</b>	<p>This mnemonic defines the source of the filter's pre-populated value defined in the name mnemonic.</p> <p>If this mnemonic is left blank, the default behavior is as follows:</p> <ul style="list-style-type: none"> <li>- If the field has been broadcast from another zone, the broadcast value is used.</li> <li>- If no value is broadcast, the portal context is checked to determine if this field exists (if so, its value is taken).</li> </ul>	<p><b>G</b></p> <hr/> <p><b>P</b></p> <hr/> <p><b>D</b></p>	<p>Indicates that the zone should look for the filter value in global context.</p> <hr/> <p>Indicates that the zone should look for the filter value in portal context.</p> <hr/> <p>Indicates that the zone should look for the filter value in the page data model.</p>

Mnemonic	Description	Valid Values	Comments
	<p>- If still no value, the global context is checked.</p> <p>- If still no value, no default value is shown.</p>		
<b>type=</b>	Defines the visual metaphor used to capture the filter values.	<b>DATE</b> <b>DATE/TIME</b> <b>STRING</b> <b>MONEY</b> <b>NUMBER</b> <b>LOOKUP</b> <b>TABLE</b> <b>CHARTYPE</b> <b>ASIS</b>	<p>Filters of this type capture a date.</p> <p>Filters of this type capture a date and time.</p> <p>Filters of this type capture a string</p> <p>Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic.</p> <p>Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic.</p> <p>Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic.</p> <p>Filters of this type capture an administrative table's value (code and description). This type of filter must also reference the table mnemonic.</p> <p>Filters of this type capture predefined characteristic values for a characteristic type (code and description). This type of filter must also reference the chartype mnemonic.</p> <p>Filters of this type capture a list of values to be referenced within an 'IN' clause within the SQL statement.</p>
<b>label=</b>	Defines the filter's label that appears in the zone's description bar and in the input area.	<b>FIELD_NAME</b>  'text'	<p>Enter a valid field name whose label should be used for the filter label. This should always be the option used if multiple languages are needed.</p> <p>Defines the text directly.</p>
<b>cur=</b>	Defines the currency code applied when <b>type=MONEY</b> .	<b>CURRENCY_CD</b>	Enter a reference to a valid currency code.
<b>dec=</b>	Defines the number of decimal places when <b>type=NUMBER</b> .	N	It is optional. If provided it should be an integer. If not provided, the number of decimals will default to the number of decimal places defined on the currency code specified on the installation record.
<b>lookup=</b>	Defines the lookup flag whose values appear when <b>type=LOOKUP</b> .	<b>LOOKUP_FIELD_NAME</b>	Enter a reference to a valid lookup field name.
<b>table=</b>	Defines the admin table whose values appear when <b>type=TABLE</b> .	<b>TABLE_NAME</b>	Enter a reference to a valid admin table name.
<b>chartype=</b>	Defines the characteristic type code whose values appear when <b>type=CHARTYPE</b> .	<b>CHAR_TYPE_CD</b>	Enter a reference to a valid characteristic type code.
<b>xpath=</b>	This mnemonic is used in conjunction with a Filter Area UI Map. For each filter, you must specify the xpath to the corresponding UI map schema element.	nameFromUIMap	The <b>type=</b> mnemonic must also be appropriate for the map's input field, otherwise the query's SQL could fail.
<b>likeable=</b>	This mnemonic defines if a likeable search is performed on the entered value when <b>type=STRING</b> .	<b>S</b> <b>P</b>	<p>The query will add % to the suffix of the filter value.</p> <p>The query will add % to the prefix of the filter value.</p>

Mnemonic	Description	Valid Values	Comments
		<b>PS</b>	The query will add % to the prefix and suffix of the filter value.
<b>divide=</b>	The mnemonic controls if a divider line appears above and/or below the filter.  Note, you can specify this parameter twice if you want divider lines placed above and below a filter, e.g., <b>divide=above divide=below</b> .	<b>above</b>  <b>below</b>	This results in a divider line placed above the filter.  This results in a divider line placed below the filter.
<b>searchField=</b>	This mnemonic controls the initial population of the filter when the zone is launched as a search from a UI map.	FIELD_NAME	Enter the field name that exactly matches the searchField name specified in the oraSearchField html element in the UI map.
<b>encrypt=</b>	This mnemonic defines if the user filter is encrypted and needs to be searched by hashed value.	[TBL_NAME,FLD_NAME,WHERE_FLD,WHERE_VALUE]	A valid table name and field name are required.  The WHERE_FLD and WHERE_VALUE are optional, but if entered, both are required. Use this to only encrypt the field in FIELD_NAME if another field has a certain value. For example encrypt=[CI_PERSON,PER_ID_NBR,ID_TYPE_NBR,SSN].

Examples:

- **label=F1\_NBR\_DAYS type=NUMBER**
- **label=F1\_SHOW\_ALL\_REQ\_FLG type=LOOKUP lookup=F1\_SHOW\_ALL\_REQ\_FLG**
- Filter value where a Filter UI Map is defined and Description is one of the filters. **type=STRING xpath=description likeable=S**
  - **type=STRING label=DESCR likeable=S divide=below**
  - **label=REQ\_TYPE\_CD type=TABLE table=F1\_REQ\_TYPE**

## Hidden Filters

Data explorer zones include the ability to define Hidden filters to restrict the rows and / or columns that appear in the zone. The following are the potential sources of a hidden filter's value:

- The global area contains the fields whose values are maintained in [global context](#).
- The portal area contains the fields describing the object currently displayed in a portal.
- Other zones on a portal can broadcast information to the portal area, which can then in turn be used by the zone as a hidden filter.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1-DE**)
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**)
- Info Data Explorer - Single SQL (**F1-DE-SINGLE**)

A hidden filter is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
<b>name=</b>	This mnemonic defines the name of the field that needs to be broadcast from other zones or populated in the portal context	FIELD_NAME	
<b>datasource=</b>	This mnemonic defines the source of the hidden filter's value.	<b>G</b>	Indicates that the zone should look for the filter value in global context.

Mnemonic	Description	Valid Values	Comments
	<p>If this mnemonic is left blank, the default behavior is as follows:</p> <ul style="list-style-type: none"> <li>- If the field has been broadcast from another zone, the broadcast value is used.</li> <li>- If no value is broadcast, the portal context is checked to determine if this field exists (if so, its value is taken).</li> <li>- If still no value, the global context is checked.</li> <li>- If still no value, the zone appears as per the poprule mnemonic.</li> </ul>	<p><b>P</b></p> <hr/> <p><b>D</b></p>	<p>Indicates that the zone should look for the filter value in portal context.</p> <hr/> <p>Indicates that the zone should look for the filter value in the page data model.</p>
<b>poprule=</b>	This mnemonic controls what happens if the hidden filter is not present.	<p><b>R</b></p> <hr/> <p><b>O</b></p>	<p>Indicates that a value for the filter is required. The zone will be set to the "empty state" and the "please broadcast" message will appear in the zone. This is the default value.</p> <hr/> <p>Indicates that the value is optional. If no value is required, the zone is still built without that value.</p>
<b>type=</b>	Defines the visual metaphor used to capture the filter values.	<p><b>DATE</b></p> <hr/> <p><b>DATE/TIME</b></p> <hr/> <p><b>STRING</b></p> <hr/> <p><b>MONEY</b></p> <hr/> <p><b>NUMBER</b></p> <hr/> <p><b>LOOKUP</b></p> <hr/> <p><b>TABLE</b></p> <hr/> <p><b>CHARTYPE</b></p> <hr/> <p><b>ASIS</b></p>	<p>Filters of this type capture a date.</p> <hr/> <p>Filters of this type capture a date and time.</p> <hr/> <p>Filters of this type capture a string</p> <hr/> <p>Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic.</p> <hr/> <p>Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic.</p> <hr/> <p>Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic.</p> <hr/> <p>Filters of this type capture an administrative table's value (code and description). This type of filter must also reference the table mnemonic.</p> <hr/> <p>Filters of this type capture predefined characteristic values for a characteristic type (code and description). This type of filter must also reference the chartype mnemonic.</p> <hr/> <p>Filters of this type capture a list of values to be referenced within an 'IN' clause within the SQL statement.</p>
<b>label=</b>	Defines the filter's label that appears in the zone's description bar.	<p>FIELD_NAME</p> <hr/> <p>'text'</p>	<p>Enter a valid field name whose label should be used. This should always be the option used if multiple languages are needed.</p> <hr/> <p>Defines the text directly.</p>
<b>cur=</b>	Defines the currency code applied when <b>type=MONEY</b> .	CURRENCY_CD	Enter a reference to a valid currency code.
<b>dec=</b>	Defines the number of decimal places when <b>type=NUMBER</b> .	N	It is optional. If provided it should be an integer. If not provided, the number of decimals will default

Mnemonic	Description	Valid Values	Comments
			to the number of decimal places defined on the currency code specified on the installation record.
<b>lookup=</b>	Defines the lookup flag whose values appear when <b>type=LOOKUP</b> .	LOOKUP_FIELD_NAME	Enter a reference to a valid lookup field name.
<b>table=</b>	Defines the admin table whose values appear when <b>type=TABLE</b> .	TABLE_NAME	Enter a reference to a valid admin table name.
<b>chartype=</b>	Defines the characteristic type code whose values appear when <b>type=CHARTYPE</b> .	CHAR_TYPE_CD	Enter a reference to a valid characteristic type code.
<b>searchField=</b>	This mnemonic controls the initial population of the filter when the zone is launched as a search from a UI map.	FIELD_NAME	Enter the field name that exactly matches the searchField name specified in the oraSearchField html element in the UI map.

## Multi-Select Action

This parameter defines an action to be included in the action area for multi-selection processing. Note that a multi-selection action can only be used if the Multi Select parameter has been set to YES, which causes a checkbox to appear on each row displayed. The action defined here will trigger against all rows selected by the user via the checkbox.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1-DE**)
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**)
- Info Data Explorer - Single SQL (**F1-DE-SINGLE**)

A multi select action has the following mnemonics:

Mnemonic	Description	Valid Values	Comments
<b>script=</b>	This mnemonic defines the script to be invoked when the action is clicked. This is required.	SCR_CD	Enter a reference to a valid BPA script or Service Script.
<b>type=</b>	This mnemonic defines how the action should be rendered.	<b>BUTTON</b>	The action is rendered as a button. This is the default.
		<b>LINK</b>	The action is rendered as hypertext.
		<b>ICON</b>	The action is rendered as a graphic icon. For this option, the icon mnemonic is required.
<b>icon=</b>	This mnemonic defines the icon to display when <b>type=ICON</b> .	DISPLAY_ICON_CD	Enter a reference to a valid display icon code.
<b>refresh=</b>	This mnemonic indicates how and if a refresh should occur after the script completes.	<b>NO</b>	Indicates that no refresh is performed. This is the default.
		<b>ZONE</b>	Indicates that a refresh of the zone is performed.
		<b>PORTAL</b>	Indicates that a refresh of the entire portal is performed.
<b>label=</b>	By default, the button label, link text or icon	FIELD_NAME	Enter a valid field name whose label

Mnemonic	Description	Valid Values	Comments
	tooltip will come from the script description. Use this mnemonic to override that label.	'text'	should be used. This should always be the option used if multiple languages are needed.  Enter the text directly in single quotes.
<b>list=</b>	When executing the script, the framework builds an XML list containing information from each row selected. This list must be defined in the script's schema and referenced in this mnemonic.	listElementName	Enter a valid list element name from the script schema.
<b>context=[elementName1=rowData1 elementName2=rowData2]</b>	<p>This mnemonic is used to populate the list with the appropriate information from each selected row. The mnemonic supports passing multiple values.</p> <p>In each case the element in the schema list is defined first followed by an equal sign, followed by information about the data used to populate the element defined using one of the valid values defined in the next column.</p> <p>One or more values may be defined. Each context value is defined separated by spaces. The whole set of context values should be surrounded by square brackets.</p> <p>Example of a schema:</p> <pre data-bbox="610 1314 935 1482">&lt;schema&gt;   &lt;accountInfo type="list"&gt;     &lt;accountId/&gt;     &lt;name/&gt;     &lt;amount/&gt;     &lt;process/&gt;   &lt;/accountInfo&gt; &lt;/schema&gt;</pre> <p>Example of list and context mnemonics.</p> <p><b>list=accountInfo</b>  <b>context=[accountId=ACCT_ ID name=C2 amount=P3 process='O']</b></p>	<p>Cx</p> <hr/> <p>Px</p> <hr/> <p>COLUMN_NAME</p> <hr/> <p>'constant'</p>	<p>Indicates that the element should be populated with a value in the referenced column parameter.</p> <hr/> <p>Indicates that the element should be populated with a value in the referenced post processing parameter.</p> <hr/> <p>Indicates that the element should be populated with a value from a column in the SQL statement.</p> <hr/> <p>Indicates that the value defined in single quotes should be passed.</p>
<b>class=</b>	Use this mnemonic to override the look and feel of the action using a different CSS style.	'className1' 'className2'	Enter one or more classes in single quotes to be appended to the standard class(es). Multiple class names may be provided.

Mnemonic	Description	Valid Values	Comments
style=	Use this mnemonic to override the look and feel of the action element using the indicated css style.	Standard style= format.	All allowed css style definitions may be used.

## Defining Context-Sensitive Zones

A context-sensitive zone allows you to associate a zone with a specific user-interface transaction. A context-sensitive zone appears at the top of the Dashboard when a user accesses a page for which the zone is specified as the context. For example, when viewing a business object, additional zones are visible that are specific to the business object page.

**CAUTION:** Make sure that the zone is appropriate for the transaction on which you are specifying it. For example, if your zone requires a business object ID as one of its keys, it should not be displayed on the To Do entry transaction.

Select **Admin > Context Sensitive Zone** to maintain context-sensitive zones.

### Description of Page

The **Navigation Key** is a unique identifier of a tab page within the system. **Owner** indicates if this navigation key is owned by the base product or by your implementation (**Customer Modification**).

**CAUTION:** Important! When introducing a new context sensitive zone, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

The grid contains the list of context-sensitive zones and the sequence in which they appear in the dashboard for the selected navigation key. The grid contains the following fields:

- **Zone** is the name of the zone to display in the Dashboard.
- **Sequence** is the sequence in which the zone is displayed (if multiple context-sensitive zones are defined).
- **Owner** indicates if this context sensitive zone is owned by the base product or by your implementation (**Customer Modification**).

### Where Used

A context-sensitive zone displays at the top of the Dashboard whenever a user accesses the transaction for with the zone is specified.

## Defining Portals

This transaction is used to define / change portals. An implementation may define their own portals. In addition, an implementation may override some of the settings for base product provided portals.

### Portal - Main

Navigate to this page using **Admin > System > Portal**.

#### Description of Page

Enter a meaningful and unique **Portal** code and **Description**. Please be aware that for [stand-alone portals](#), the Description is the portal's title (i.e., the end-users will see this title whenever they open the portal).



---

**CAUTION:** Important! When introducing a new portal, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

**Owner** indicates if this portal is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a portal. This information is display-only.

**Type** flag indicates whether the portal is a **Standalone Portal**, a **Tab Page Portal** or the **Dashboard**. Refer to [There Are Three Types of Portals](#) for more information.

The following fields are only enabled for **Standalone Portals**:

- **Navigation Option** defines the navigation option that is used to navigate to this portal from menus, scripts and your favorite links. The navigation option is automatically created when a **Standalone Portal** is added.
- You'll find an **Add To Menu** button adjacent. This field is only enabled if the navigation option is not referenced on a menu. When you click this button, a pop-up appears where you define a menu. If you subsequently press **OK**, a menu item is added to the selected menu. This menu item references the portal's navigation option. You can reposition the menu item on the menu by navigating to the [Menu page](#). Refer to [Putting Portals on Menus](#) for more information.
- **Application Services** defines the service used to secure this portal. The application service is automatically created when a **Standalone Portal** is added. Please note that only users with access to this application service will be able to view this portal and its zones. Refer to [Granting Access to A Portal](#) for more information.
- **Show on Portal Preferences** indicates if a user is allowed to have individual control of the zones on this portal. The portal will not appear in the accordion on the user's Portal Preferences page if this value is set to No. Note that an implementation may change this value for a product delivered portal.

The grid contains a list of zones that are available in the portal. Click + to add a new zone to the portal. Click - to remove a zone from the portal. The grid displays the following fields:

- **Zone** is the name of the zone as defined on the Zone page.
- **Description** is a description of the zone as defined on the Zone page.
- **Display** controls whether or not the zone is visible in the portal. For portals that are configured to Show on Portal Preferences, users may override this value for their view of the portal.
- **Initially Collapsed** controls whether or not the zone is initially collapsed in the portal. For portals that are configured to Show on Portal Preferences, users may override this value for their view of the portal.
- **Default Sequence** is the default sequence number for the zone within the portal. It does not need to be unique within the portal. Note that a sequence of zero will appear last, not first, in the portal. For portals that are configured to Show on Portal Preferences, users may override this value for their view of the portal.
- **Override Sequence** can be used by an implementation team to override the Default Sequence value that is set in the base product.
- **Refresh Seconds** defines in seconds how often the zone is automatically refreshed. The minimum valid value is 15. The maximum valid value is 3600 (1 hour). A value of 0 indicates no automatic refresh. Implementers can change this value as needed.
- **Owner** indicates if this portal / zone relationship is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

---

**NOTE: Removing zones from a portal.** You cannot remove a base product zone from a base product portal. An implementation may override the Display setting to prevent a zone from displaying on the portal. In addition, you cannot remove a zone if a user has enabled it on their Portal Preferences. To remove a zone from the portal list, first make sure that no user has it enabled in their portal preferences.

---

## Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_PORTAL](#).

## Portal - Options

Use this page to maintain a portal's options. Open this page using **Admin > System > Portal** and then navigate to the **Options** page.

### Description of Page

The options grid allows you to configure the options that provide additional information related to the portal.

Select the **Option Type** dropdown to define its **Value**. **Detailed Description** may display additional information on the option type.

Set the **Sequence** to **1** unless the option can have more than one value.

**Owner** indicates if this is owned by the base product or by your implementation (**Customer Modification**).

---

**NOTE: You can add new option types.** Your implementation may want to add additional portal option types. To do that, add your new values to the customizable lookup field **PORTAL\_OPT\_FLG**.

---

## Defining Display Icons

Icons are used to assist users in identifying different types of objects or instructions. A limited number of control tables allow administrative users to select an icon when they are configuring the system. Select **Admin > System > Display Icon Reference** to maintain the population of icons available for selection.

### Description of Page

Each icon requires the following information:

- **Display Icon** is a code that uniquely identifies the icon.
- **Icon Type** defines how big the icon is (in pixels). The permissible values are: **30 x 21**, **21 x 21**, and **20 x 14**. Note that only icons that are **20 x 14** can be used on base product instructions.
- **Description** contains a brief description of the icon.
- **URL** describes where the icon is located. Your icons can be located on the product's web server or on an external web server.
- To add a new icon to the product web server, place it under the **/cm/images** directory under the **DefaultWebApp**. Then, in the **URL** field, specify the relative address of the icon. For example, if the icon's file name is **myIcon.gif**, the **URL** would be **/cm/images/myIcon.gif**.
  - If the icon resides on an external web server, the **URL** must be fully qualified (for example, **http://myWebServer/images/myIcon.gif**).
  - **Owner** indicates if this icon is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL\\_DISP\\_ICON](#).

## Defining Navigation Keys

Each location to which a user can navigate (e.g., transactions, tab pages, tab menus, online help links, etc.) is identified by a navigation key. A navigation key is a logical identifier for a URL.

## Navigation Key Types

There are two types of navigation keys:

- **System navigation keys** define locations where the URL is derived from other entities within the system. The items that are derived include the program location (i.e., physical location on the server), file name (i.e. program component name), and the file extension (e.g. COB). System navigation keys refer to program components, such as tab menus or tab pages.
- **External navigation keys** define locations simply as a URL. External URLs can be specified as relative to the product web server or fully qualified. External navigation keys always launch in a new instance of a browser window. Examples of external navigation keys include online help links and URLs to external systems.

## Navigation Key vs. Navigation Option

The system has two entities that work in conjunction with each other to specify how navigation works:

- **Navigation Key** defines a unique location to which a user can navigate. For example, each page in the system has a unique navigation key. Navigation keys can also define locations that are "outside" of the system. For example, you can create a navigation key that references an external URL. Think of a navigation key as defining "where to go".
- **Navigation Option** defines how a page is opened when a user wants to navigate someplace. For example, you might have a navigation key that identifies a specific page. This navigation key can then be referenced on two navigation options; the first navigation option may allow users to navigate to the page in "add mode", while the second navigates to the page in "update mode".
- Please note that a wide variety of options can be defined on a navigation option. In addition to defining if a page is opened in add or update mode, you can define which tab is opened, which fields should be passed to the page, which search program is used, etc.

## The Flexibility of Navigation Keys

Navigation keys provide a great deal of functionality to your users. Use navigation keys to:

- Allow users to navigate to new pages or search programs
- Allow users to transfer to an external system or web page. After setting up this data, your users may be able to access this external URL from a menu, a context menu, their favorite links, etc. [Refer to Linking to External Locations](#) for more information.

Refer to the Tool Suite Guide for more information on developing program components.

---

**NOTE: Replacing Base-Package Pages or Searches.** If your new page or search has been designed to replace a module in the base-package, the navigation key must indicate that it is [overriding an existing navigation key](#).

---

## Linking to External Locations

If you want to include links to external systems or locations from within the system, you need to:

- Define a [navigation key](#) that specifies the URL of the location. For example, define an external navigation key that as a URL of <http://www.oracle.com/>.
- Define a [navigation option](#) that specifies from where in the system a user can go to your external location. For example, define a navigation option with a usage of **Favorites** or with a usage of **Menu**. Your navigation option points to the navigation key you defined above.

- Add your navigation option to the appropriate location within the system. For example, have users add the navigation option to their [Favorite Links](#) or add the navigation option as an item on a [menu](#).

## Overriding Navigation Keys

Your implementation may choose to design a program component (e.g., a maintenance transaction or search page) to replace a component provided by the system. When doing this, the new navigation key must indicate that it is overriding the system navigation key. As a result, any menu entry or navigation options that reference this overridden navigation key automatically navigates to the custom component.

For example, if you have a custom On-line Batch Submission page and would like users to use this page rather than the one provided by the system, setting up an override navigation key ensures that if a user chooses to navigate to the On-line Batch Submission from Menu or a context menu, the user is brought to the custom On-line Batch Submission page.

To create an override navigation key, you need to:

- Define a [navigation key](#) using an appropriate [naming convention](#).
- If the URL Location of the navigation key being overridden is **External**, specify a URL Location of **Override (External)** and define the appropriate URL Override Location.
- If the URL Location of navigation key being overridden is **System**, specify a **URL Location of Override (System)** and populate the Program Component ID with your custom program component ID.
- Specify the navigation key that you are overriding in the **Overridden Navigation Key** field.

Refer to the Tool Suite Guide for more information about developing your own program components.

## Maintaining Navigation Keys

Select **Admin > System > Navigation Key** to maintain navigation keys.

---

**CAUTION:** Important! When introducing a new navigation key, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

### Description of Page

The **Navigation Key** is a unique name of the navigation key for internal use. Try to use a name that is easily recognizable.

**Owner** indicates if this navigation key is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

For **URL Location**, you can select from the following options:

- Use **External** to create a navigation key where the location is specified in the **URL Override** field.
- Use **Override (External)** to create a navigation key that overrides another external navigation key. If you use this option, you specify the name of the navigation key you are overriding in the **Overridden Navigation Key** field.
- Use **Override (System)** to point a system navigation key to a different location. If you use this option, you specify the name of the navigation key you are overriding in the **Overridden Navigation Key** field.
- Use **System** to create a navigation key for a custom program component developed for your implementation.

---

**FASTPATH:** Refer to [Navigation Key Types](#) for more information about system and external navigation keys.

---

**FASTPATH:** Refer to [Overriding Navigation Keys](#) for more information about settings required to override a system navigation key.

---

**Program Component ID** is the name of the program component identified by the key (for system navigation keys). The program component ID can also be used to specify the transaction with which an online help link is associated.

**Overridden Navigation Key** is the name of the navigation key that the current navigation key is overriding (if **Override (External)** or **Override (System)** is selected for the **URL Location**). Refer to [Overriding Navigation Keys](#) for more information.

**URL Override** is the specific URL for the navigation key (external navigation keys only). The URL can be relative to the product web server or fully qualified.

**Open Window Options** allows you to specify options (e.g., width and height) for opening a browser window for an external navigation key. (External navigation keys always launch in a new browser window.) You can use any valid features available in the `Window.open()` JavaScript method. The string should be formatted the same way that it would be for the features argument (e.g., **height=600,width=800,resizeable=yes,scrollbars=yes,toolbar=no**). Refer to a JavaScript reference book for a complete list of available features.

**Application Service** is the application service that is used to secure access to transactions associated with **External** navigation keys. If a user has access to the specified application service, the user can navigate to the URL defined on the navigation key. Refer to [The Big Picture of Application Security](#) for more information.

The grid displays menu items that reference the navigation key (actually, it shows menu items that reference navigations options that, in turn, reference the navigation key).

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CL\\_MD\\_NAV](#).

## Defining Navigation Options

Every time a user navigates to a transaction, the system retrieves a navigation option to determine which transaction should open. For example,

- A navigation option is associated with every menu item. When a user selects a menu item, the system retrieves the related navigation option to determine which transaction to open.
- A navigation option is associated with every [favorite link](#). When a user selects a favorite link, the system retrieves the related navigation option to determine which transaction to open.
- A navigation option is associated with every node in the various trees. When a user clicks a node in a tree, the system retrieves the related navigation option to determine which transaction to open.
- Etc.

Many navigation options are shipped with the base product and cannot be modified as these options support core functionality. As part of your implementation, you may add additional navigation options to support your specific business processes.

Navigation options may also be used to launch a BPA script.

The topics in this section describe how to maintain navigation options.

---

**CAUTION:** In order to improve response times, navigation options are cached the first time they are used after a web server is started. If you change a navigation option and you don't want to wait for the cache to rebuild, you must clear the cached information so it will be immediately rebuilt using current information. A special button has been provided on the Main tab of the navigation option transaction that performs this function. Please refer to [Caching Overview](#) for information on the various caches.

---

## Navigation Option - Main

Select **Admin > System > Navigation Option** to maintain a navigation option.

## Description of Page

Enter a unique **Navigation Option** code and **Description**.

---

**CAUTION:** When introducing a new navigation option, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

The **Flush System Login Info** button is used to flush the cached navigation options so you can use any modified navigation options. Refer to [Caching Overview](#) for more information.

**Owner** indicates if this navigation option is owned by the base product or by your implementation (**Customer Modification**). This field is display-only. The system sets the owner to **Customer Modification** when you add a navigation option.

---

**NOTE:** You may not change navigation options that are owned by the base product.

---

Use **Navigation Option Type** to define if the navigation option navigates to a **Transaction**, launches a BPA **Script** or opens an **Attachment**.

---

**NOTE:** The **Attachment** option type is only applicable to navigation options that are used to launch a file attached to a record in the Attachment maintenance object.

---

For navigation option types of **script**, indicate the **Script** to launch. You can use the **Context Fields** at the bottom of the page if you want to transfer the contents of specific fields to temporary storage variables available to the script. The script engine creates temporary storage variables with names that match the Context Field names.

For navigation option types of **transaction**, define the **Target Transaction** (navigation key) and optionally a specific **Tab Page** (also a navigation key) if a specific tab on the transaction (other than the Main tab) should be opened when navigating to that transaction.

---

**NOTE: Finding transaction navigation keys.** When populating the **Target Transaction** and **Tab Page** you are populating an appropriate navigation key. Because the system has a large number of transactions, we recommend using the "%" metaphor when you search for the transaction identifier. For example, if you want to find the currency maintenance transaction, enter "%currency" in the search criteria.

---

The additional information depends on whether the target transaction is a fixed page or a portal-based page:

- For portal-based pages:
  - **Navigation Mode** is not applicable and should just be set to **Add Mode**.
  - If navigating to a query portal, by default the query portal will open with the default search option defined. If the navigation should open a different search option, define the **Multi-Query Zone** for that query portal and indicate the **Sub-Query Zone** to open by default. Note that for this configuration, it is common to define **Context Fields** to pre-populate search criteria in the target query zone. When using this configuration, be sure that the target query zone's user filters are defined to populate data from context.
- For fixed pages:
  - **Navigation Mode** indicates if the **Target Transaction** should be opened in **Add Mode** or **Change Mode**.
  - **Add Mode** should be used if the option is used to navigate to a transaction ready to add a new object. You can use the **Context Fields** at the bottom of the page if you want to transfer the contents of specific fields to the transaction when it opens.
  - **Change Mode** is only applicable for fixed pages and should be used if the option is used to navigate to a transaction ready to update an object. You have two ways to define the object to be changed:
    - Define the name of the fields that make up the unique identifier of the object in the **Context Fields** (and make sure to turn on **Key Field** for each such field).

- Define the **Search Transaction** (navigation key) if you want to open a search window to retrieve an object before the target transaction opens. Select the appropriate **Search Type** to define which search method should be used. The options in the drop down correspond with the sections in the search (where **Main** is the first section, **Alternate** is the 2<sup>nd</sup> section, **Alternate 2** is the 3<sup>rd</sup> section, etc.). You should execute the search window in order to determine what each section does.

When you select a **Search Type**, define appropriate **Context Fields** so that the system will try to pre-populate the search transaction with these field values when the search first opens. Keep in mind that if a search is populated with field values the search is automatically triggered and, if only one object is found that matches the search criteria, it is selected and the search window closes.

- **Search Group** is only visible if the **Development Tools** module is **not turned off**. It is used to define the correlation between fields on the search page and the tab page. You can view a tab page's **Search Groups** by viewing the HTML source and scanning for **allFieldPairs**.

The **Go To Tooltip** is used to specify the label associated with the tool tip that appears when hovering over a **Go To** object. Refer to the **Usage** grid below.

The **Usage** grid defines the objects on which this navigation option is used:

- Choose **Favorites** if the navigation option can be used as a **favorite link**.
- Choose **Menus** if the navigation option can be used as a user's **home page** or as a menu or context menu item.
- Choose **Script** if the navigation option can be used in a **script**.
- Choose **Foreign Key** if the navigation option can be used as a **foreign key reference**.
- Choose **Go To** if the navigation option can be used as a "go to" destination ("go to" destinations are used on Go To buttons, tree nodes, and hyperlinks).
- If your product supports marketing campaigns, you can choose **Campaign** if the navigation option can be used as a "post completion" transaction on a campaign. For more information refer to that product's documentation for campaigns.

The **Context Fields** grid contains the names of the fields whose contents will be passed to the **Target Transaction** or **Script** or used to launch an **Attachment**. The system retrieves the values of these fields from the "current" page and transfers them to the target transaction or to the script's temporary storage. Turn on **Key Field** for each context field that makes up the unique identifier when navigating to a transaction in **Change Mode**.

---

**NOTE:** For an **Attachment**, the grid should contain the Attachment ID.

---

**NOTE: Navigating from a Menu.** The standard followed for many base main **menu** navigation options for fixed transactions is that navigation options launched from the Menu dropdown are configured with no context.

---

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_NAV\\_OPT](#).

## Navigation Option - Tree

This page contains a tree that shows how a navigation option is used. Select **Admin > System > Navigation Option** and navigate to the **Tree** tab to view this page.

### Description of Page

The tree shows every menu item, favorite link, and tree node that references the navigation option. This information is provided to make you aware of the ramifications of changing a navigation option.



# Database Tools

---

This section describes a variety of database tools that are supplied with the your product.

## Defining Table Options

The topics in this section describe the transaction that allows you to define metadata for the application's tables.

### Table - Main

Navigate using **Admin > Database > Table**.

#### Description of Page

**Table Name** is the identifier of this table.

**Description** contains a brief description of the table.

**System Table** defines if the table includes rows that are owned by the base product.

**Enable Referential Integrity** defines if the system performs referential integrity validation when rows in this table are deleted.

**Data Group ID** is used for internal purposes.

**Table Usage** is not in use at this time.

**Table Type** defines if the table is an **External Table**, a **View** or a physical **Table**.

**Date / Time Data Type** defines if the system shows times on this table in **Local Legal Time** or in **Local Standard Time**. Local Legal Time is the time as adjusted for daylight savings / summer time.

**Table Classification Type** specifies the category of data the table will hold. This is for information purposes only and is not used by any system processing. Valid values are **Admin System Table**, **Admin Non System Table**, **Master Table**, **Transaction Table**, and **Unclassified**.

**Table Volume Type** specifies the expected amount of data the table will hold. This is for information purposes only and is not used by any system processing. Valid values are **High Volume**, **Low Volume**, **Medium Volume**, and **Unclassified**. The volume of data in a particular table in the system may differ greatly from one implementation to another based on unique business requirements. The values populated for base product tables are set to volumes that are typical but may not be true for a given implementation. The value may be updated to reflect the situation for a given implementation.

**Audit Table** is the name of the table on which this table's audit logs are stored. Refer to [The Audit Trail File](#) for more information.

Use **Audit Program Type** to define if the audit program is written in **Java** or **Java (Converted)**, meaning it was converted into Java.

---

**NOTE:** **Java (Converted)** program types are not applicable to all products.

---

**Audit Program** is the name of the program that is executed to store an audit log. Refer to [Turn On Auditing For a Table](#) for more information.

---

**NOTE:** **View the source.** If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the [Java docs viewer](#).

---

**Upgrade** controls what happens to the rows in this table when the system is upgraded to a new release:



- **Keep** means that the rows on this table are not touched during an upgrade
- **Merge** means that the rows on this table are merged with rows owned by the base product
- **Refresh** means that the rows on this table are deleted and refreshed with rows owned by the base product.

**Data Conversion Role** controls if / how the table is used by the conversion tool:

- **Convert (Retain PK)** means that the table's rows are populated from the conversion schema and the prime key in the conversion schema is used when the rows are converted. A new key is not assigned by the system.
- **Convert (New PK)** means that the table's rows are populated from the conversion schema and the prime key is reassigned by the system during conversion.
- **Not Converted** means that the table's rows are not managed by the conversion tool.
- **View of Production** means that the conversion tool uses a view of the table in production when accessing the rows in the table. This is commonly used for administrative tables.

A **Language Table** is specified when fields containing descriptions are kept in a child table. The child table keeps a separate record for each language for which a description is translated.

**Enable Data Dictionary** defines if the table is to be included in the [Data Dictionary](#) application viewer.

A **Key Table** is specified when the prime-key is assigned by the system. This table holds the identity of the prime keys allocated to both live and archived rows.

**Type of Key** specifies how prime key values are generated when records are added to the table:

- **Other** means a foreign-system allocates the table's prime-key.
- **Sequential** means a sequence number is incremented whenever a record is added to the table. The next number in the sequence determines the key value.
- **System-generated** means a program generates a random key for the record when it is added. If the record's table is the child of another table, it may inherit a portion of the random number from its parent's key.
- **User-defined** means the user specifies the key when a record is added.

**Inherited Key Prefix Length** defines the number of most significant digits used from a parent record's primary key value to be used as the prefix for a child record's key value. This is only specified when the Type of Key is **System-generated** and the high-order values of the table's key is inherited from the parent table.

**Java Table Name.** This field is used to identify the entity/Java class name of the class that represents the table in the Java code. It should contain a short "camelCased" name to be used as the name of the entity within the system. It must also be a valid Java name, and must be unique across the system. This name is used as follows:

- As the short class name on all classes in the Java hierarchy for the class: the Impl class, the Gen, and the interface.
- In HQL queries, it is used to identify the hibernate entity being selected.

**Caching Regime** determines if the table's values should be cached when they are accessed by a batch process. The default value is **Not Cached**. You should select **Cached for Batch** if you know the values in the table will not change during the course of a batch job. For example, currency codes will not change during a batch process. Caching a table's values will reduce unnecessary SQL calls and improve performance.

**Key Validation** determines if and when keys are checked for uniqueness. The default value is **Always Check Uniqueness**. Select **Check Uniqueness Online Only** when the database constructs the keys in the table, such as in log tables. Select **Never Perform Uniqueness Checking** when you know that the database constructs the keys in the table and that users cannot add rows directly to the table, such as in log parameter tables. This will reduce unnecessary SQL calls and improve performance.

**Help URL** is the link to the user documentation that describes this table.

**Help Text** contains additional information about the table.

**Extract for Translation** is only visible in a development environment. It indicates whether or not the table includes strings that are eligible for product translation.

**Translation Context** is only visible in a development environment. It is used to provide information to a translator about the nature and purpose of rows in the table.

---

**NOTE: Changes to base owned tables.** Only the following attributes of tables that are owned by the product are modifiable: Audit Table, Audit Program Type, Audit Program, Caching Regime, Key Validation and Table Volume Type.

---

The grid contains an entry for every field on the table. Drilling down on the field takes you to the [Table Field](#) tab where you may modify certain attributes. The following fields may also be modified from the grid: **Description, Override Label, Audit Delete, Audit Insert** and **Audit Update**. Refer to the Table Field tab for descriptions of these fields.

## Table - Table Field

Navigate using **Admin > Database > Table** and click the **Table Field** tab.

### Description of Page

**Field Name** is the name of the field. It is followed by its **Java Field Name**.

**Field Help Text** displays the help text listed for this field on the Field page, if populated.

**Nullable, Required** and **Validate** are for internal use.

Turn on **Audit Delete** if an audit record should be stored for this field when a row is deleted. Refer to [How To Enable Auditing](#) for more information.

Turn on **Audit Insert** if an audit record should be stored for this field when a row is added. Refer to [How To Enable Auditing](#) for more information.

Turn on **Audit Update** if an audit record should be stored for this field when it is changed. Refer to [How To Enable Auditing](#) for more information.

The **Allow Customization** is only applicable if the table is an Admin System Table. It indicates fields that an implementation is allowed to change for a base owned record. Changes to the field value of one of these types of fields by an implementation are maintained when upgrading to a new version of the product.

**Standard Time Type** is only enabled if the Table indicates that the Date/Time Data Type is **Local Standard Time**. Each field that represent date/time should define if it uses **Logical Standard Time, Physical Standard Time** or uses a **Referenced Time Zone**.

**Sequence** is a unique sequence of this field with respect to other fields on the table.

The **Label** column is used to define a special label for this field when it relates to this table if it should be different from the field's label. Note that this only impacts the label on a fixed page user interface. Labels on portal based user interfaces do not use this information.

The **Override Label** is provided if an implementation wants to override the base-product label.

---

**NOTE:** If you want the Override Label to be shown in the [data dictionary](#), you must regenerate the data dictionary.

---

**Help Text** contains any additional information about the field with respect to its use on this table.

**Field Usage** is not used at this time.

**Extract for Translation** is only visible in a development environment. For tables marked to extract for translation, each translatable field on the table should indicate **Yes**.

**Translation Context** is only visible in a development environment. It is used to provide information to a translator about the nature and purpose of the data in this field on this table.

---

**NOTE: Changes to base owned table / fields.** Only the following attributes of table / fields that are owned by the product are modifiable: Audit Delete, Audit Insert, Audit Update, Override Label.

---

## Table - Constraints

Select **Admin > Database > Table** and navigate to the **Constraints** tab to view the constraints defined on the table.

### Description of Page

The fields on this page are protected as only the product development group may change them.

This page represents a collection of constraints defined for the table. A constraint is a field (or set of fields) that represents the unique identifier of a given record stored in the table or a field (or set of fields) that represents a given record's relationship to another record in the system.

**Constraint ID** is a unique identifier of the constraint.

**Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**)

**Constraint Type Flag** defines how the constraint is used in the system:

- **Primary Key** represents the field or set of fields that represent the unique identifier of a record stored in a table.
- **Logical Key** represents an alternate unique identifier of a record based on a different set of fields than the Primary key.
- **Foreign Key** represents a field or set of fields that specifies identifying and non-identifying relationships to other tables in the application. A foreign key constraint references the primary key constraint of another table.
- **Conditional Foreign Key** represents rare relationships between tables where a single field (or set of fields) may reference multiple primary key constraints of other tables within the application as a foreign key.

When **Enable Referential Integrity** is checked, the system validates the integrity of the constraint when a row in the table is modified.

**Referring Constraint Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

**Referring Constraint ID** is the **Primary Key** constraint of another table whose records are referenced by records stored in this table.

**Referring Constraint Table** displays the table on which the Referring Constraint ID is defined. You can use the adjacent go-to button to open the table.

**Additional Conditional SQL Text** is only specified when the constraint is a **Conditional Foreign Key**. The SQL represents the condition under which the foreign key represents a relationship to the referring constraint table.

---

**NOTE: Additional Conditional SQL Syntax.** When specifying additional conditional SQL text, all table names are prefixed with a pound (#) sign.

---

The Constraint Field grid at the bottom of the page is for maintaining the field or set of fields that make up this constraint.

**Field** is the name of the table's field that is a component of the constraint.

**Sequence** The rank of the field as a component of the constraint.

The Referring Constraint Field grid at the bottom of the page displays the field or set of fields that make up the **Primary key** constraint of the referring constraint.

**Field** is the name of the table's field that is a component of the referring constraint.

**Sequence** is the rank of the field as a component of the referring constraint.

## Table - Referred by Constraints

Select **Admin > Database > Table** and navigate to the **Referred By Constraints** tab to view the constraints defined on other tables that reference the **Primary Key** constraint of this table.

### Description of Page

This page is used to display the collection of constraints defined on other tables that reference the table.

**Referred By Constraint Id** is the unique identifier of the constraint defined on another table.

**Referred By Constraint Owner** indicates if this constraint is owned by the base package or by your implementation (**Customer Modification**).

**Prime Key Constraint Id** is the **Primary Key** constraint of the current table.

**Prime Key Owner** indicates if this prime key is owned by the base package or by your implementation (**Customer Modification**).

**Referred By Constraint Table** is the table on which Referred By Constraint Id is defined.

When **Enable Referential Integrity** is checked, the system validates the integrity of the constraint when a row in the table is modified.

The grid at the bottom of the page displays the **Field** and **Sequence** for the fields that make up the constraint defined on the other table.

## Defining Field Options

The topics in this section describe the transaction that can be used to view information about a field and to change the name of a field on the various pages in the system.

### Field - Main

Open this page using **Admin > Database > Field**.

#### Description of Page

**Field Name** uniquely identifies this field.

---

**CAUTION:** As described in [System Data Naming Convention](#) for most system data tables, the base product follows a specific naming convention. However, this is not true for the Field table. If you introduce new fields, you must prefix the field with **CM**. If you do not do this, there is a possibility that a future release of the application could introduce a new field with the name you allocated.

---

**Owner** indicates if this field is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a field.

**Base Field** indicates that the field inherits some of its definitions from another field.

**Data Type** indicates if the type of data the field will hold. Valid values are **Character**, **Character Large Object**, **Date**, **DateTime**, **Number**, **Time**, **Varchar2** and **XML Type**. This field is protected if the field refers to a Base Field.

**Ext Data Type** or Extended Data Type is used to further define the type of data for certain data types. Valid values are **Currency Source**, **Day of Month**, **Duration**, **Money**, **Month of Year**, **Flag** and **Switch**. This field is protected if the field refers to a Base Field.

**Precision** defines the length of the field. In the case of variable length fields, it is the maximum length possible. For number fields that include decimal values, the precision includes the decimal values. This field is protected if the field refers to a Base Field.

**Scale** is only applicable for number fields. It indicates the number of decimal places supported by the field. This field is protected if the field refers to a Base Field.

**Sign** is only applicable for numbers. It indicates if the data may contain positive or negative numbers.

**Value List** is only visible for products that had at one point included COBOL. It defines the copybook that includes the list of valid values for the field.

**Description** contains the label of the field. This is the label of the field that appears on the various pages on which the field is displayed. Note, the field's label can be overridden for a specific table by specifying an Override Label on the [table / field](#) information. However, this override is not used in portal based user interfaces. It is only applicable if the field is displayed on fixed page user interfaces.

**Java Field Name** is the reference to this field used in Java code.

**Override Label** is used if an implementation would like to override the label that appear on user interfaces in the system.

---

**CAUTION:** For fixed pages, if the field's label is overridden for a specific table, that override takes precedence. In this is the case the override on the [table / field](#) page should be used.

---

**Work Field** indicates that the field does not represent a database table column.

**Help Text** is used to provide field level embedded help to this field. If the field is displayed on a user interface that supports display of embedded help, this text may be displayed.

Use **Override Help Text** to override the existing embedded help text for this field.

**Extract for Translation** is only visible in a development environment. It indicates whether or not the field and its description should be included in an extract of translatable strings when doing a product translation. This flag may be set to “No” for work fields whose label is not visible to a user on any user interface.

**Translation Context** is only visible in a development environment. It is used to provide information to a translator about the use of the field's label so that an appropriate translation can be provided.

## Field - Tables Using Field

Select **Admin > Database > Field** and navigate to the **Tables Using Field** tab to view the tables that contain a field.

### Description of Page

The grid on this page contains the **Tables** that reference the **Field**. You can use the adjacent go to button to open the [Table Maintenance](#) transaction.

## Defining Maintenance Object Options

A maintenance object defines the configuration of a given “entity” in the system. It includes the definition of the tables that together capture the physical data for the entity. In addition, the maintenance object includes options that define important information related to the maintenance object that may be accessed for logic throughout the system. Several algorithm plug-in spots are also defined on the maintenance object, allowing for business rules that govern all records for this maintenance object.

Many maintenance objects in the system support the use of business objects to further define configuration and business rules for a given record. Refer to [The Big Picture of Business Objects](#) for more information.

## Maintenance Object - Main

Select **Admin > Database > Maintenance Object** to view information about a maintenance object.

### Description of Page

Most maintenance objects are provided with the base package. An implementation can introduce custom maintenance objects when needed. Most fields may not be changed if owned by the base package.

Enter a unique **Maintenance Object** name and **Description**. **Owner** indicates if this business object is owned by the base package or by your implementation (**Customer Modification**).

---

**IMPORTANT:** If you introduce a new maintenance object, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

**Service Name** is the name of the internal service associated with the maintenance object.

Click the **View XML** hyperlink to view the XML document associated with the maintenance object service in the [Service XML Viewer](#).

Click the **View MO** hyperlink to view the definition of the maintenance object in the [Maintenance Object Viewer](#).

The grid displays the following for each table defined under the maintenance object:

- **Table** is the name of a given table maintained as part of the maintenance object.
- **Table Role** defines the table's place in the maintenance object hierarchy. Only one **Primary** table may be specified within a maintenance object, but the maintenance object may contain many **Child** tables.
- **Parent Constraint ID** specifies the [constraint](#) used to link the table to its parent table within the maintenance object table hierarchy.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

## Maintenance Object - Options

Use this page to maintain a maintenance object's options. Open this page using **Admin > Database > Maintenance Object** and then navigate to the **Options** tab.

### Description of Page

The options grid allows you to configure the maintenance object to support extensible options.

Select the **Option Type** drop-down to define its **Value**. **Detailed Description** may display additional information on the option type.

Set the **Sequence** to **1** unless the option can have more than one value.

**Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

---

**NOTE: You can add new option types.** Your implementation may want to add additional maintenance option types. For example, your implementation may have plug-in driven logic that would benefit from a new type of option. To do that, add your new values to the customizable lookup field **MAINT\_OBJ\_OPT\_FLG**.

---

## Maintenance Object - Algorithms

Use this page to maintain a maintenance object's algorithms. Open this page using **Admin > Database > Maintenance Object** and then navigate to the **Algorithms** tab.

## Description of Page

The **Algorithms** grid contains algorithms that control important functions for instances of this maintenance object. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
- If the algorithm is implemented as a script, a link to the **Script** is provided. Refer to [Plug-in Scripts](#) for more information.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

The following table describes each **System Event**.

System Event	Optional / Required	Description
<b>Audit</b>	Optional	<p>Algorithms of this type are called to notify of any changes to the maintenance object's set of tables.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>Determine BO</b>	Optional	<p>Algorithm of this type is used to determine the Business Object associated with an instance of the maintenance object. It is necessary to plug in such an algorithm on a Maintenance Object to enable the business object rules functionality.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>ILM Eligibility</b>	Optional	<p>Algorithms of this type are used for maintenance objects that are enabled for <a href="#">Information Lifecycle Management</a>. They are used to review records that have reached the maximum retention days and evaluate if they are ready to be archived.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>Information</b>	Optional	<p>We use the term "Maintenance Object Information" to describe the basic information that appears throughout the system to describe an instance of the maintenance object. The data that appears in this information description is constructed using this algorithm.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>Revision Control</b>	Optional	<p>An algorithm of this type is used to enforce revision control rules when an object is added, changed or deleted. The maintenance object</p>

System Event	Optional / Required	Description
		<p>service calls the plug-in once before the object is processed and once more after applying all business object rules. This allows revision rules to take place in proper revision timings.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
Transition	Optional	<p>The system calls algorithms of this type upon each successful state transition of a business object as well as when it is first created. These are typically used to record the transition on the maintenance object's log.</p> <p>Note that most base maintenance objects are already shipped with an automatic logging of state transitions. In this case you may use these algorithms to override the base logging functionality with your own. Refer to <a href="#">State Transitions are Audited</a> for more information.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
Transition Error	Optional	<p>The system calls this type of algorithm when a state transition fails and the business object should be saved in its latest successful <a href="#">state</a>. The algorithm is responsible for logging the transition error somewhere, typically on the maintenance object's log.</p> <p>Notice that in this case, the caller does NOT get an error back but rather the call ends successfully and the exception is recorded somewhere, as per the plug-in logic.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>

**NOTE: You can inactivate algorithms on Maintenance Objects.** Your implementation may want to inactivate one or more algorithms plugged into the base maintenance object. To do that, go to the options grid on Maintenance Object - Options and add a new option, setting the option type to **Inactive Algorithm** and setting the option value to the algorithm code.

## Maintenance Object - Maintenance Object Tree

You can navigate to the **Maintenance Object Tree** to see an overview of the tables and table relationships associated with the maintenance objects.

### Description of Page

This page is dedicated to a [tree](#) that shows the maintenance object's tables as well as [business objects](#), if you have defined any. You can use this tree to both view high-level information about these objects and to transfer to the respective page in which an object is maintained.

## Defining Valid Values

The product provides several options for defining valid values for a column on a table:

- Lookup



- Extendable Lookup
- Control Table

The following provides more information about the functionality of each of the options available for defining valid values for a column.

## Lookup

The simplest mechanism for defining valid values for a column on a table is via the Lookup table. This is sometimes referred to as a “simple” lookup to distinguish it from an extendable lookup (described below). Using the lookup table, you can define valid values and their descriptions. When choosing a valid value that is defined by a lookup, a dropdown UI metaphor is used.

The following highlights functionality related to lookups:

- Lookups are associated with a [Field](#). The field is defined as a character data type with an extended data type of **Flag**. The field’s label serves as the description for the prompt to select the valid value.
- The lookup code is limited to four characters and must be all uppercase. If there is any functionality where a valid value in the application must match valid values in an external system, the lookup table may not be the appropriate choice.
- The lookup table does not support additional attributes to be defined for each value. This option is only appropriate when a simple code and description pair is needed.
- The product may also use Lookups to define valid values for functionality unrelated to a column on a table. For example, an algorithm plug-in spot may define an input parameter that supports one or more valid values. The plug-in spot may define the valid values using a lookup, allowing for a simple way to validate the value supplied when invoking the algorithm and to document the valid values.

---

**FASTPATH:** For more information, refer to [Defining Lookup Options](#).

---

## Extendable Lookup

The extendable lookup provides a way of defining valid values for a column with additional capabilities that are not supported using the Lookup table. When choosing a valid value that is defined by an extendable lookup, a dropdown UI metaphor is used.

The following highlights functionality related to extendable lookups:

- Each Extendable Lookups is defined using a business object.
- A field should be defined for the extendable lookup code. The field defines the label for the lookup code and defines the size of the lookup code. The size is determined based on the business use case. In addition, there are standard fields included in all extendable lookups, including a description, detailed description and an override description (so that implementations can override the description of base delivered values).
- The extendable lookup may define additional information for each value if warranted by the business requirement. See [Additional Attributes](#) for technical information about additional attributes.

---

**FASTPATH:** For more information, refer to [Defining Extendable Lookups](#).

---

## Control Table

There may be scenarios where a list of valid values warrants a standalone maintenance object, which is considered an administrative or control table object. When choosing a valid value that is defined by a control, either a dropdown UI metaphor or a search metaphor is used, depending on how it has been designed.

The following points highlight some reasons why this option may be chosen:

- The records require a lifecycle such that BO status is warranted.

- The additional attributes are sophisticated enough that they warrant their own column definition rather than relying on using CLOB or flattened characteristic. For example, if a list of information needs to be captured with several attributes in the list and the information in the list needs to be searchable.

In this situation, if a product has provided a control table for this type of functionality, it will be documented fully in the appropriate functional area. If an implementation determines that a custom control table is warranted, all the standard functionality for a maintenance object is required: database tables, maintenance object metadata, appropriate Java maintenance classes, portals, zones, etc. Refer to the Software Development Kit for more information. No further information is provided in this section for this option.

## Defining Lookup Options

Lookup fields may be used to define valid values for a column in a table or for other types of values like parameters to an algorithm.

---

**FASTPATH:** Refer to [Defining Valid Values](#) for some background information.

---

The base product provides many different lookup fields and their values as part of the product. The following points highlight some functionality related to base-package lookups.

- Fields that are owned by the product will typically provide base lookup values. Implementations are not permitted to remove base delivered lookup values. Implementations may be able to add custom values to base owned lookups. This is controlled with the Custom switch on lookup.
  - When the custom switch is unchecked, it means that there is functionality controlled by the base values and an implementation may not extend or customize this functionality. An example of this type of lookup is the Data Type field on the [Field](#) table. The system supports a distinct list of data types and an implementation may not add additional values.
  - When the custom switch is checked, it means that there is base functionality supplied for the base values but that an implementation can extend the functionality by supplying their own values. An example of this type of lookup is the Access Mode on [Application Service](#). The product provides many values for the access mode lookup, representing various actions a user may perform. Implementations may add their own values to this lookup. Documentation should indicate when functionality may be extended and should highlight the lookup value that can be extended.

---

**CAUTION:** Important! If you introduce new lookup values, you must prefix the lookup value code with **X** or **Y**. If you do not do this, there is a possibility that a future release of the application could introduce a new lookup value with the name you allocated.

---

- There may be some scenarios where the product supplies a base field and base lookup field with no base lookup values supplied. This occurs when the product doesn't have any base functionality driven by the lookup values. Typically this type of lookup is for information or categorization purposes. The configuration guide for the functional area associated with the lookup should include a configuration step regarding defining values for this type of lookup.
- The description of base delivered values may be overridden by an implementation.

An implementation may also identify the need for defining a new lookup field with its values.

## Lookup - Main

Select **Admin > Database > Lookup** to maintain lookup values.

### Description of Page

**Field Name** is the name of the field whose lookup values are maintained in the grid. If you need to add a new lookup field, you must first add the lookup field here, then navigate to the [Field](#) page to create a field with a data type of **Character** and an extended data type of **Flag**.

**Owner** indicates if this lookup field is owned by the base package or by your implementation (**Customer Modification**). This information is display-only.

**Custom** switch is used to indicate whether you are allowed to add valid values for a lookup field whose owner is not **Customer Modification**.

- If this switch is turned on, you may add new values to the grid for system owned lookup fields.
- If this switch is turned off, you may not add, remove or change any of the values for system owned lookup fields, with the exception of the override description.

This field is always protected for system owned lookup fields because you may not change a field from customizable to non-customizable (or vice versa).

**Java Field Name** indicates the name of the field as it is referenced in Java code.

The grid contains the lookup values for a specific field. The following fields are visible:

**Field Value** is the unique identifier of the lookup value. If you add a new value, it must begin with an **X** or **Y** (in order to allow future upgrades to differentiate between your implementation-specific values and base-package values).

**Description** is the name of the lookup value that appears on the various transactions in the system

**Java Value Name** indicates the unique identifier of the lookup value as it is referenced in Java code.

**Status** indicates if the value is **Active** or **Inactive**. The system does not allow **Inactive** values to be used (the reason we allow **Inactive** values is to support historical data that references a value that is no longer valid).

**Detailed Description** is the detailed description for a lookup value, which is provided in certain cases.

**Override Description** is provided if your implementation wishes to override the description of the value provided by the product.

---

**NOTE:** If you wish the override descriptions of your lookup values to appear in the application viewer, you must [regenerate](#) the data dictionary application viewer background process.

---

**Owner** indicates if this lookup value is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add lookup values to a field. This information is display-only.

## Defining Extendable Lookups

Extendable lookups are a way of defining valid values that are more sophisticated than simple lookups.

---

**FASTPATH:** Refer to [Defining Valid Values](#) for some background information.

---

The base product provides extendable lookups as part of the product. The following points highlight some functionality related to base-package extendable lookups.

- The base product may supply base extendable lookup values. Implementations are not permitted to remove base delivered extendable lookup values. It is also possible that implementations may be able to add custom values to base owned lookups. If an implementation is not permitted to add lookup values to the base extendable lookup, the extendable lookup's business object will include validation to prevent this. There is no equivalent of the Custom switch that is on the [lookup](#) field.
- There may be some scenarios where the product supplies a base extendable lookup with no base lookup values supplied. This occurs when the product doesn't have any base functionality driven by the extendable lookup values. The configuration guide for the functional area associated with the extendable lookup should include a configuration step regarding defining values for this type of extendable lookup.
- The description of base delivered values may be overridden by an implementation.

Open this page using **Admin > Database > Extendable Lookup**.

You are brought to the **Extendable Lookup Query** where you need to search for the extendable lookup object (i.e., its business object).

Once you have found the appropriate extendable lookup, select the value and you are brought to a standard [All-in-One](#) portal that lists the existing lookup values for the extendable lookup. The standard actions for an All-in-One portal are available here.

## Defining Additional Attributes

The product provides a few different ways to define additional values for an extendable lookup. Some of the methods are only relevant for base delivered lookup values as they may impact whether or not an implementation can update the values.

The following table highlights the options available and some summary information about what the option provides.

Option	Brief Description	Extendable Lookup Value Searchable by this Attribute?	Base Delivered Value Modifiable?
Element mapped to BO_DATA_AREA	The element is mapped to a CLOB field that allows for base delivered values to be modified.	No	Yes
Element mapped to BASE_BO_DATA_AREA	The element is mapped to a CLOB field that does not allow for base delivered values to be modified.	No	No
Flattened characteristic	The element is defined using the flattened characteristic mechanism.	Yes	No

The following points highlight information from the table above:

- The decision of defining an additional attribute using a CLOB mapping or a flattened characteristic will depend on whether the functionality expects that the lookup will be known when the attribute is needed (in which case a CLOB mapping is appropriate) or if the functionality expects to determine the extendable lookup based on the attribute (in which case, a flattened characteristic is appropriate).
- When the base product defines an extendable lookup with additional attributes and intends to provide base extendable lookup values, it needs to determine whether or not implementations may update the additional attribute or not.
  - If no and the value is mapped to a CLOB, it will map the value to the BASE\_BO\_DATA\_AREA column. This means that implementations will receive an owner mismatch error when attempting to change the value. In addition, upgrading to a new release will replace the value with the base value.
  - If yes and the value is mapped to a CLOB, it will map the value to the BO\_DATA\_AREA column. This means that implementations will be able to change the value for a base owned record. In addition, upgrading to a new release will not make any changes to the value.
  - For values mapped to a characteristic, the product does not support an implementation changing the value of a base delivered record. If the product would like to support an implementation overriding this type of value, the business object will need to be designed with a corresponding “override” element (also a flattened characteristic), similar to how the product supplies an Override Description field to support an implementation overriding the base product delivered description for a base value. This element will not be delivered with any value and will allow an implementation to populate that value.

**NOTE:** Note that in this situation, the product functionality that uses this value must cater for the override value.

- All of this detail is only relevant for base provided extendable lookup values. If an implementation adds custom values for a base supplied extendable lookup, all the additional attributes may be populated as appropriate.

- If an implementation defines a custom extendable lookup business object and wants to define an additional attribute using a CLOB, it doesn't matter which CLOB column is used. Both BO\_DATA\_AREA and BASE\_BO\_DATA\_AREA provide the same functionality for custom business objects.

## The Big Picture Of Audit Trails

The topics in this section describe auditing, enabling auditing for fields, and auditing queries that you can use to view audit records.

### Captured Information

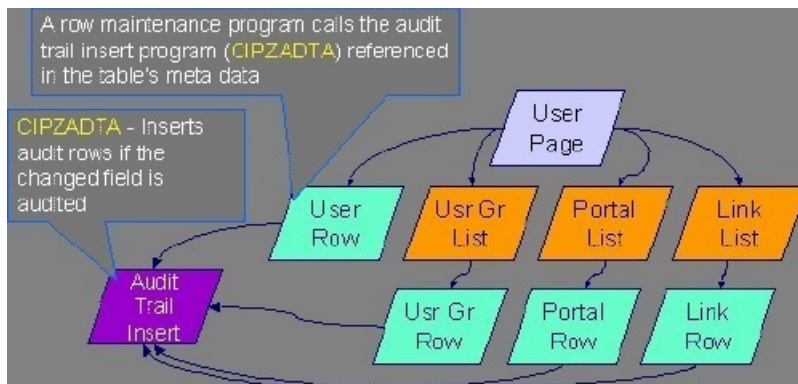
When auditing is enabled for a field, the following information is recorded when the field is changed, added and/or deleted (depending on the actions that you are auditing for that field):

- User ID
- Date and time
- Table name
- Row's prime key value
- Field name
- Before image (blank when a row is added)
- After image (blank when a row is deleted)
- Row action (add, change, delete)

### How Auditing Works

You enable auditing on a table in the table's meta-data by specifying the name of the table in which to insert the audit information (the audit table) and the name of the program responsible for inserting the data (the audit trail insert program). Then you define the fields you want to audit by turning on each field's audit switch in the table's field meta-data. You can audit fields for delete, insert and update actions.

Once auditing is enabled for fields in a table, the respective row maintenance program for the table assembles the list of changed fields and calls the audit trail insert program ( **CIPZADTA** is supplied with the base package). If any of the changed fields are marked for audit, **CIPZADTA** inserts audit rows into the audit table ( **CI\_AUDIT** is the audit table supplied with the base package).



**NOTE: Customizing Audit Information.** You may want to maintain audit information other than what is described in [Captured Information](#) or you may want to maintain it in a different format. For example, you may want to maintain audit

information for an entire row instead of a field. If so, your implementation team can use **CIPZADTA** and **CI\_AUDIT** as examples when creating your own audit trail insert program and audit table structures.

---

## The Audit Trail File

Audit log records are inserted in the audit tables you define. The base product contains a single such table (called **CI\_AUDIT**). However, the audit insert program (**CIPZADTA**) is designed to allow you to use multiple audit tables.

If you want to segregate audit information into multiple tables, you must create these tables. Use the following guidelines when creating new audit tables (that use the **CIPZADTA** audit insert program):

- The new audit tables must look identical to the base table (**CI\_AUDIT**).
- The new tables must be prefixed with **CM** (e.g., **CM\_AUDIT\_A**, **CM\_AUDIT\_B**, etc.).
- The name of the new table must be referenced on the various tables whose changes should be logged in the new table.

---

**NOTE: Interesting fact.** It's important to note if you use your own tables (as opposed to using the base package table called **CI\_AUDIT**), the SQL used to insert and access audit trail records (in **CIPZADTA**) is dynamic. Otherwise, if the base package's table is used, the SQL is static.

---

## How To Enable Auditing

Enabling audits is a two-step process:

- First, you must turn on auditing for a table by specifying an audit table and an audit trail insert program.
- Second, you must specify the fields and actions to be audited for the table.

The following topics describe this process.

### Turn On Auditing For a Table

In order to tell the system which fields to audit, you must know the name of the table on which the field is located. You must specify the audit table and the audit trail insert program for a table in the table's meta-data.

---

**NOTE:** Most of the system's table names are fairly intuitive. For example, the user table is called **SC\_USER**, the navigation option table is called **CL\_NAV\_OPT**, etc. If you cannot find the table using the search facility on the [Table Maintenance](#) page, try using the [Data Dictionary](#). If you still cannot find the name of the table, please contact customer support.

---

To enable auditing for a table:

- Navigate to the [Table maintenance](#) page and find the table associated with the field(s) for which you want to capture audit information.
- Specify the name of the **Audit Table**.

---

**NOTE: Specifying the Audit Table.** You can use the audit table that comes supplied with the base package (**CI\_AUDIT**) to audit multiple tables and fields. All the audit logs are combined in a single table (**CI\_AUDIT**). However, you can also have a separate audit table for each audited table. Refer to [The Audit Trail File](#) for more information.

---

- Specify the name of the **Audit Program** (**CIPZADTA** is the default audit program supplied with the base package).

---

**CAUTION:** By default, none of a table's fields are marked for audit. Even though you have enabled auditing for a table, you must still specify the fields and actions on those fields to be audited (see below).

---

## Specify The Fields and Actions To Be Audited

The system only audits actions (insert, update and delete) made to fields that you want audited.

To specify the fields and actions to be audited:

- Navigate to the [Table - Table Field maintenance](#) page for a table on which you have enabled auditing.
- For each field you want to audit, specify the actions you want to audit by turning on the **Audit Delete**, **Audit Insert** and **Audit Update** switches as appropriate.

---

**NOTE: Note.** You can also turn on the audit switches using the Field grid at the bottom of the [Table maintenance page](#).

---

**CAUTION:** Audit Program Caching! The audit program from the table meta-data is read into a program cache on the application server whenever the date changes or when the server starts. If you implement new auditing on a table, your audit trail does not become effective until this program cache is reloaded. In other words, new audits on tables where the audit program was not previously specified do not become effective until the next day (or the next restart of the application server). However, if you change the fields to be audited for a table where the audit program is already in the cache, your changes are effective immediately.

---

## Audit Queries

There are two queries that can be used to access the audit information.

### Audit Query by User

This transaction is used to view changes made by a user that are stored on a given [Audit Trail File](#).

---

**CAUTION:** The system only audits changes that you've told it to audit. Refer to [The Big Picture Of Audit Trails](#) for more information.

---

Navigate to this page by selecting **Admin > Audit Query > Audit Query By User**.

#### Description of Page

To use this transaction:

- Enter the **User ID** of the user whose changes you wish to view.
- Enter the name of the table on which the audit trail information is stored in **Audit Table**. Refer to [The Audit Trail File](#) for more information about this field.

---

**NOTE: Default Note.** If only one audit table is used to store audit trail information, that table is defaulted.

---

- Specify a date and time range in **Created between** to restrict the records that result from the query.

---

**NOTE: Default Note.** The current date is defaulted.

---

- Click the search button to display all changes recorded on a specific audit table associated with a given user.

Information on this query is initially displayed in reverse chronological order.



The following information is displayed in the grid:

- **Row Creation Date** is the date and time that the change was made.
- **Audited Table Name** contains the name of the table whose contents were changed.
- **Primary Key** is the prime key of the row in the **Audited Table** whose contents were changed.
- **Audited Field Name** is the name of the field that was changed.
- **Audit Action** indicates whether the row action was **Add**, **Change** or **Delete**.
- **Field Value Before Update** contains the content of the field before the change. This column is blank if information was **Added**.
- **Field Value After Update** contains the content of the field after the change. This column is blank if information was **Deleted**.

## Audit Query by Table / Field / Key

This transaction is used to view audited changes made to a given table.

---

**CAUTION:** The system only audits changes that you've told it to audit. Refer to [The Big Picture Of Audit Trails](#) for more information.

---

**NOTE:** Most of the system's table names are fairly intuitive. For example, the user table is called [SC\\_USER](#), the navigation option table is called [CI\\_NAV\\_OPT](#), etc. If you cannot find the table using the search facility on the [Table Maintenance](#) page, try using the [Data Dictionary](#). If you still cannot find the name of the table, please contact customer support.

---

This transaction can be used in several different ways:

- You can view all audited changes to a table. To do this, enter the **Audited Table Name** and leave the other input fields blank.
- You can view all audited changes to a given row in a table (e.g., all changes made to a given user). To do this, enter the **Audited Table Name** and row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**).
- You can view all audited changes to a given field in a table (e.g., all changes made to all customers' rates). To do this, enter the **Audited Table Name** and the **Audited Field Name**.
- You can view all audited changes to a given field on a specific row. To do this, enter the **Audited Table Name**, the **Audited Field Name**, and row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**).

Navigate to this page by selecting **Admin > Audit Query > Audit Query By Table/Field/Key**.

### Description of Page

To use this transaction:

- Enter the name of the table whose changes you wish to view in **Audited Table Name**.
- If you wish to restrict the audit trail to changes made to a specific field, enter the **Audited Field Name**.
- If you wish to restrict the audit trail to changes made to a given row, enter the row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**). These fields are dynamic based on the **Audited Table Name**.
- Specify a date and time range in **Created between** to restrict the records that result from the query.

---

**NOTE:** The current date is defaulted.

---

- Click the search button to display all changes made to this data.

Information on this query is initially displayed in reverse chronological order by field.



The following information is displayed in the grid:

- **Create Date/Time** is the date / time that the change was made.
- **User Name** is the name of the person who changed the information.
- **Primary Key** is the prime key of the row in the **Audited Table** whose contents were changed.
- **Audited Field Name** is the name of the field that was changed.
- **Audit Action** indicates whether the row action was **Add**, **Change** or **Delete**.
- **Value Before Update** contains the content of the field before the change. This column is blank if information was **Added**.
- **Value After Update** contains the content of the field after the change. This column is blank if information was **Deleted**.

## Bundling

The topics in this section describe the bundling features in the application.

### About Bundling

Bundling is the process of grouping entities for export or import from one environment to another.

For example, you might export a set of business objects and service scripts from a development environment and import them into a QA environment for testing. The group of entities is referred to as a bundle. You create export bundles in the source environment; you create import bundles in the target environment.

Working with bundles involves the following tasks:

- Configuring entities for bundling if they are not preconfigured
- Creating an export bundle, which contains a list of entities to be exported from the source environment
- Creating an import bundle to import those entities to the target environment
- Applying the import bundle, which adds or updates the bundled entities to the target environment

### Sequencing of Objects in a Bundle

Bundle entities are added or updated to the target environment in the sequence defined in the bundle

Typically, the sequence of entities does not matter. However, sequence is important in the following situations:

- Entities that are referenced as foreign keys should be at the top of the sequence, before the entities that reference them. Specify zones last, as they typically contain numerous foreign key references.
- When importing a business object, specify the business object first, then its plug-in scripts, then the algorithms that reference the scripts, and then the algorithm types that reference the algorithms.
- When importing a portal and its zones, specify the portal first and then its zones.
- When importing a multi-query zone, specify the referenced zones first and then the multi-query zone.
- Always specify algorithm types before algorithms.

You can specify the sequence when you define the export bundle or when you import the bundle to the target environment.

## Recursive Key References

Recursive foreign keys result when one object has a foreign key reference to another object that in turn has a foreign key reference to the first object.

For example, a zone has foreign keys to its portals, which have foreign keys to their zones. If the objects you want to bundle have recursive relationships, you must create a 'bundling add' business object that has only the minimal number of elements needed to add the entity. A bundling add business object for a zone contains only the zone code and description, with no references to its portals. In the same way, a bundling add business object for a portal defines only its code and description.

When you apply the bundle, the system initially adds the maintenance object based on the elements defined in the bundling add business object. Before committing the bundle, the system updates the maintenance object with the complete set of elements based on its physical business object.

## Owner Flags on Bundled Entities

The owner flag of the entities in an import bundle must match the owner flag of the target environment.

If you need to import objects that your source environment does not own, you must replace the owner flag in the import bundle with the owner flag of the target environment.

## Configuring Maintenance Objects for Bundling

All base package meta-data objects are pre-configured to support bundling. All other objects must be manually configured.

If a base package maintenance object is pre-configured for bundling, the **Eligible For Bundling** option will be set to "Y" on the Options tab for the maintenance object.

To configure other objects for bundling, review the configuration tasks below and complete all those that apply:

Complete this configuration task...	for...
Make maintenance objects eligible for bundling	All objects to be included in the bundle
Add a foreign key reference	All objects to be included in the bundle
Create a physical business object	All objects to be included in the bundle
Create a bundling add business object	Objects with recursive foreign key references
Add the Current Bundle zone	All objects, if you want the Current Bundle zone to appear on the maintenance object's dashboard. This is not required by the bundling process.
Create a custom Entity Search zone and add it to the Bundle Export portal	All objects, if you want them to be searchable in the Bundle Export portal. This is not required by the bundling process.

## Making Maintenance Objects Eligible for Bundling

The "Eligible For Bundling" maintenance object option must be set to "Y" for all bundled objects.

To make maintenance objects eligible for bundling:

1. Select **Admin > Maintenance Object > Search** and search for the maintenance object.
2. On the Options tab, add a new option with the type **Eligible For Bundling**.
3. Set the value to "Y" and click **Save**.

## Adding a Foreign Key Reference

Each maintenance object in a bundle must have a foreign key reference. Bundling zones use the foreign key reference to display the standard information string for the maintenance object.

To add a foreign key reference to the maintenance object:

1. Select **Admin > FK Reference > Add** and set up a foreign key reference for the primary table of the maintenance object.
2. Select **Admin > Maintenance Object > Search** and search for the maintenance object.
3. On the **Option** tab, add a new option with the type **Foreign Key Reference**. The value is the name of the foreign key reference you just created.

## Creating a Physical Business Object

Each maintenance object in a bundle must have a physical business object. The physical business object's schema represents the complete physical structure of the maintenance object, and includes elements for all fields in the maintenance object's tables. The bundling process uses this schema to generate the XML for the import bundle.

To create a physical business object for the maintenance object:

1. Select **Admin > System > Business Object > Add** and specify the maintenance object.
2. Click **Generate** in the **BO Schema** dashboard zone to generate a schema that looks like the physical structure of the maintenance object.
3. Save the physical business object.
4. Select **Admin > Maintenance Object > Search** and search for the maintenance object.
5. On the **Option** tab, add a new option with the type **Physical Business Object**. The value is the name of the physical business you just created.

## Creating a Bundling Add Business Object

If the objects to be bundled have recursive foreign key references, you must create a bundling add business object to avoid problems with referential integrity.

To create a bundling add business object:

1. Select **Admin > System > Business Object > Add** and specify the maintenance object.
2. Click **Generate** in the **BO Schema** dashboard zone to generate a schema that looks like the physical structure of the maintenance object.
3. Remove all elements that are not essential. Typically, only a code and description are required.
4. Save the physical business object.
5. Select **Admin > Maintenance Object > Search** and search for the maintenance object you want to bundle.
6. On the **Option** tab, add a new option with the type **Bundling Add BO**. The value is the name of the bundling add business object you just created.

## Adding the Current Bundle Zone

If you want the Current Bundle zone to appear on the maintenance object's dashboard, you must add the Current Bundle zone as a context-sensitive zone for the maintenance object.

To add the Current Bundle zone to the maintenance object:

1. Select **Admin > System > Context Sensitive Zone** and search for the navigation key for the maintenance object.
2. Add the Current Bundle zone, F1-BNDLCTXT, to that navigation key.

## Adding a Customized Entity Search Query Zone to the Bundle Export Portal

If you want the maintenance object to be searchable in the Bundle Export portal, you must first create an entity-specific query zone to search for the maintenance object. Then you must create a customized entity search zone that references this new query zone. Finally, you must add the customized entity search zone to the Bundle Export portal.

To make the maintenance object searchable:

1. Create an entity-specific query zone to search for the maintenance object:
  - a) Select **Admin > System > Zone > Search** and search for one of the base query zones, such as the Algorithm Search zone, F1-BNALGS.
  - b) Click the **Duplicate** button in the page actions toolbar.
  - c) Enter a name for the new zone.
  - d) Click **Save**.
  - e) Locate the **User Filter** parameter in the parameter list. Add SQL to search for the maintenance object(s) you want to appear in the zone.
  - f) Save the query zone.
2. Create a customized entity search zone:

This step only needs to be done once. If you already have a customized search zone in the Bundle Export portal go to step 3

  - a) Select **Admin > System > Zone > Search** and search for the F1-BNDLENTQ Entity Search zone.
  - b) Duplicate this zone (as described above).
  - c) Remove any references to base query zones.
3. Add the new entity-specific query zone to the customized entity search zone:
  - a) Locate the customized entity search zone for your Bundle Export portal. This is the zone created in Step 2.
  - b) Locate the Query Zone parameter in the parameter list. Add the name of the query zone you created in Step 1.
  - c) Save the entity search zone.
4. Add the customized entity search zone to the Bundle Export portal:

This step needs to be done only once.

  - a) Select **Admin > System > Portal > Search** and search for the Bundle Export portal, F1BNDLEM.
  - b) In the zone list, add the entity search zone you created in Step 2. (Add the new zone after the base entity search zone).
  - c) Save the portal.

## Working with Bundles

Use the Bundle Export portal to create an export bundle. The export bundle contains a list of entities to be exported from the source environment. When you are ready to import the objects, use the Bundle Import portal to import the objects to the target environment.

### Creating Export Bundles

An export bundle contains a list of entities that can be imported into another environment.

To create an export bundle:

1. Log on to the source environment from which objects will be exported.
2. Select **Admin > System > Bundle Export > Add**.
3. Complete the fields in the Main section to define the bundle's basic properties.

---

**NOTE:** You can use the Entities section to add bundle entities now, or save the bundle and then add entities as described in step 5.

---

4. Click **Save** to exit the Edit dialog. The export bundle status is set to Pending.
5. While an export bundle is in Pending state, use any of the following methods to add entities to the bundle:
  - a) Use the **Entity Search** zone on the Bundle Export portal to search for entities and add them to the bundle. If an entity is already in the bundle, you can remove it.
  - b) To import entities from a .CSV file, click **Edit** on the Bundle Export portal, and then click **CSV File to Upload**. Specify the file name and location of the .CSV file containing the list of entities. Click **Submit** to upload the file, and then click **Save** to save the changes.
  - c) Use the **Current Bundle** zone in the dashboard of the entity you want to add. (All entities that are configured to support bundling display a Current Bundle zone). This zone displays a list of all pending export bundles to which you can add the entity.
  - d) When you check an entity into revision control, specify the export bundle on the **Revision Info** dialog.
6. When you have added all entities, click **Bundle** in the Bundle Actions zone on the Bundle Export portal. The export bundle state is set to Bundled and the Bundle Details zone displays the XML representation of every entity in the bundle.

---

**NOTE:** The owner flags of the entities in the bundle must match the owner flag of the bundle itself. If the owner flags do not match, the system displays a warning message. Click **OK** to continue or **Cancel** to abort the bundle. If you click OK, you will need to resolve the owner flag discrepancy before you import the bundle to the target environment.

---

7. Copy the XML from the **Bundle Detail** zone to the clipboard (or to a text file). You can now create an import bundle and apply it to the target environment.

---

**NOTE:** If you need to make additional changes to the bundle, you must change the bundle state by selecting the **Back to Pending** button in the **Bundle Actions** zone.

---

## Creating and Applying Import Bundles

Import bundles define a group of entities to be added or updated in the target environment.

Before you create an import bundle, you must have already created an export bundle, added entities, and set the bundle's state to **Bundled**.

To create an import bundle and apply it to the target environment:

1. If you have not already copied the XML from the export bundle, do so now:
  - a) Select **Admin > System > Bundle Export** and search for the bundle.
  - b) Copy the XML from the **Bundle Detail** zone to the clipboard (or to a text file).
2. Log on to the target environment.
3. Select **Admin > System > Bundle Import > Add**.
4. In the **Bundle Actions** zone, click **Edit XML**.
5. Paste the contents of the clipboard (or text file if you created one) into the **Bundle Detail** zone.
6. Make any necessary changes to the XML and click **Save**. The status of the import bundle is set to **Pending**.

---

**NOTE:** Use caution when editing the XML to avoid validation errors.

---

7. To remove entities from the import bundle or change their sequence, click **Edit**. Enter your changes and click **Save** to exit the Edit dialog.
8. When you are ready to apply the bundle, click **Apply**. The import bundle state is set to **Applied** and the entities are added or updated in the target environment.

## Editing Export Bundles

You can add or remove entities from an export bundle when it is in **Pending** state. You can also change the sequence of entities.

To edit to an export bundle that has already been bundled, you must change the bundle state by selecting the **Back to Pending** button on the Bundle Export portal.

To edit a pending export bundle:

1. Open the bundle in edit mode.
2. Click **Edit** on the Export Bundle portal.
3. Make any necessary changes on the edit dialog and then click **Save**.

## Editing Import Bundles

You can remove entities from an import when it is in **Pending** state. You can also change the sequence of entities and edit the generated XML.

To edit a pending import bundle:

1. Open the bundle in edit mode.
2. To edit the XML snapshot, click **Edit XML**. Edit the XML code as needed, then click **Save**.

---

**NOTE:** Use caution when editing the XML to avoid validation errors.

---

3. To remove entities or change their sequence, click **Edit**. Make any necessary changes and click **Save**.

## Revision Control

The topics in this section describe the revision control features in the application.

### About Revision Control

Revision control is a tool provided for the development phase of a project to allow a user to check out an object that is being worked on. In addition, it captures the version of the maintenance object when users check in an update, maintaining a history of the changes to the object.

If revision control is enabled for an object you must check out the object to change it. While the object is checked out no one else can work on it. You can revert all changes made since checking out an object, reinstate an older version of an object, recover a deleted object, and force a check in of an object if someone else has it checked out.

---

**NOTE:** Revision control does not keep your work separate from the environment. Because the metadata for maintenance objects is in the central database, any changes you make to an object while it is checked out will be visible to others and may impact their work.

---

Many of the maintenance objects used as configuration tools are already configured for revision control, but it is turned off by default. For example, business objects, algorithms, data areas, UI maps, and scripts are pre-configured for revision control.

### Turning On Revision Control

Revision control is turned off by default for maintenance objects that are configured for revision control.

To turn on revision control:

1. Add the base package **Checked Out** zone to the Dashboard portal.
  - a) Select **Admin > System > Portal**.
  - b) Search for the portal **CI\_DASHBOARD**.
  - c) In the zone list for the **Dashboard** portal, add the zone **F1-USRCHKOUT**.
2. Set up application security.

For users to have access to revision control, they must belong to a user group that has access to the application service **F1-OBJREVBOAS**.
3. Add the revision control algorithm to the maintenance object that you want to have revision control.

This step must be repeated for each maintenance object that you want to have revision control.

  - a) Select **Admin > System > Maintenance Object** and search for the maintenance object that you want to have revision control.
  - b) On the **Algorithms** tab of the maintenance object, add the revision control algorithm **F1-REVCTL**.

## Configuring Maintenance Objects for Revision Control

Most configuration tool maintenance objects are pre-configured for revision control. You can configure other maintenance objects for revision control, as well.

To configure other objects for revision control:

### 1. Create a physical business object for the maintenance object.

A physical business object is one that has a schema with elements for all of the fields for the tables in the maintenance object. Follow these steps to create a physical business object:

- a) Select **Admin > System > Business Object > Add** and specify the maintenance object.
- b) Use the **BO Schema** dashboard zone to generate a schema that looks like the physical structure of the maintenance object.
- c) Save the physical business object.
- d) Select **Admin > Database > Maintenance Object > Search** and search for the maintenance object for which you want to enable revision control.
- e) On the **Options** tab of the maintenance object add a new option with the type **Physical Business Object**. The value is the name of the physical business object that you just created.

### 2. Add a foreign key reference to the maintenance object.

The revision control zones will display the standard information string for the object based on the foreign key reference. Follow these steps to create a foreign key reference:

- a) Select **Admin > Database > FK Reference > Add** and set up a foreign key reference for the primary table of the maintenance object.
- b) Select **Admin > Database > Maintenance Object > Search** and search for the maintenance object.
- c) On the **Options** tab of the maintenance object, add a new option with the type **Foreign Key Reference**. The value is the name of the foreign key reference that you just created.

### 3. Add the **Revision Control** zone to the maintenance object.

- a) Select **Admin > System > Context Sensitive Zone** and search for the navigation key for the maintenance object.
- b) Add the Revision Control zone, F1-OBJREVCTL, to that navigation key.

### 4. Add the revision control algorithm to the maintenance object.

- a) Select **Admin > Database > Maintenance Object** and search for the maintenance object that you want to have revision control.
- b) On the **Algorithms** tab of the maintenance object, add the revision control algorithm F1-REVCTL.

## Working with the Revision Control Zones

You use two zones in the dashboard to work with revision controlled objects when revision control is turned on.

The **Revision Control** zone gives you several options for managing the revision of the currently displayed object. This zone also shows when the object was last revised and by whom. This information is linked to the **Revision Control Search** portal which lists all of the versions of the object.

Using the Revision Control zone you can:

- Check out an object in order to change it.
- Check in an object so others will be able to work on it.



- Revert the object back to where it was at the last checkout.
- Force a check in of an object that is checked out by someone else. You need special access rights to force a check in.
- Delete an object.

The **Checked Out** zone lists all of the objects that you currently have checked out. Clicking on an object listed in this zone will take you to the page for that object. The zone is collapsed if you have no objects checked out.

See [Revision Control Search](#) for more information about Check In, Force Check In, and Check Out one or more records simultaneously.

## Checking Out an Object

You must check out a revision controlled object in order to change it.

An object must have revision control turned on before you can check it out.

---

**NOTE:** When you first create or update an object a dialog box informs you that the object is under revision control. You can select **OK** to check out the object and save your changes, or **Cancel** to stop the update.

---

1. Go to the object that you want to work on.
2. Select **Check Out** in the **Revision Control** dashboard zone.

## Checking In an Object

You must check in a revision controlled object in order to create a new version of it. Checking in an object also allows others to check it out.

1. Select a link in the **Checked Out** dashboard zone to go to the object that you want to check in.
2. Select **Check In** in the **Revision Control** dashboard zone.
3. Provide details about the version:
  - In the **External References** field state the bug number, enhancement number, or a reason for the revision.
  - In the **Detailed Description** field provide additional details regarding the revision.
  - In the **Keep Checked Out** box specify if you want to keep the object checked out. If you keep the object checked out then your revision is a new version that you can restore later.
  - In the **Add To Bundle** box specify if the object belongs to a bundle.
4. Select **OK** to check in the object.

## Reverting Changes

Reverting changes will undo any changes you made since you checked out an object.

To revert changes:

1. Go to the object that you want to revert.
2. Select **Revert** in the **Revision Control** dashboard zone.
3. In the confirmation dialog box select **OK** to confirm the action or **Cancel** to return to the object page.

Once reverted, the object can be checked out by another user.

## Forcing a Check In or Restore

You can force a check in if an object is checked out by another user and that person is not available to check it in.

You must have proper access rights to force a check in or restore.

To force a check in or restore:

1. Go to the object that is checked out by another user.
2. Select **Force Check In** or **Force Restore** in the Revision Control zone.

The **Force Check In** option is the same as a regular check in. The **Force Restore** option checks in the object and restores it to the previously checked in version.

## Deleting an Object

If revision control is turned on for an object, you must use the **Revision Control** zone to delete it.

The object must be checked in before it can be deleted.

To delete a revision controlled object:

1. Go to the object that you want to delete.
2. Select **Delete** in the **Revision Control** zone.
3. Provide details regarding the deletion.
4. Select **OK** to delete the object.

The system creates a revision record before the object is deleted so that the deleted object can be restored.

## Restoring an Object

You can restore an older version of either a current object or a deleted object.

An object must be checked in before an older version can be restored.

To restore an object:

1. Go to the **Revision History** portal for the object.  
If the object was deleted you must search for it by going to **Admin > Revision Control**.
2. Select the desired entity by clicking the hyperlink in the **Details** column.
3. Locate the row in the version history that has the version that you want to restore and click **Restore**.
4. In the confirmation dialog box select **OK** to confirm the action or **Cancel** to return to the object page.

## Working with the Revision Control Portal

The **Revision Control** portal lists information about each version of a revision controlled object.

You can navigate to the **Revision Control** portal from either a link in the **Revision Control** dashboard zone or by going to **Revision Control** portal through **Admin**.

If you want to find the Revision History entry for an earlier version or deleted object, you must search for the object using the **Revision Control Search** portal. Once you select the desired entry, you can restore a previous version of the object

clicking **Restore** in the row for the version that you want to restore. You can also see the details of each version by clicking the broadcast icon for that version.

See [Working with Revision Control Zones](#) for more information about tasks that can be performed through Revision Control.

## Revision Control Search

The **Revision Control Search** portal allows users to search for entities that have a revision history. The **Search By** dropdown provides additional functionality so that users can search for revisions that are associated to theirs or other's user ID. Users can also **Check In**, **Force Check In**, or **Check Out** one or more entities through this portal.

### Zone Options

- **Revision History Search** (*default*) allows the user to query for revised entities based on a combination of criteria.
  - In the **User ID** field, enter the user ID that is associated with a revision.
  - In the **External Reference** field, enter an ID from an external system and is associated with a revision.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key 1**, **Key 2**, **Key 3**, **Key 4**, **Key 5** fields, enter the primary identifier(s) for the revised entity. Typically, the entity only requires a single key, but some entities require more than one (for example, Oracle Utilities Customer Care and Billing SA Type require CIS Division and SA Type).
  - In the **Status** dropdown menu, select the entity status for your search.
- **Check In** allows the user to search for entities currently checked out to the logged in user ID and a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them in.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key** field, enter the primary identifier(s) for the revised entity.
- **Force Check In** allows the user to search for entities that are currently checked out by other user IDs (excluding the logged in user ID) based on a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them in.
  - In the **Checked Out By User** field, enter the user ID that has the entity in a Checked Out status.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key** field, enter the primary identifier(s) for the revised entity.
- **Check Out** allows the user to search for entities currently checked in user ID and a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them out.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key** field, enter the primary identifier(s) for the revised entity.

Please see [Working with Revision Control Zones](#) for more information about working with individual entities.

## Information Lifecycle Management

Information Lifecycle Management (ILM) is designed to address data management issues, with a combination of processes and policies so that the appropriate solution can be applied to each phase of the data's lifecycle.

Data lifecycle typically refers to the fact that the most recent data is active in the system. As time progresses, the same data becomes old and unused. Older data becomes overhead to the application not only in terms of storage, but also in terms of performance. This older data's impact can be reduced by using advanced compression techniques, and can be put into slower and cheaper storage media. Depending on how often it's accessed, it can be removed from the system to make an overall savings of cost and performance. The target tables for ILM are transactional tables that have the potential to grow and become voluminous over time.

## The Approach to Implementing Information Lifecycle Management

This section describes the product approach to implementing ILM for its maintenance objects (MOs).

---

**NOTE:** The term archiving is used to cover any of the possible steps an implementation may take in their data management strategy, including compression, moving to cheaper storage, and removing the data altogether.

---

Age is the starting point of the ILM product implementation for some of its high volume data. In general "old" records are considered eligible to be archived. In the product solution, maintenance objects (MOs) that are enabled for ILM have an ILM Date on the primary table and the date is typically set to the record's creation date. (An MO may have special business rules for setting this date, in which case, a different date may be used to set the initial ILM Date). For implementations that want to use ILM to manage the records in the MO, the ILM date is used for defining partitions for the primary table.

There are cases where a record's age is not the only factor in determining whether or not it is eligible to be archived. There may be some MOs where an old record is still 'in progress' or 'active' and should not be archived. There may be other MOs where certain records should never be archived. To evaluate archive eligibility using information other than the ILM Date, the ILM enabled MOs include an ILM Archive switch that is used to explicitly mark records that have been evaluated and should be archived. This allows DBAs to monitor partitions based on age and the value of this switch to evaluate data that may be ready to be archived.

Evaluating records to determine their archive eligibility should still occur on "old" records. The expectation is that a large percentage of the old records will be eligible for archiving. The small number that may be ineligible could be updated with a more recent ILM date. This may cause the records to move into a different partition and can delay any further evaluation of those records until more time has passed.

For each MO enabled for ILM, the product provides a batch process to review "old" records and an ILM eligibility algorithm that contains business logic to evaluate the record and mark it eligible for archiving or not. The following sections provide more information about the batch process and algorithm functionality.

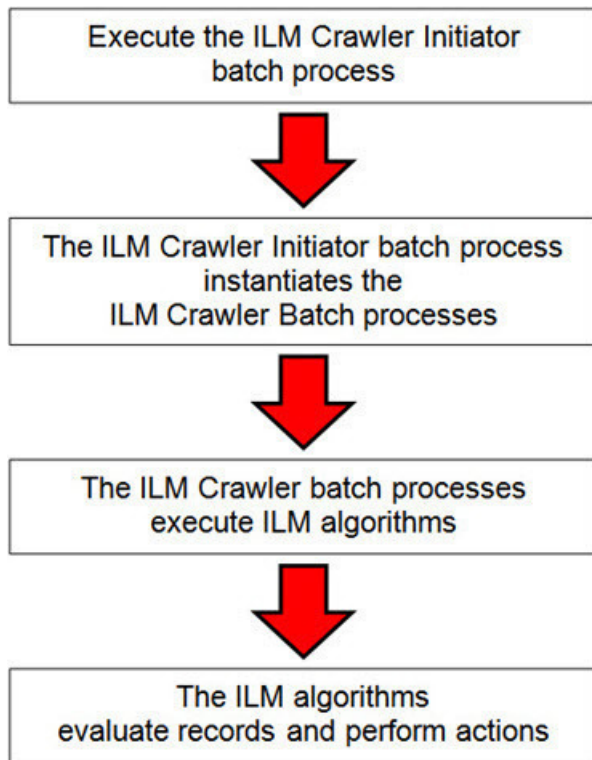
### Batch Processes

There are two main types of batch processes that manage data for ILM in the application: ILM Crawler Initiator and individual ILM Crawlers (one for each MO that is configured for ILM).

- **ILM Crawler Initiator: (F1-ILMIN)** - The ILM Crawler Initiator is a *driver* batch process that starts the individual ILM Crawler batch control as defined by the MO's options.

**Restartable:** In case of server failure, the ILM Crawler Initiator process can be restarted, which will also restart the ILM Crawler processes.

- **ILM Crawler:** Each maintenance object that is configured for ILM defines an ILM Crawler. These are *child* batch processes that can be started either by the ILM Crawler Initiator or by a standalone batch submission.



The ILM Crawler batch process selects records whose retention period has elapsed and invokes the MO's ILM eligibility algorithm to determine if the record is ready to be archived or not. The ILM eligibility algorithm is responsible for setting the record's ILM archive switch to 'Y' and updating the ILM date, if necessary.

The retention period defines the period that records are considered active. It spans the system date and cutoff date (calculated as system date - retention days).

The retention days of an MO is derived as follows:

- If the ILM Retention Days MO option is defined, that is used.
- Otherwise, the Default Retention Days from the ILM Master Configuration record is used.

An error is issued if no retention period is found.

The crawler calculates the cutoff date and selects all records whose ILM archive switch is 'N' and whose ILM date is prior to the cutoff date. Each record returned is subject to ILM eligibility.

If the Override Cutoff Date parameter is supplied, it will be used instead of the calculated cutoff date. An error is issued if the override cutoff date is later than the calculated cutoff date. This parameter is useful if an object has many years of historic data eligible for archiving. Setting this parameter allows for widening the retention period and therefore limiting the process to a shorter period for initial processing

---

**NOTE:** ILM Crawler batch processes are designed not to interfere with current online or batch processing. Because of this, these batch processes can run throughout the day.

---

**NOTE:** Before passing the cut-off date to the algorithm, the ILM crawler ensures that the number of days calculated (System Date – override cut-off date) is more than the retention period specified in the MO option or the Master Configuration. If the number of days calculated is **less than** the retention period specified on the MO option or the Master Configuration, then it throws an error.

---

## Eligibility Algorithm

Algorithms are triggered by the ILM batch crawler for the maintenance object. The key responsibility of the ILM algorithm is to determine whether a record can be marked as ready to be archived or not. If a record is determined to be ready for archive, the algorithm should set the ILM Archive switch to Y. If not, the algorithm leaves the switch set to N and may decide to update the ILM Date to something more recent (like the System Date) to ensure that the record does not get evaluated again until the future.

This algorithm is plugged into the [Maintenance Object — Algorithm](#) collection.

Oracle Utilities Application Framework provides the algorithm **ILM Eligibility Based on Status (F1-ILMELIG)** to support the ILM batch crawler. Refer to the algorithm type description for details about how this algorithm works. If a maintenance object has special business rules that are evaluated to determine the eligibility for ILM, a custom algorithm can be created and applied by the implementation team.

## Enabling ILM for Supported Maintenance Objects

In order to enable ILM for one or more maintenance objects, several steps are needed in both the configuration and in the database. This section describes some configuration enabled by default and some steps that must be taken in order to fully implement ILM.

There is some configuration enabled by default, but it won't be used unless ILM is fully configured. Each maintenance object that the product has configured for ILM has the following provided out of the box:

- **Special Table Columns:** Maintenance objects that support ILM include two specific columns: ILM Archive Switch (**ILM\_ARCH\_SW**) and ILM Date (**ILM\_DT**).
- **Crawler Batch Process:** A “crawler” batch process is provided for each maintenance object that supports ILM and it is plugged into the MO as an option. Refer to [Batch Processes](#) for more information.
- **ILM Eligibility Algorithm:** Each maintenance object that is configured for ILM defines an [eligibility algorithm](#) that executes the logic to set the ILM Archive switch appropriately. This is plugged in to the MO algorithm collection.

If an implementation decides to implement ILM, there are steps that need to be followed, which are highlighted below.

### Create the Master Configuration Record

The first step when enabling ILM is to create the **ILM Configuration** master configuration record. Navigate using **Admin Menu > Master Configuration**.

The Master Configuration for ILM Configuration defines global parameters for all ILM eligible Maintenance Objects. For example, the Default Retention Period. In addition, your product may implement additional configuration. Refer to the inline help for specific details about the information supported for your product's ILM configuration.

In addition, the user interface for this master configuration record displays summary information about all the maintenance objects that are configured to use ILM.

### Confirm the Maintenance Objects to Enable

In viewing the list of maintenance objects that support ILM in the ILM master configuration page, your implementation may choose to enable ILM for only a subset of the supported maintenance objects. For example, some of the maintenance objects may not be relevant for your implementation. Or perhaps, the functionality provided by the maintenance object is used, but your implementation does not expect a high volume of data.

For each maintenance object that your implementation has confirmed for ILM, the following steps should be taken:

- Determine if the maintenance object should have a different default retention days than the system wide value defined on the master configuration. If so, use the MO option **ILM Retention Period in Days** to enter an override option for this maintenance object.
- Review the functionality of the ILM Eligibility algorithm provided by the product for this maintenance object. Each algorithm may support additional configuration based on business needs. If your organization has special business rules that aren't satisfied by the algorithm provided by the product, a custom algorithm may be provided to override the base algorithm.

For each maintenance object that your implementation does not want to enable for ILM, inactivate the eligibility algorithm. This will ensure that the ILM Crawler Initiator background process does not submit the crawler batch job for the maintenance object in question.

- Go to the [Maintenance Object — Algorithm](#) tab for each maintenance object and take note of the **ILM Eligibility** algorithm code.
- Go to the [Maintenance Object — Option](#) tab for the same maintenance object and add an option with an option type of **Inactivate Algorithm** and the value set to the ILM eligibility algorithm noted in the previous step.

## Database Administrator Tasks

In order to implement ILM for each of the maintenance objects determined above, your database administrator must perform several steps in the database for the tables related to each MO. The following points are a summary of those steps. More detail can be found in the Information Lifecycle Management section of your product's *Database Administration Guide*.

- **Initializing ILM Date:** Your existing tables that are enabled for ILM may not have the ILM Date and ILM Archive switch initialized on all existing records. When choosing to enable ILM, a first step is to initialize this data based on recommendations provided in the DBA guide.
- **Referential Integrity:** The recommended partitioning strategy for child tables in a maintenance object is referential partitioning. In order to implement this, database referential integrity features must be enabled.
- **Partitioning:** This provides a way in which the data can segregate into multiple table partitions and will help in better management of the lifecycle of the data.

## Schedule the ILM Crawler Initiator

The final step of enabling the system for ILM is to schedule the ILM crawler initiator **F1-ILMIN** regularly based on your implementation's need. It is recommended to only schedule this batch process once all the required database activities are complete.

## Ongoing ILM Tasks

For an environment where ILM is enabled, besides the periodic execution of the ILM crawler batch processes to review and mark records, your database administrator has ongoing tasks.

The DBA reviews and maintains partitions and identifies partitions that may warrant some type of archiving step. The Information Lifecycle Management section of your product's *Database Administration Guide* provides more information for your DBA.

## Archived Foreign Keys

If your DBA choose to archive a partition, there may be records in the system that refer to one of the archived records as a foreign key.

When a user attempts to view a record using a hyperlink or drill down mechanism, if the implementation of the navigation uses FK Reference functionality, the system will first check if the record exists. If not, it will display a message to the user indicating that the record has been archived.

## Configuration Tools

---

This section describes tools to facilitate detailed business configuration. The configuration tools allow you to extend both the front-end user interface as well as create and define specialized back-end services.

## Business Objects

A [maintenance object](#) defines the physical tables that are used to capture all the possible details for an entity in the system. A business object is tool provided to further define business rules for a maintenance object.

This section provides an overview of business objects and describes how to maintain them.

## The Big Picture of Business Objects

The topics in this section describe background topics relevant to business objects.

## What Is A Business Object?

A business object (BO) is a powerful tool used throughout the system. For many maintenance objects, a BO is a key attribute of the record used to define the data it captures, its user interface behavior and its business rules. Some business objects support the definition of a lifecycle, capturing different states that a record may go through, allowing for different business rules to be executed along the way. This type of business object is considered the “identifying” or “governing” business object. We will see later that other types of BOs exist that are different from the “identifying” business object.

The use of business objects allows for extensibility and customization of product delivered maintenance objects. There are many options to adjust the behavior of base delivered business objects. In addition, implementations may introduce their own business objects if the base product delivered objects do not meet their business needs.

---

**NOTE:** Not all maintenance objects in the product support business objects as a “identifying” or “governing” tool. This is the standard going forward for new maintenance objects. However, there are some maintenance objects created before this became a standard.

---

## A Business Object Has a Schema

A business object has elements. The elements are a logical view of fields and columns in one of the maintenance object’s tables. The structure of a business object is defined using an XML schema. The main purpose of the schema is to identify all the elements from the maintenance object that are included in the business object and map them to the corresponding maintenance object fields. Every element in the BO schema must be stored somewhere in the maintenance object. The BO may not define elements that are derived.

When defining elements for the primary table or the language table (for an administrative object) there is no need to define the name of the physical table in the schema. The system infers this information. The following is a snippet of a schema:

```
<schema>
  <migrationPlan mapField="MIGR_PLAN_CD" suppress="true" isPrimeKey="true"/>
  <bo mapField="BUS_OBJ_CD" fkRef="Fl-BUSOB"/>
  <customizationOwner mapField="OWNER_FLG" suppress="input"/>
  <version mapField="VERSION" suppress="true"/>
  <description mapField="DESCR"/>
```



```
<longDescription mapField="DESCRLONG" />
...
```

Many maintenance objects have child table collections (e.g., a collection of names for a person, or a collection of persons on an account). Depending on the requirements, the business object may define the full collection such that the user will maintain the information in a grid. However, the schema also supports “flattening” records in a child table so that they can be treated as if they were singular elements. The following are examples of each:

**Example** of a child table. This is a snippet of the Instructions collection on the migration plan business object. You can see that the list attribute defines the child table and all elements within it map to the appropriate column in that table.

```
<migrationPlanInstruction type="list" mapChild="F1_MIGR_PLAN_INSTR">
  <migrationPlan mapField="MIGR_PLAN_CD" suppress="true" />
  <sequence mapField="PLAN_INSTR_SEQ" suppress="true" />
  <instructionSequence mapField="INSTR_SEQ" />
  <instructionType mapField="INSTR_TYPE_FLG" />
  <parentInstructionSequence mapField="PARENT_INSTR_SEQ" />
  <businessObject mapField="BUS_OBJ_CD" fkRef="F1-BOMO" />
...
```

**Example** of a simple “flattened” field. The business object for Status Reason includes an element called Usage, which maps to a pre-defined characteristic of type **F1-SRUSG**. The “row” defines which child table is being flattened and the attributes of the row in that child that uniquely identify it.

```
<usage mdField="STATUS_RSN_USAGE" mapField="CHAR_VAL">
  <row mapChild="F1_BUS_OBJ_STATUS_RSN_CHAR">
    <CHAR_TYPE_CD is="F1-SRUSG" />
    <SEQ_NUM is="1" />
  </row>
</usage>
```

**Example** of a “flattened row”. This business object for Account includes a single row for the Person collection where only the “financially responsible, main” customer is defined. The “accountPerson” attribute defines one field from that row (the Person Id) and includes the ‘flattening’ criteria in the “row” information. In addition, a second field from that same row (“accountRelType”) is defined. Instead of having to repeat the flattening criteria, the “rowRef” attribute identifies the element that includes the flattening.

```
<accountPerson mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true" />
    <FIN_RESP_SW default="true" />
  </row>
</accountPerson>
<accountRelType mapField="ACCT_REL_TYPE_CD" rowRef="accountPerson" dataType="string" />
```

**Example** of a “flattened list”. The business object for Tax Bill Type includes a list of valid algorithms for “bill completion”. The Sequence and the Algorithm are presented in a list. The list element identifies the child table and the ‘rowFilter’ identifies the information about the list that is common.

```
<taxBillCompletion type="list" mapChild="C1_TAX_BILL_TYPE_ALG">
  <rowFilter suppress="true" private="true">
    <TAX_BILL_TYPE_SEVT_FLG is="C1BC" />
  </rowFilter>
  <sequence mapField="SEQ_NUM" />
  <algorithm mapField="ALG_CD" fkRef="F1-ALG" />
</taxBillCompletion>
```

In addition, many maintenance objects support an XML structure field within the entity. These fields may be of data type CLOB or XML. One or more business object elements may be mapped to the MO's XML structure field. These elements may be defined in different logical places in the business object schema based on what makes sense for the business rules. When updating the MO, the system builds a type of XML document that includes all the elements mapped to the XML structure and stores it in one column. The following is an example of elements mapped to an XML column:

```
<filePath mdField="F1_FILE_PATH" mapXML="MST_CONFIG_DATA" required="true" />
>
  <characterEncoding mdField="F1_CHAR_ENCODING" mapXML="MST_CONFIG_DATA" />
```

Elements mapped to an MO XML structure field are typically ones that do not have to be exposed for SQL indexing or joining.

---

**NOTE:** If the MO's XML structure field is of the data type XML, the database will allow searching for records based on that data, assuming appropriate indexes are defined. If the MO's XML structure field is of the data type CLOB, indexing or joining to elements in this column via an SQL statement is not typically supported. Note that most MOs are currently using the CLOB data type for the XML structure column, if provided.

---

Some business objects may have child tables that allow data to be stored in an XML structure field. The schema language supports defining elements from those fields in your schema as well.

Besides including information about the physical "mapping" of the element to its appropriate table / field location in the maintenance object, the schema supports additional syntax to provide the ability to define basic validation and data manipulation rules, including:

- Identifying the primary key of the record or the primary key of the a row in a list.
- Identifying which elements are required when adding or changing a record.
- Default values when no data is supplied on an Add.
- For elements that are lookup values, the lookup may be specified to validate that the value of the element is a valid lookup value.
- For elements that are foreign keys to another table, the FK Reference may be specified to validate the data.

The system will check the validity of the data based on the schema definition obviating the need for any special algorithm to check this validation.

In addition, the schema language may include some attributes that are used to auto-render the view of the record on the user interface, such as the **suppress** attribute. Refer to [BO Defines its User Interface](#) for more information.

---

**NOTE: Schema Definition Tips.** A context sensitive "Schema Tips" zone appears when you open the [Business Object](#) page to assist you with the schema definition syntax. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

---

A business object's schema may include a subset of the fields and tables defined in the maintenance object. There are two reasons for this:

- The fields or tables may not be applicable to the type of record the business object is governing. For example, a field that is specific to gas may not be included on a Device business object that is specific to electric meters.
- The information is not maintained through the business object, but rather maintained separately. For example, many BO based maintenance objects include a Log table. The records in the log table are typically not included the BO because they are viewed and maintained separately from the business object.

## A Business Object May Define Business Rules

A business object may define business rules that govern the behavior of entities of this type.

- Simple element-level validation is supported by schema attributes. Note that element-level validation is executed before any maintenance object processing. For more sophisticated rules you create **Validation** algorithms and associate them with your business object. BO validation algorithms are only executed after "core validation" held in the MO is passed.
- A **Pre-Processing** algorithm may be used to "massage" a business object's elements prior to any maintenance object processing. For example, although simple element-level defaulting is supported by schema attributes. you may use this type of algorithm to default element values that are more sophisticated.
- A **Post-Processing** algorithm may be used to perform additional steps such as creating a To Do Entry or add a log record as part of the business object logical transaction. These plug-ins are executed after all the validation rules are executed.
- An **Audit** algorithm may be used to audit changes made to entities of this type.  
Any time a business entity is added, changed or deleted, the system detects and summarizes the list of changes that took place in that transaction and hands it over to **Audit** plug-ins associated with the business object. These plug-ins are

executed after all the post-processing rules are executed. It is the responsibility of such algorithms to log the changes if and where appropriate, for example as a log entry or an entry in an audit trail table or an entry in the [business event log](#)

By default all elements of the business object are subject to auditing. You can however mark certain elements to be excluded from the auditing process using the **noAudit** schema attribute. Marking an element as not auditable will prevent it from ever appearing as a changed element in the business object's audit plug-in spot. In addition, if the only elements that changed in a BO are ones marked to not audit, the audit algorithm is not even called. Refer to the "Schema Tips" context sensitive zone associated with the Business Object maintenance page for more information on this attribute.

Refer to [Business Object - Algorithms](#) for more information on the various types of algorithms.

The system applies business object rules (schema based and algorithms) whenever a business object instance is added, changed or deleted. This is only possible when the call is made via the maintenance object service. For example, when made via business object interaction ("invoke BO"), the MO's maintenance page or inbound web services that reference the BO. In addition the system must be able to [determine the identifying business object](#) associated with the actual object being processed. If the business object cannot be determined for a maintenance object instance business object rules are not applied.

---

**NOTE:**

**Pre-Processing is special.** The pre-processing algorithm plug-in spot is unique in that it only applies during a BO interaction. It is executed prior to any maintenance object processing. It means that when performing add, change or delete via the maintenance object service, the pre-processing plug-in is not executed.

---

**CAUTION:** Direct entity updates bypass business rules! As mentioned above, it is the maintenance object service layer that applies business object rules. Processes that directly update entities not via the maintenance object service bypass any business object rules you may have configured.

---

**FASTPATH:** Refer to [BO Algorithm Execution Order](#) for a summary of when these algorithms are executed with respect to lifecycle algorithms.

---

The plug-in spots described above are available for all business objects and they are executed by the system when processing adds or updates to the business object. It is possible for a specific maintenance object to define a special plug-in spot for business objects of that MO. When this happens, the maintenance object identifies the special algorithm entity lookup value as an [MO option](#): **Valid BO System Event**, causing the BO Algorithm collection to include that system event in its list.

## A Business Object Defines its User Interface

One of the responsibilities of an identifying business object is to define its user interface rules for viewing and maintenance of its record. The standard implementation for maintaining a business object is that a [maintenance portal](#) is used to display a record. This portal includes a "map" zone that displays the information about the business object. To add or make changes to a record, the user clicks a button that launches a maintenance BPA script which displays a maintenance "map".

The display and maintenance "maps" are driven by the business object. The BO may define a full [UI map](#) where all the information is displayed based on the map's HTML. Note that for a child BO, the maps may be [inherited](#) by a parent BO (or any BO "up the chain").

The standard going forward is to use schema definition and UI Hints to define user interface behavior so that a full UI map is not needed but rather the HTML is derived. The schema language includes some basic display attributes such as **label** and **suppress**. UI hints provide many additional tags and elements that allow dynamic generation of formatted UI Maps. For more complex behavior in the user interface, for example where javascript is needed, UI map fragments may be defined within the schema via UI hints. In this way only complex UI behavior warrants small snippets of javascript and HTML. However the rendering of standard fields can be dynamically rendered. UI map fragments also allow for derived fields to be included in the user interface.

A business object schema may include [data areas](#) for segments of its schema definition to allow for reusable components. In this case the data area would also include schema attributes and UI hints for the elements that it is including.

---

**NOTE: Schema Definition Tips.** The context sensitive "Schema Tips" zone appears when you open the [Business Object](#) page. This includes detailed information about supported UI hint definition syntax.

---

As mentioned in [Business Object Inheritance](#), schemas are not inherited on a child business object. As such, when using UI hints for automatic UI rendering, the child BO must define the full schema with all the definitions. A good business object hierarchy will be designed for reuse meaning that the child BO will include the parent BO schema or alternatively, the BO schemas will include reusable data areas.

## Invoking A Business Object

We have talked about defining a business object. This section describes how business objects are used throughout the system to view, add and update records.

- Various parameters for zones that are used to display data in the system include support for retrieving data by referencing a business object. The zone code will “invoke” the BO, meaning that the record will be retrieved using the referenced BO.
- The system’s [scripting](#) language includes a step type to “invoke BO”. This allows for BPA scripts, service scripts and plug-in scripts to retrieve information and add or update records using BO interaction.
- [Inbound web services](#) may reference a business object in its operations collection. This allows external systems to add or update records in our product via web service interaction.

Often when configuring a zone or writing a script, the BO to use in the “invoke BO” statement should be the identifying BO of the record. As such, often the script will include steps prior to the “invoke BO” step to “[determine the identifying BO of the record](#)” and once the identifying BO is found, the script step will invoke that BO. Note that zones and inbound web services reference a BO directly. In each case, if the BO to use should be dynamic, then the zone / inbound web service should reference a service script that can perform the steps to identify the BO and then invoke that BO.

It should be noted however that the BO used in an “invoke BO” statement (or referenced in an inbound web service) **does not have to match** the identifying BO for the record. Here are some examples of where this may be true:

- A script may only require a subset of elements for a record and not the entire record. In this case, it is better for performance purposes to define a special BO (sometimes called a “lite” BO or a “mini” BO) that only defines the needed elements. When the system retrieves the data, it will only access the tables that are included in the BO’s schema definition. In addition, if there are no elements that map to an XML structure field, the system will skip any parsing of that column. Similarly, if a script is **updating** a subset of elements on a record, it may be beneficial to use a “mini” BO to do the updates.

---

**NOTE:** Please note the following with respect to using a mini BO. This BO is only used for its schema. This type of BO would not define algorithms or a lifecycle. Because the BO is special, it often should not be able to be used as any record’s identifying BO. To control that, these BOs are often configured to not allow new instances. Refer to [Determine the Identifying BO](#) for more information.

---

- The maintenance object to be added or updated in a script may not support business objects as “identifying BOs”. For example, Batch Control maintenance object does not have an identifying BO. However, scripts may still wish to retrieve data (or make updates) to these types of records. An easy way to achieve that goal is to define a business object and use “invoke BO” to access the data.

---

**NOTE:** Not all maintenance objects support being maintained through a business object interaction. This is true in a small number of older objects where the underlying maintenance service includes additional functionality besides

simply updating the database tables. These maintenance objects are identified via the [MO option](#) **BO Maintenance**, set to **N**.

---

- Some functionality may be trying to add or update records for a maintenance object in a ‘physical’ manner and do not want or need to use the object’s identifying BO. Or the MO may not have an identifying BO. For example, revision control takes a snapshot of a record for audit purpose and to be able to restore a previous version. In this case, the system wants to capture a full “physical” view of the record. To do this, a special “physical” BO may be created that includes all (or most of) the columns and the child tables.

---

**NOTE:** Like the mini BO, the physical BO would not define algorithms or a lifecycle and should not be able to be used as any record’s identifying BO. To control that, these BOs are often configured to not allow new instances. Refer to [Determine the Identifying BO](#) for more information.

---

**NOTE:** To reiterate, the BO referenced in the “invoke BO” statement or referenced in an inbound web service does not have to match the identifying BO and does not have to be configured to “allow new instances”.

---

## Determine the Identifying BO

As mentioned in other topics, the identifying BO is the business object that governs the business rules for a record. This is the business object that the record will be validated against when any additions or changes are made to the record as long as updates are made via the maintenance service. This includes using “invoke BO” for add or update, using inbound web service interaction and for access to the maintenance page service (via an old style fixed page or via a business service).

How does the system determine the identifying BO? An algorithm plugged into the maintenance object (the **Determine BO** plug-in spot) is responsible for this. If the maintenance object is not configured with an algorithm for this plug-in spot, or no BO is found by the algorithm, no BO business rules are applied.

Most maintenance objects in the system capture the record’s identifying BO directly on the record. However, it is possible to define the identifying BO somewhere else. For example, there may be some maintenance objects that are master or transaction objects with an associated “type” object where the identifying BO is defined on its “type” object. Note that the standard Determine BO algorithm plugged into most maintenance objects (**F1-STD-DTMBO — Determine Standard Business Object**) checks for these two conditions.

There may also be cases where a single identifying BO is used for all BOs for a given MO. This may be an option used for some older maintenance object created prior to the business object functionality when implementations wish to introduce custom business rules that are common for all records of that MO. The product provides a base algorithm type (**F1-MOBO — Determine Specific Business Object**) that captures the BO as a parameter.

## Base Business Objects

For each maintenance object (MO) that supports an “identifying” business object, the type of business object provided by the product depends on the functionality and expected use by implementations. The following are some common patterns.

- There are MOs where the product provides base BOs that implementations may use if applicable for their business rules. In addition, it is expected that implementation will define custom BOs to support their business needs. Good examples of this type of MO are any of the various “rule” MOs. For example, calculation rule in Oracle Utilities Customer Care and Billing or the usage rule in Oracle Utilities Meter Data Management or the form rule in Oracle Public Sector Revenue Management. The product provides business objects for common rules but each implementation could have special rules that they need to implement and will need to create custom business objects.
- There are MOs where the product provides base BOs that supply common behavior for an object. Implementations may find that supplied the business objects match their business requirements and use the BOs as is. It is expected, however that for many implementations, their business rules will require additional elements to be captured or have special rules

to apply. In this case the base business objects may be extended. This scenario may apply to ‘master’ data objects in various products such as the Device or Meter or Tax Role.

- There are MOs where the product may deliver a base BO that is not expected to satisfy most implementations because different jurisdictions or different implementations will typically have their own rules. In this case the base delivered BO can be used as a template or starting point for custom defined BOs. Some examples of this are Rebate Claim in Oracle Utilities Customer Care and Billing or the Appeal object in Oracle Public Sector Revenue Management.
- There are MOs where the expectation is that every implementation will have different requirements for the type of data to capture and the product will not supply base BOs that can be used as the “identifying” BO. However, it may supply a “parent” BO that defines the lifecycle and many of the business rules that it expects all records to follow. In these scenarios, the implementations will create “child” BOs that will serve as the “identifying” BOs and refer to the base “parent” BO for many of its rules through [inheritance](#). Some examples of the are Tax Form in Oracle Public Sector Revenue Management or Activity in Oracle Utilities Mobile Workforce Management.
- There are some scenarios where the base product provides business objects and the expectation is that implementations will use the business objects as delivered with little or no customization. This is a case where the system used business objects to implement product functionality, not because there is an expectation that the implementers will extend the functionality, but because the business object model is the favored development tool even for the product. The objects delivered for Configuration Migration Assistant are an example.

---

**NOTE:** Not all maintenance objects in the product support business objects as a “identifying” or “governing” tool. This is the standard going forward for new maintenance objects. However, there are some maintenance objects created before this became a standard.

---

For all maintenance objects, the base product may provide additional BOs that are not meant to be “identifying” BOs, but instead are provided to support functionality to interact with the MO using the BO as a tool as described in [Invoking a BO](#).

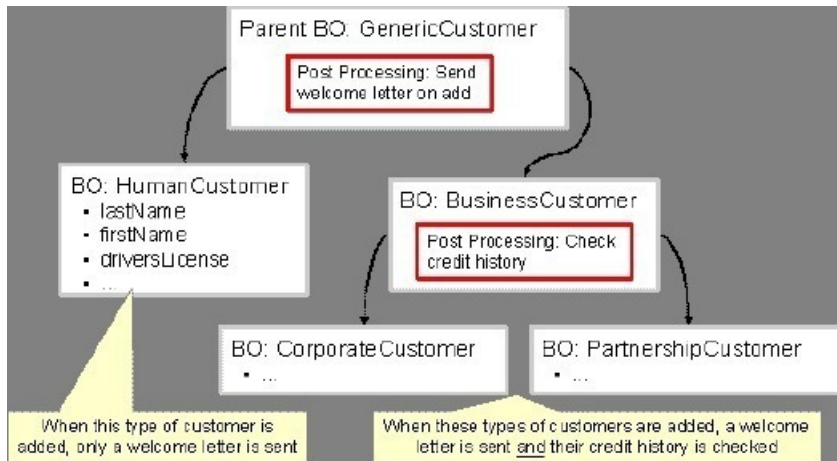
- One or more “mini” or “lite” BOs may be supplied for a maintenance object. This may be found when the product has functionality to retrieve a subset of elements for the maintenance object via scripting or via a user interface.
- A “physical” BO may be supplied. This a BO that typically includes all tables and all fields of the maintenance object in there “physical” form. In other words, there is no “flattening” of child tables and any XML structure fields are defined as a single field. Physical BOs are used in system processing where the full record needs to be captured as is. Some functionality that uses a physical BO includes [bundling](#) and [revision control](#).

## Business Object Inheritance

A business object may inherit business rules from another business object by referencing the latter as its parent. A child business object can also have children, and so on. A parent's rules automatically apply to all of its children (no compilation - it's immediate). A child business object can always introduce rules of its own but never remove or bypass an inherited rule.

The following is an illustration of multiple levels of business object inheritance.





Notice how the "Business Customer" business object extends its parent rules to also enforce a credit history check on all types of customers associated with its child business objects.

Most types of business object system events allows for multiple algorithms to be executed. For example, you can have multiple **Validation** algorithms defined for a business object. For these, the system executes all algorithms at all levels in the inheritance chain starting from the highest-level parent business object moving on to lower levels.

Other types of system events allows for a single algorithm to be executed. For example, you can only have one **Information** algorithm to format the standard description of a business object instance. For these, the system executes the one at the level nearest to the business object currently being processed.

---

**NOTE:** The parent and its children must reference the same maintenance object.

---

**NOTE: Data structures are not inherited.** While you can declare schemas on parent business objects, their children will not inherit them. A good practice is to design child business object schemas to **include** their parent business object's schema.

---

**NOTE: User interface maps are inherited.** When determining if the business object has a UI map to use for UI rendering, the system looks for display and maintenance maps linked to the BO as options. If the identifying BO does not have maps defined, the system follows "up the chain" of inheritance until it finds a map to use. This allows for a child BO to be used to extend business rules of a parent BO but inherit its user interface behavior. Refer to [Business Object Defines its User Interface](#) for more information.

---

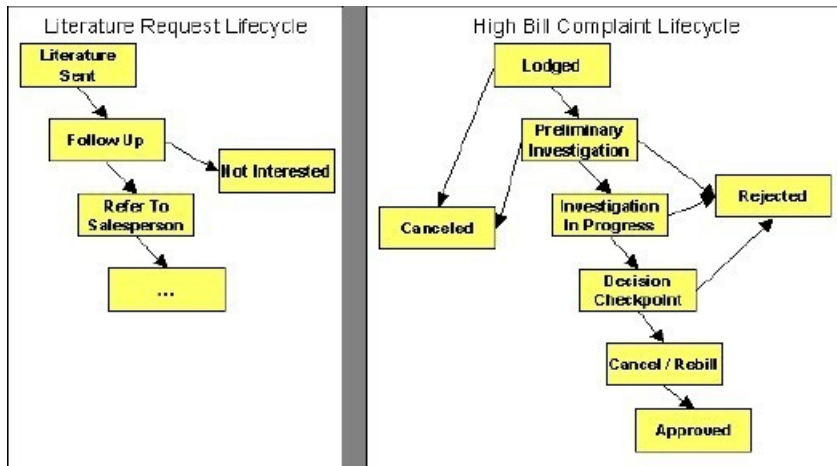
**NOTE: Use Inheritance Wisely.** While it is intellectually attractive to abstract behavior into parent BOs to avoid redundant logic and simplify maintenance, before doing this weigh the reuse benefits against the cost in transparency, as it is not easy to maintain a complex hierarchy of business objects.

---

## Each Business Object Can Have A Different Lifecycle

Many maintenance objects have a status column that holds the business entity's current state within its lifecycle. Rules that govern lifecycle state transition (e.g., what is its initial state, when can it transition to another state, etc.) and the behavior associated with each state are referred to as lifecycle rules. Older Maintenance Objects, such as To Do Entry, have predefined lifecycles whose rules are governed by the base-package and cannot be changed. The lifecycle of newer Maintenance Objects exists in business object meta-data and as such considered softly defined. This allows you to have completely different lifecycle rules for business objects belonging to the same maintenance object.

Here are examples of two business objects with different lifecycles that belong to the same maintenance object.



**NOTE:** A Maintenance Object supports soft lifecycle rules if it is defined with a **Status Field Maintenance Object** option.

The topics that follow describe important lifecycle oriented concepts.

## Valid States versus State Transition Rules

The boxes in the above diagram show the potential valid states a business entity of the above business object can be in. The lines between the boxes indicate the state transition rules. These rules govern the states it can move to while in a given state. For example, the above diagram indicates a high bill complaint that's in the **Lodged** state can be either **Canceled** or moved into the **Preliminary Investigation** state.

When you set up a business object, you define both its valid states and the state transition rules.

## One Initial State and Multiple Final States

When you set up lifecycle states, you must pick one as the initial state. The initial state is the state assigned to new entities associated with the business object. For example, the above high-bill complaint business object defines an initial state of **Lodged**, whereas the literature request one defines an initial state of **Literature Sent**.

You must also define which statuses are considered to be "final". Typically when an entity reaches a "final" state, its lifecycle is considered complete and no further processing is necessary.

**NOTE: Allowing An Entity To Be "Reopened".** You can set up your state transition rules to allow a business entity to be "reopened" (i.e., to be moved from a final state to a non-final state). Neither of the above examples allows this, but it is possible if you configure your business object accordingly.

## State-Specific Business Rules

For each state in a business object's lifecycle, you can define the following types of business rules.

**FASTPATH:** Refer to [BO Algorithm Execution Order](#) for a summary of when these lifecycle algorithms are executed with respect to BO level algorithms.



## Logic To Take Place When Entering A State

You can define algorithms that execute before a business entity enters a given state. For example, you could develop an algorithm that requires a cancellation reason before an entity is allowed to enter the *Canceled* state.

You can also incorporate state auto-transitioning logic within this type of algorithms. Refer to [auto-transition](#) for more information.

## Logic To Take Place When Exiting A State

You can define algorithms that execute when a business entity exists a given state. For example, you could develop an algorithm that clears out error messages when a given entity exits the *Error* state.

## Monitor Rules

You can define algorithms to monitor a business entity while it is in a given state. This type of logic is typically used to check if the conditions necessary to [transition](#) the entity to another state exist (and, if so, transition it). For example, transition an entity to the *Canceled* state if it's been in the **Error** state too long. Another common use is to perform ancillary work while an entity is in a given state. For example, update statistics held on the object while it's in the **Active** state.

[Monitor algorithms](#) are invoked when a business entity first enters a state and periodically after that in batch. You have the option to defer the monitoring of a specific state until a specific monitoring batch job runs. You do so by associating the state with a specific monitoring process. In this case the system will only execute the monitoring rules of this state when that specific batch process runs. This is useful when processing one type of record typically creates another type of record. You may want the processing of the second set of records to be deferred to a later time.

A monitor algorithm can carry out any business logic. In addition it can optionally tell the system to do either of the following:

- Stop monitoring and transition to another state. The system will not call any further monitoring algorithm plugged in on the state and attempt to transition the entity to the requested new state.
- Stop monitoring. Same as above except that no transition takes place. You may want to use this option to prevent transitions while some condition is true.

If none of the above is requested the system keeps executing subsequent monitoring algorithms.

---

**FASTPATH:** Refer to [Business Object - Lifecycle](#) for more information on how to set up state-specific algorithms.

---

## Inheriting Lifecycle

If a business object references a parent business object, it always [inherits](#) its lifecycle from the highest-level business object in the hierarchy. In other words, only the highest-level parent business object can define the lifecycle and the valid state transitions for each state. Child business objects, in all levels, may still extend the business rules for a given state by introducing their own state-specific algorithms.

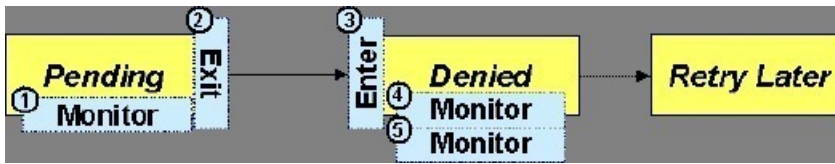
The system executes all state-specific algorithms at all levels in the inheritance chain starting from the highest-level parent business object moving on to lower levels.

## Auto-Transition

In a single transition from one state to another, the system first executes the **Exit** algorithms of the current state, transitions the entity to the new state, executes the **Enter** algorithms of the new state followed by its **Monitor** algorithms. At this point if a **Monitor** algorithm determines that the entity should be further automatically transitioned to another state the remaining monitoring algorithm defined for the current state are NOT executed and the system initiates yet another transition cycle.

Notice that an **Enter** algorithm can also tell the system to automatically transition the entity to another state. In this case the remaining **Enter** algorithm as well as all **Monitor** algorithms defined for the current state are NOT executed.

The following illustration provides an example of an auto-transition chain of events.



In this example a business entity is in a Pending state. While in that state a **Monitor** algorithm determines to auto-transition it to the Denied state. At this point the following takes place:

- No further **Monitor** algorithms of the Pending state are executed
- Pending state **Exit** algorithms are executed
- The system transitions the entity to the Denied state
- Denied state **Enter** algorithms are executed. No further auto-transition is requested.
- Denied state **Monitor** algorithms are executed. No further auto-transition is requested.

## Keeping An Entity In Its Last Successful State

By default, any error encountered while transitioning a business entity from one state to another rolls back ALL changes leaving the entity in its original state.

When applicable, the Maintenance Object can be configured to always keep an entity in its last successful state rather than rolling all the way back to the original state. This practice is often referred to as "taking save-points". In case of an error, the entity is rolled back to the last successfully entered state and the error is logged on the [maintenance object's log](#). Notice that with this approach no error is returned to the calling process, the error is just logged.

The logic to properly log the error is in a **Transition ErrorMaintenance Object plug-in**. The system considers a maintenance object to practice "save-points" when such an algorithm is plugged into it.

Even if the maintenance object practices "save-points", in case of an error the system will not keep an entity in the last successfully entered state if that state is either marked as **transitory** or one of its **Enter** algorithms has determined that the entity should proceed to a next state. The system will roll back to the first previous state that does not match these conditions.

## Monitoring Batch Processes

The base package provides a periodic monitoring batch process for each maintenance object that supports soft state transition. The process periodically executes the monitoring algorithms associated with the current state of an entity, excluding states explicitly referencing a deferred monitoring batch process.

A deferred monitoring process works in the same way but only considers entities whose current state references this particular batch control as their monitor process. Deferred monitoring is only needed for certain states based on business requirements.

Your business rules will dictate the execution frequency of each monitoring process and the order in which they should be scheduled.

---

**NOTE: Updates to the business object.** When the monitor algorithms indicate that the business object should transition, the monitor batch processes are responsible for ensuring the business object is transitioned appropriately and that the appropriate exit, enter and monitor algorithms are executed. Please note that the business object is not updated

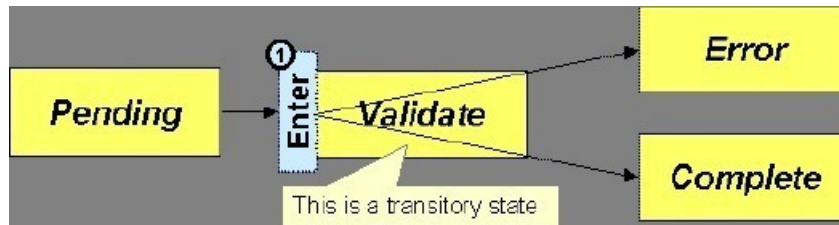
using a call to the maintenance object service and therefore the [business rules](#) plugged in to the business object are not executed.

---

## Transitory States

You can define a state as **Transitory** if you do not wish the business entity to ever exist in that particular state.

The following illustrates a lifecycle with a transitory Validate state.



In this example, the business entity is saved still not validated in the Pending state. At some point, the user is ready to submit the entity for validation and transitions it into a transitory state Validate whose **Enter** rules contain the validation logic. The responsibility of the transitory state's **Enter** algorithms is to decide if the entity is valid or in error and then transitions it into the appropriate final state. In this scenario, you may not ever want the business entity to exist in the Validate state.

Let's also assume that the maintenance object in this example is practicing "[save-points](#)" and requires the entity to be kept in its last successful state. If an error were to occur during the transition from **Validate** to the next state, the system would roll back the entity back to Pending, and not Validate even though the entity has successfully entered the Validate state. Refer to the [Auto Transition](#) section for more information.

## State Transitions Are Audited

Most Maintenance Objects that support soft lifecycle definition also have a log to hold significant events throughout a business entity's lifecycle. For example, log entries are created to record:

- When the business entity is created (and who created it)
- When its status changes (and who changed it)
- If a transition error occurred (and the error message)
- References to other objects created throughout the entity's lifecycle. For example, if a To Do entry is created as part of processing the entity, the To Do Entry is referenced in the log.
- Manual entries added by a user (think of these as "diary" entries)

When a business entity is first created and when it transitions into a new state the system calls **Transition** algorithm(s) plugged in on the [Maintenance Object](#) to record these events. If the maintenance object supports a log these events can be captured as log entries.

---

**NOTE:** Most base package maintenance objects supporting a log may already provide state transition logging as part of their core logic. In this case you only need to provide a **Transition** plug-in if you wish to override base logging logic with your own.

---

## Required Elements Before Entering A State

You can define additional elements that are required before a business entity can enter a given state. For example, let's assume that a Cancel Reason must be defined before an object can enter the *Canceled* state. You do this by indicating that element as a **Required Element** [state-specific option](#) on the appropriate state on the business object.

## Capturing a Reason for Entering a State

Some business objects support configuring certain states to allow or require a status reason when an object enters the state. The product provides a centralized status reason table that may be used to define the valid BO status reasons for various business objects and various states. The status reasons are defined using the [Status Reason](#) portal.

The following sections provide additional information about the BO status reason functionality.

### Maintenance Object Must Support Status Reason

In order for a business object to use the centralized status reason table to define reasons, the maintenance object must first support the status reason. MOs that support status reason have the following characteristics:

- The primary table includes a column for Status Reason. This represents the status reason for the record's current status, if applicable.
- The log table includes a column for Status Reason. The standard logic for capturing a log record when entering a state also captures the status reason, if applicable. This allows a user to review the history of the changes in status and the status reason captured for a previous state transition, if applicable.
- The maintenance object option collection includes an option that defines the Status Reason field. This setting is a trigger for business objects of this MO to be able to configure states to allow or require status reason.

### Business Object State Indicates if Reasons are Applicable

Once the MO is configured to support status reason, configuration on the business object is required to indicate the states where a reason is applicable. States may be configured to require status reasons, allow status reasons as optional or not allow status reasons. With this configuration, the framework will automatically get the list of valid reasons for a state that allows or requires them and then prompt the user for a status reason when a manual state transition occurs for that state. It also automatically triggers an error if the state requires a status reason and no reason is provided.

---

**NOTE:** The status reason configuration on the business object state is customizable. That means that for a product owned business object, an implementation may opt to change the delivered configuration.

---

Status reasons are defined for the parent (or “lifecycle”) business object. All business objects in the hierarchy of the parent business object have the same valid reasons for their states.

The status reason code must be unique for the centralized status reason table. Business object and status are required fields, so it is not possible to share a common reason code (like “Not applicable”) across multiple business objects or states. If multiple BOs / states want to support a reason “Not applicable” then each must define a unique record for it. This point should be considered when planning for your status reasons.

### Selectable vs. Not Selectable

When defining a status reason, you may indicate whether it's **Selectable** or **Not Selectable**. When a manual transition is performed and a user is prompted for a status reason, only the **Selectable** reasons are presented. The **Not Selectable** reasons may be defined to support transitions that occur via algorithm processing.

---

**NOTE:** The Selectable setting is customizable. That means that if a product provides a base owned status reason for a business object state, an implementation may opt to change whether it is selectable or not. Careful consideration should be made before changing a base delivered status reason from **Not Selectable** to **Selectable** as this may affect base provided algorithm functionality that could be relying on the setting of **Not Selectable**.

---

### Status Reason Business Object

The status reason maintenance object, as with many maintenance objects in the product, references a business object used to define attributes and behavior related to defining status reasons. The framework provides a business object for status reason

**(F1–BOStatusReason)**. For the business objects that have states that require a status reason (let’s call these “transactional BOs”), if there is some special logic required for defining the status reasons, it is possible to define a different status reason BO. In this situation, the override status reason BO to use for capturing status reasons should be defined as a BO option on the transactional BO using the **Status Reason Business Object** option type. If a transactional BO does not define any status reason BO option, then the **F1–BOStatusReason** is used when adding a status reason.

## Defining a Usage

The base product status reason BO provides the ability to define a “usage” value. This is useful for algorithms that perform state transitions where a status reason is needed and where the algorithm is usable by more than one business object. In this case, the status reason to use cannot be provided as a parameter because each business object must define its own set of status reasons for each state. The Usage value can be used instead. Each business object can configure the status reason to use in the algorithm and set the appropriate usage value. The algorithm can reference the usage value and retrieve the correct status reason to use based on the record’s transactional BO.

The status reason business object provided with the framework product (**F1–BOStatusReason**) supports capturing a usage. The valid usage values are defined in the **Status Reason Usage** characteristic type.

## Alternatives for Defining Reasons

There may be business objects in the system that capture reasons that are defined somewhere besides the BO status reason table. For example, some objects may have an explicit administrative table for status reasons. Some objects may use a Lookup or an Extendable Lookup to capture reasons. Refer to the business object description for information about how valid reasons are defined, if applicable.

If a business object supports a reason that is not related to a state transition (such as a creation reason), the BO status reason would not be used. One of the alternate methods for defining a reason, described above, would be used.

## BO Algorithm Execution Summary

This table highlights the processing steps that occur when adding or changing a record that is governed by a business object.

Invoke BO	
Event	Comments
BO Pre-processing algorithms executed	These algorithms are only executed when Invoke BO is used. The business object in the Invoke BO is the one whose rules are executed.
MO Processing	
Event	Comments
Determine if status has changed.	The system keeps a note of the new status value but initially proceeds with the old value.
MO Processing.	Standard MO processing, including MO validation is executed.
Determine BO algorithm executed.	The MO level algorithm is executed to <a href="#">determine the identifying BO</a> .
BO Validation algorithms executed.	
<a href="#">State transition rules</a> are performed if the status has changed.	BO Status Exit algorithms for the “old” status executed. Status updated to the new value. BO Status Enter algorithms for the “new” status executed. If no error — MO Transition algorithms are executed.  <b>FASTPATH:</b> Refer to <a href="#">State Transitions are Audited</a> for more information.  If error and there are “save points” the MO Transition Error algorithms are executed.  <b>FASTPATH:</b> Refer to <a href="#">Keeping An Entity In Its Last Successful State</a> for more information.

	Otherwise, the error is reported.
BO Status Monitor algorithms are executed.	If the record transitions again, the prior step (State transition rule step) is repeated for the new transition.
BO Post-processing algorithms are executed.	
BO Audit algorithms are executed.	These algorithms are only executed if the system detects a change in elements that are not marked with "no audit".

**NOTE:** To emphasize, the steps in the MO Processing table are only executed when the maintenance object service is invoked. Any add or update initiated by an "invoke BO" statement will invoke the MO service. This is also true for web service that invoke the business object. Records processed by the [Monitor Batch Process](#) does not invoke the maintenance service. The monitor batch process only executes the monitor algorithms and the state transition rules (if the monitor algorithms indicate that a status change should occur).

**NOTE:** For records that do not have a status, the state transition rules and the monitor rules are not applicable.

## Granting Access To Business Objects

Every business object must reference an [application service](#). When you link a business object to an application service, you are granting all users in the group access to its instances. You can prevent users from transitioning a business object into specific states by correlating each business object status with each application service action (and then don't give the user group rights to the related action).

**FASTPATH:** Refer to [The Big Picture Of Application Security](#) for information about granting users access rights to an application service.

The system checks if a user has access rights each time the application is invoked to add, change, delete, read, or transition a business object. However, if a business object invokes another business object, we assume that access was controlled by the initial business object invocation and we do not check access for other business objects that it invokes. In other words, access rights are only checked for the initial business object invoked in a service call.

In order to apply business object security the system must be able to determine the business object associated with the actual object being processed. To do that the Maintenance Object itself has to have a **Determine BO** algorithm [plugged in](#). If this algorithm is not plugged in or it cannot determine the BO on the MO, the system will NOT invoke any BO rules. If the business object cannot be determined for a maintenance object instance, business object security is not checked. In this case the system checks the user's access rights using standard maintenance object security.

**NOTE: Parent business objects are ignored.** If a child business object exists, a user need only have access to the child business object's application service (not to every application service in the business object hierarchy).

## Defining Business Objects

The topics in this section describe how to maintain business objects.

Note that several context sensitive dashboard zones appear on this page and are visible on all tabs.

- **Schema Tips.** This zone provides several links to view details related to the valid schema syntax and UI Hint syntax.
- **View UI Rendering.** This zone provides buttons to view the automatic rendering of the Display map or Input map based on the attributes defined in the schema, including UI hints.
- **Generate Schema.** This zone includes a button that can be used to generate a "physical" schema based on the maintenance object definition. The element names are taken from the Java field name for each column. Once generated, adjust the schema as desired.



- **Create a BO Algorithm.** This zone includes a button to create a script based algorithm related to this business object. You are then prompted for information regarding the plug-in spot (BO or BO Status) and the system event, the name, description, etc. Once all the information is provided, the system creates an algorithm type, algorithm, links the algorithm to the business object, creates the script and brings you to the script step to start defining the logic for the plug-in script.
- **BOs Linked to the MO.** This zone displays other business objects for the same maintenance object as the BO currently displayed. You may drill into any of the other BOs by clicking its description.

## Business Object - Main

Use this page to define basic information about a business object. Open this page using **Admin > System > Business Object**

### Description of Page

Enter a unique **Business Object** name and **Description**. Use the **Detailed Description** to describe the purpose of this business object in detail. **Owner** indicates if this business object is owned by the base package or by your implementation (**Customer Modification**).

---

**CAUTION:** Important! If you introduce a new business object, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

Enter the **Maintenance Object** that is used to maintain objects of this type.

Enter a **Parent Business Object** from which to [inherit](#) business rules.

**Lifecycle Business Object** is only displayed for child business objects, i.e. those that reference a parent business object. It displays the highest-level business object in the inheritance hierarchy. Refer to [Inheriting Lifecycle](#) for more information.

**Application Service** is the application service that is used to provide security for the business object. Refer to [Granting Access To Business Objects](#) for more information. The application service on the child business object must have the same valid actions as the application service on the parent business object.

Use **Instance Control** to allow or prevent new entities from referencing the business object. Typically only the [identifying BOs](#) are marked to allow new instances.

Click the **View Schema** hyperlink to view the business object's expanded schema definition. Doing this opens the [schema viewer](#) window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

Click the **View MO** hyperlink to view the maintenance object in the [Maintenance Object Viewer](#). You may find it useful to leave the application viewer window open while defining your business object schema.

The options grid allows you to configure the business object to support extensible options. Select the **Option Type** dropdown to define its **Value**. **Detailed Description** may display additional information on the option type. Set the **Sequence** to **1** unless the option can have more than one value. **Owner** indicates if this option is owned by the base package or by your implementation (**Customer Modification**).

---

**NOTE:** You can add new options types. Your implementation may want to add additional option types. For example, your implementation may have plug-in driven logic that would benefit from a new option. To do that, add your new values to the customizable lookup field **BUS\_OBJ\_OPT\_FLG**. If you add a new option type for a business option, you must update its maintenance object to declare this new option type. Otherwise, it won't appear on the option type dropdown. You do that by referencing the new option type as a **Valid BO Option Type** [maintenance object option](#).

---

### Where Used

Follow this link to open the data dictionary to view the tables that reference [FI\\_BUS\\_OBJ](#).

## Business Object - Schema

Use this page to maintain a business object's schema. Open this page using **Admin > System > Business Object** and then navigate to the **Schema** tab.

### Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the business object.

Click the **View Schema** hyperlink to view the business object's expanded schema definition. Doing this opens the [schema viewer](#) window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

The [Schema Designer](#) zone allows you to edit the business object's schema. The purpose of the schema is to describe the business object's properties and map them to corresponding maintenance object fields.

---

**NOTE: Generating a Schema** A context sensitive "Generate Schema" zone is associated with this page. The zone provides a button that allows the user to generate a basic schema that includes all the fields for all the tables for the BO's maintenance object.

---

**NOTE: Schema Definition Tips.** A context sensitive "Schema Tips" zone is associated with this page. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

---

**NOTE: View UI Rendering.** A context sensitive "View UI Rendering" zone is associated with this page. The zone is useful for [business objects that define the user interface detail](#) using schema attributes and UI Hints. The buttons allow you to view the automatically rendered display and input maps.

---

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

## Business Object - Algorithms

Use this page to maintain a business object's algorithms. Open this page using **Admin > System > Business Object** and then navigate to the **Algorithms** tab.

### Description of Page

The **Algorithms** grid contains algorithms that control important functions for entities defined by this business object. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
- If the algorithm is implemented as a script, a link to the **Script** is provided. Refer to [Plug-In Scripts](#) for more information.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).



The following table describes each **System Event**. Refer to [A Business Object May Define Business Rules](#) for more information about these system events.

<b>System Event</b>	<b>Optional / Required</b>	<b>Description</b>
<b>Audit</b>	Optional	<p>Algorithms of this type may be used to audit certain changes made to business object instances.</p> <p>The system hands over to the algorithms a summary of all the elements that were changed throughout a specific call to update an object. Excluded from this processing are elements explicitly marked on the schema as requiring no audit. For each element its original value before the change as well as its new value are provided.</p> <p>It is the responsibility of the algorithms to record corresponding audit information.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to <a href="#">Business Object inheritance</a> for more information.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>Information</b>	Optional	<p>We use the term "Business Object Information" to describe the basic information that appears throughout the system to describe an entity defined by the business object. The data that appears in this information description is constructed using this algorithm.</p> <p>The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number found on the business object closest to the current business object in the inheritance hierarchy. Refer to <a href="#">Business Object inheritance</a> for more information.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>Post-Processing</b>	Optional	<p>Algorithms of this type may be used to perform additional business logic after a business object instance has been processed.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to <a href="#">Business Object inheritance</a> for more information.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>Pre-Processing</b>	Optional	<p>Algorithms of this type further populates a request to maintain a business object instance right before it is processed.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to <a href="#">Business Object inheritance</a> for more information.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>Validation</b>	Optional	<p>Algorithms of this type may be used to validate a business object instance when added, updated or deleted.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to <a href="#">Business Object inheritance</a> for more information.</p>

System Event	Optional / Required	Description
		Click <a href="#">here</a> to see the algorithm types available for this system event.

**NOTE: Generate Algorithm.** A context sensitive "Generate a BO Algorithm" zone is associated with this page. Refer to [Defining Business Objects](#) for more information about this zone.

**NOTE: You can add new system events.** Your implementation may want to add additional business object oriented system events. For example, your implementation may have plug-in driven logic that would benefit from a new system event. To do that, add your new values to the customizable lookup field **BO\_SEVT\_FLG**. If you add a new business object system event, you must update the maintenance object to declare this new system event. Otherwise, it won't appear on the system event dropdown. You do that by referencing the new system event as a **Valid BO System Event** [maintenance object option](#).

**NOTE: You can inactivate algorithms on base Business Objects.** Your implementation may want to use a business object provided by the base product, but may want to inactivate one or more algorithms provided by the base business object. To do that, on the business object where this algorithm is referenced, go to the options grid on Business Object - Main and add a new option, setting the option type to **Inactive Algorithm** and setting the option value to the algorithm code.

## Business Object - Lifecycle

Use this page to maintain a business object's lifecycle oriented business rules and options. Open this page using **Admin > System > Business Object** and then navigate to the **Lifecycle** tab.

### Description of Page

The **Status** accordion contains an entry for every status in the object's [lifecycle](#). The entry appears differently for a child business object as it can only extend its inherited lifecycle by introducing algorithms and options of its own.

Use **Status** to define the unique identifier of the status. This is NOT the status's description, it is simply the unique identifier used by the system. Only the highest-level business object can define lifecycle statuses. For a child business object the inherited status description is displayed allowing navigation to the corresponding entry on the business object defining the lifecycle.

Use **Description** to define the label of the status. This field is hidden for a child business object.

Use **Access Mode** to define the action associated with this status. Refer to [Access Rights](#) for the details of how to use this field to restrict which users can transition a business entity into this state. This field is hidden for a child business object.

Enter a **Monitor Process** to defer the monitoring of entities in this state until the specific batch process runs. Refer to [Monitor Rules](#) for more information. This field is hidden for a child business object.

The **Status Reason** dropdown indicates if users should be prompted to provide a specific reason when the business object enters this state. This field appears only if the Status Reason Field is configured as an option on the business object's maintenance object. Valid values are blank, **Optional**, and **Required**. The default value is blank (users are not prompted to provide a status reason). See [Configuring Status Reasons](#) for more information about status reasons.

Use **Status Condition** to define if this status is an **Initial**, **Interim** or **Final** state. Refer to [One Initial State and Multiple Final States](#) for more information about how this field is used. This field is hidden for a child business object.

Use **Transitory State** to indicate whether a business entity should ever exist in this state. Only **Initial** or **Interim** states can have a transitory state value of **No**. Refer to [transitory states](#) for more information. This field is hidden for a child business object.

Use **Alert Flag** to indicate that being in this state warrants an application alert. This may be used by custom logic to provide an alert to a user that entities exist in this state. This field is hidden for a child business object.

Use **Display Sequence** to define the relative order of this status for display purposes. For example when displayed on the status accordion and on the summary tab page. This field is hidden for a child business object.

## Algorithms

The **Algorithms** grid contains algorithms that control important functions for a given status. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
- If the algorithm is implemented as a script, a link to the **Script** is provided. Refer to [Plug-In Scripts](#) for more information.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

The following table describes each **System Event**.

System Event	Optional / Required	Description
Enter	Optional	<p>Algorithms of this type apply business rules when a business object instance enters a given state.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to <a href="#">Business Object Inheritance</a> for more information.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
Exit	Optional	<p>Algorithms of this type apply business rules when a business object instance exits a given state.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to <a href="#">Business Object Inheritance</a> for more information.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
Monitor	Optional	<p>Algorithms of this type monitor a business object instance while in a given state. Typically these are used to auto-transition it to another state.</p> <p>The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to <a href="#">Business Object Inheritance</a> for more information.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>

**NOTE: Generate Algorithm.** A context sensitive "Generate a BO Algorithm" zone is associated with this page. Refer to [Defining Business Objects](#) for more information about this zone.

**NOTE: You can inactivate status level algorithms on base Business Objects.** Your implementation may want to use a business object provided by the base product, but may want to inactivate one or more of the status oriented algorithms provided by the base business object. To do that, on the business object and status where this algorithm is referenced, go to the options grid and add a new option, setting the option type to **Inactive Algorithm** and setting the option value to the algorithm code.

## Next Statuses

Use the **Next Statuses** grid to define the valid statuses a business entity can transition into while it's in this state. This section is hidden for a child business object. Refer to [Valid States versus State Transition Rules](#) for more information. Please note the following about this grid:

- **Status** shows the statuses for the top-level business object, the **Status Code**, the **Lifecycle BO description**, and the **Status description** for each status.
- Use **Action Label** to indicate the verbiage to display on the button used to transition to this status.
- **Sequence** controls the relative order of one status compared to others for display purposes. This information may be used to control the order in which buttons are presented on a user interface.
- **Default** controls which next state (if any) is the default one. This information may be used by an **Enter** or **Monitor** algorithm to determine an auto-transition to the default state. It may also be used to also mark the associated button as the default one on a user interface.
- **Transition Condition** may be configured to identify a common transition path from the current state. By associating a given "next status" with a transition condition value, you can design your auto-transition rules to utilize those flag values without specifying a status particular to a given business object. Thus, similar logic may be used across a range of business objects to transition a business entity into, for example, the next **Ok** state for its current state. You'll need to add your values to the customizable lookup field **BO\_TR\_COND\_FLG**.
- **Transition Role** controls whether only the **System** or both **System and User** have the ability to transition a business entity into a given "next status".
- When you initially set up a business object lifecycle, none of the statuses will reside on the database and therefore you can't use the search to define a "next status". We recommend working as follows to facilitate the definition of this information:
  - Leave the Next Statuses grid blank when you initially define a business object's statuses
  - After all statuses have been saved on the database, update each status to define its Next Statuses (this way, you can use the search to select the status).

## Options

The options grid allows you to configure the business object status to support extensible options. Select the **Option Type** drop-down to define its **Value**. **Detailed Description** may display additional information on the option type. Set the **Sequence** to **1** unless the option can have more than one value. **Owner** indicates if this option is owned by the base package or by your implementation (**Customer Modification**).

---

**NOTE: You can add new options types.** Your implementation may want to add additional option types. For example, your implementation may have plug-in driven logic that would benefit from a new option. To do that, add your new values to the customizable lookup field **BO\_OPT\_FLG**. If you add a new option type for a status, you must update the business object's maintenance object to declare this new option type. Otherwise, it won't appear on the option type dropdown. You do that by referencing the new option type as a **Valid BO Status Option Type** [maintenance object option](#).

---

## Business Object - Summary

This page summarizes business object information in a high level. Open this page using **Admin > System > Business Object > Search** and then navigate to the **Summary** tab.

### Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the business object.

Click the **View Schema** hyperlink to view the business object's expanded schema definition. Doing this opens the [schema viewer](#) window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

The **Business Object Hierarchy** zone displays in a tree view format the [hierarchy](#) of child business object associated with the current business object. It also shows the current business object's immediate parent business object.

For business objects with a lifecycle, the **Lifecycle Display** zone shows a graphical depiction of the lifecycle. Refer to the inline help of that zone for more information.

The **Options** zone summarizes business object and state specific options throughout the [inheritance](#) chain.

The **Rules** zone summarizes business object and state specific rules throughout the [inheritance](#) chain.

## Defining Status Reasons

Status Reasons are used to provide more information about why a business object transitioned to a given state. The status reason table provides a centralized place where status reasons can be defined across many different business objects and states.

---

**NOTE:** Refer to [Defining Reasons for Entering a State](#) for overview information.

---

If a business object has one or more states that are configured to capture a status reason, you may configure the valid reasons by navigating to the status reason portal using **Admin > System > Status Reason**.

The topics in this section describe the base-package zones that appear on the Status Reason portal.

### Business Objects with Status Reason List

This zone displays the business objects that have one or more status values that allow status reasons to be defined.

Click the broadcast icon to open other zones that contain more information about the business object's status reasons.

### Status Reasons

The **Status Reasons** zone contains a list of the existing status reasons for the broadcasted business object.

## Business Services

A business service is used to expose a back-end service so that it may be invoked by a script or a zone or a map to retrieve information or perform functions, depending on the related service.

As with the business object, the business service's interface to the internal service is defined using its schema. The schema maps the business service's elements to the corresponding elements in the internal service program's XML document. Just as a business object can simplify the schema of its maintenance object by only defining elements that it needs and "flattening" entries in a child collection to be defined as a singular element, a business service schema may simplify its service XML in a similar way.

---

**NOTE: Schema Definition Tips.** A context sensitive "Schema Tips" zone appears when you open the [Business Service](#) page to assist you with the schema definition syntax. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

---

[Inbound web services](#) and [scripts](#) support interaction with business services. You can also invoke a business service from a Java class.

## Service Program

This transaction defines services available in the system. These include user interface services as well as stand-alone services that perform a specific function. A service may be referenced by a business service. Use this transaction to view existing service and introduce a new stand-alone service to be made available to a Business Service.

Select **Admin > System > Service Program** to maintain service programs.

### Description of Page

**Service Name** is the unique identifier of the service.

---

**CAUTION:** Important! When adding new service programs, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

**Owner** indicates if this service is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a service. This information is display-only.

**Description** describes the service.

**Service Type** indicates whether the service is a **Java Based Service** or a **Java (Converted) Service**.

This **Program Component** grid shows the list of program user interface components associated with the service. For a stand-alone service, this list is typically not applicable.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_MD\\_SVC](#).

## Defining Business Services

The topics in this section describe how to maintain business services.

Note that several context sensitive dashboard zones appear on this page and are visible on all tabs.

- **Schema Tips.** This zone provides several links to view details related to the valid schema syntax.
- **Generate Schema.** This zone includes a button that can be used to generate the schema based on the XML of its related Service. Once generated, adjust the schema as desired.

## Business Service - Main

Use this page to define basic information about a Business Service. Open this page using **Admin > System > Business Service**

### Description of Page

Enter a unique **Business Service** name and **Description**. Use the **Detailed Description** to describe the purpose of this business service in detail. **Owner** indicates if the business service is owned by the base package or by your implementation (**Customer Modification**).

---

**CAUTION:** Important! If you introduce a new business service, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

Enter the internal **Service Name** being called when this business service is invoked.

Enter the **Application Service** that is used to provide security for the business service. The application service must have an Access Mode of Execute.

Click the **View Schema** to view the business service's expanded schema definition. Doing this opens the [schema viewer](#) window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

Click the **View XML** hyperlink to view the XML document used to pass data to and from the service in the [Service XML Viewer](#). You may find it useful to leave the application viewer window open while defining your business service schema.

---

**NOTE: XML document may not be viewable.** If you create a new service program and do not regenerate the application viewer, you will not be able to view its XML document.

---

### Where Used

Follow this link to open the data dictionary to view the tables that reference [FI\\_BUS\\_SVC](#).

## Business Service - Schema

Use this page to maintain a Business Service's schema and to see where the Business Service is used in the system. Open this page using **Admin > System > Business Service** and then navigate to the **Schema** tab.

### Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the business service.

The [Schema Designer](#) zone allows you to edit the business service's schema. The purpose of the schema is to map the business service's elements to the corresponding fields of the back-end service program it rides on.

---

**NOTE: Generating a Schema** A context sensitive "Generate Schema" zone is associated with this page. The zone provides a button that allows the user to generate a basic schema that includes all the elements that are found in the XML of the BS's service.

---

---

**NOTE: Schema Definition Tips.** A context sensitive "Schema Tips" zone is associated with this page. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

---

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

## User Interface (UI) Maps

The User Interface (UI) map holds HTML to be rendered within [portal zones](#) and [Business Process Assistant \(BPA\) scripts](#). UI maps allow your implementation to create input forms and output maps that closely match your customer's business practices. In other words, the UI Map is designed to facilitate the capture and display of your [business objects](#) and [business services](#).

The UI map is a repository for a single HTML document paired with an XML schema where the schema defines the data that the HTML document displays and/or modifies. The UI Map HTML gives you the ability to craft the display by any method that an html document can support, including JavaScript and full CSS functionality.

Configuration tool support for UI Maps hinges around the ability to inject and extract an XML document from the HTML. For more information on the specialized support for HTML and JavaScript functionality - please refer to the HTML tips document (more on this below).



```

<html>
<head>
<title>Output Personal Information</title>
<link rel="stylesheet" type="text/css" href="cm_templates/cmStyles.css"/>
</head>
<body>

<table width="100%" border="0" cellpadding="2" style="margin-top:15px;" >

  <tr>
    <td/>
    <td/> <!-- locate the edit button on the bottom of the third column -->
    <td rowspan="99" style="vertical-align:bottom; margin-left:5px">
      <input type="button" value="edit" onClick="oraRunScript('HumanInfoU','personId');"/>
    </td>
  </tr>

  <tr>
    <td class="outputLabel">Name:</td>
    <td><span oraField="name" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Home Phone:</td>
    <td><span oraField="homePhone" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Business Phone:</td>
    <td><span oraField="businessPhone" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Cell Phone:</td>
    <td><span oraField="cellPhone" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">FAX:</td>
    <td><span oraField="fax" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Social Security:</td>
    <td><span oraField="socialSecurity" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Drivers License:</td>
    <td><span oraField="driversLicense" class="outputData"></span></td>
  </tr>
  <tr>
    <td class="outputLabel">Email:</td>
    <td><span oraField="email" class="outputData"></span></td>
  </tr>

</table>

</body>

<xml>
<root>
  <name>Greer, Johan</name>
  <email>jurgen.greer@media.com</email>
  <socialSecurity>939-30-3939</socialSecurity>
  <driversLicense>C8392020</driversLicense>
  <homePhone>(838) 030-0303</homePhone>
  <cellPhone>(444) 444-4040</cellPhone>
  <businessPhone>(737) 393-3838</businessPhone>
  <fax>(373) 939-3939</fax>
  <personId>1239997654</personId>
</root>
</xml>
</html>

```

Figure 1: HTML to Display Customer Business Object

Name:	<b>Greer, Johan</b>	
Home Phone:	<b>(838) 030-0303</b>	
Business Phone:	<b>(737) 393-3838</b>	
Cell Phone:	<b>(444) 444-4040</b>	
FAX:	<b>(373) 939-3939</b>	
Social Security:	<b>939-30-3939</b>	
Drivers License:	<b>C8392020</b>	
Email:	<b>jurgen.greer@media.com</b>	<input type="button" value="edit"/>

Figure 2: Customer HTML Rendered (Output Data for Zone)



UI maps are typically crafted as output tables when used in conjunction with portal zones - please refer to [Map Zones](#) for more information. When referenced within [BPA scripts](#), UI maps are typically crafted as forms for the capture and update of data.

**Edit Personal Information**

[err](#)

**Name:**   
Last-name suffix, prefix first-name middle-name/initial

**Home Phone:**   -

**Business Phone:**   -

**Cell Phone:**   -

**FAX:**   -

**Social Security:**  -  -

**Drivers License:**

**Email:**

Figure 3: HTML Input Form Rendered (for BPA Script)

Portal zones can reference a UI map for the zone header. They may also utilize a UI map to define their filter area. This type of UI map is not a complete HTML document, but is instead configured as a UI Map "fragment".

---

**NOTE:** UI Map Tips. A context sensitive "UI Map Tips" zone appears when you open the [UI Map](#) page to assist you with the HTML document and schema definition syntax. The "tips" describe methods to take advantage of the configuration tools, however, they are not an HTML reference.

---

**NOTE:** Editing HTML. You can use a variety of HTML editors to compose your HTML, which you can then cut, and paste into your UI map. In addition, the zone provides a complete list of the XML schema nodes and attributes available to you when you construct the map's data schema.

---

## Defining UI Maps

The topics in this section describe how to maintain UI Maps.

### UI Map - Main

Use this page to define basic information about a user interface (UI) Map. Open this page using **Admin > System > UI Map**

#### Description of Page

Enter a unique **Map** name. **Owner** indicates if the UI map is owned by the base package or by your implementation (**Customer Modification**).

---

**CAUTION:** Important! If you introduce a new UI map, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

Use **UI Map Type** to indicate whether the map is a **Complete HTML Document** or an **HTML Fragment**. Portal zones can reference a UI map to describe a fragment of their HTML, for example the zone header or filter area. In this case the UI map is not a complete HTML document, but is instead configured as a UI Map "fragment".

---

**NOTE:** A context sensitive "UI Map Tips" zone is associated with this page to assist you with the HTML document definition syntax. Refer to the tips zone for more information on how to use fragment UI maps to construct portal zone HTML components.

---

Enter a **Description**. Use the **Detailed Description** to describe how this map is used in detail.

Click on the **View Schema** to view the UI map's expanded schema definition. Doing this opens the [schema viewer](#) window.

Use the **Test UI Map** hyperlink to render your html in a test window.

---

**NOTE:** The **Test UI Map** hyperlink also exercises the proprietary functionality that binds an xml element with an html element so you can get immediate feedback on your html syntax.

---

### Where Used

Follow this link to open the data dictionary to view the tables that reference [F1\\_MAP](#).

## UI Map - Schema

Use this page to maintain a UI Map's HTML and schema and to see where the UI Map is used in the system. Open this page using **Admin > System > UI Map** and then navigate to the **Schema** tab.

### Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the UI Map.

The **HTML Editor** zone allows you to edit the HTML document of the map.

---

**NOTE: HTML Definition Tips.** A context sensitive "UI Map Tips" zone is associated with this page to assist you with the HTML document definition syntax. The "tips" describe good ways to produce simple HTML, however, they are not an HTML reference. Note that you can use a variety of HTML editors to compose your HTML, which you can then cut and paste into your UI map.

---

**NOTE: Providing Help.** A [tool tip](#) can be used to display additional help information to the user. This applies to section elements as well as individual elements on a map. Refer to the tips context sensitive zone for more information on how to enable and provide UI map help.

---

The [Schema Designer](#) zone allows you to edit the data schema part of the map. The purpose of the schema is to describe the data elements being displayed by the map.

---

**NOTE: Schema Definition Tips.** The same "UI Map Tips" zone above also provides a complete list of the XML nodes and attributes available to you when you construct the map's data schema.

---

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

## Ensuring Unique Element IDs for UI Maps

The following describes how to modify JavaScript code to ensure the proper rendering of unique element IDs for UI Maps. The modification is required only for code that renders HTML using a `getElementById()` (or similar) function to generate list IDs and avoid account verification or related errors.

The following sample snippet contains the necessary modifications:

```
...
function getElementsFromList(namePrefix) {
    var ret = [];
    var elements = document.getElementsByTagName("INPUT");
    for(var i=0;i<elements.length;i++) {
        var elemID = elements[i].id;
        if((id) && (id.startsWith(namePrefix + '_'))) {
            ret.push(elements[i]);
        }
    }
    ...
return ret;
}
```

Since IDs aren't necessarily unique in generated UI Map IDs, the code shown above ensures uniqueness at runtime by appending an underscore and row number (e.g., `myField_1`, `myField_2`) for proper handling by Framework in the rendered HTML, while still allowing you to reference the unmodified IDs contained in the generated UI Map.

A switch in the `spl.properties` file also permits you to disable the generation of unique IDs for elements in a grid (as described below), though, for standards compliance reasons, it is highly recommended that this switch be left at its default value.

```
Property Name: spl.runtime.compatibility.uiMapDisableGenerateUniqueHtmlIDs
File Name: spl.properties (under web project in FW)
Default Value: false
Accepted Values: true or false
Description: This property controls the generation of unique IDs for all input elements inside a list. When t
compliant.
```

## Maintaining Managed Content

The Managed Content maintenance object is used to store content such as XSL files used to create vector charts, JavaScript include files, and CSS files. These files may then be maintained in the same manner as the HTML in UI Maps.

The topics in this section describe the Managed Content portal.

### Managed Content - Main

This page is used to define basic information about the content. Open this page using **Admin > System > Managed Content**.

#### Description of Page

Enter a unique name for the content in the **Managed Content** field.

**Owner** indicates if the content is owned by the base package or by your implementation.

Use **Managed Content Type** to indicate the type of content, for example, XSLT or JavaScript.

Enter a **Description**.

Use the **Detailed Description** to describe in detail how this map is used.

## Managed Content - Schema

This page is used to create and maintain the managed content. Open this page using **Admin > System > Managed Content** and then navigate to the **Schema** tab.

### Description of Page

The General Information zone displays the main attributes of the content. The Editor zone allows you to edit the content.

## Data Areas

The data area has no business purpose other than to provide a common schema location for re-used schema structures. It exists solely to help eliminate redundant element declaration. For example, if you have multiple schemas that share a common structure, you can set up a stand-alone data area schema for the common elements and then include it in each of the other schemas.

Be aware that a stand-alone data area can hold elements that are mapped to true fields. For example, you might have 50 different types of field activities and all might share a common set of elements to identify where and when the activity should take place. It would be wise to declare the elements that are common for all in a stand-alone data area and then include it in the 50 field activity business objects.

It's strongly recommended that you take advantage of stand-alone data areas to avoid redundant data definition!

---

**CAUTION: Dynamic inclusion!** When the system renders a schema, all schemas included within it are expanded real-time. This means that any change you make to a data area will take effect immediately within all schemas it is referenced within.

---

**NOTE: Schema Definition Tips.** A context sensitive "Schema Tips" zone appears when you open the [Data Area](#) page to assist you with the schema definition syntax. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

---

Data areas may be included in a business object that does not define a full UI map for display or input. Rather, it is using auto-rendering by defining UI attributes in its schema and via UI hints.

---

**NOTE: View UI Rendering.** A context sensitive "View UI Rendering" zone appears on this page. It may be used for a data area that is part of a business object that is using auto-rendering for the display and input maps. The buttons allow you to view the rendered UI for the segment of the schema that is defined by the data area.

---

## Defining Data Areas

The topics in this section describe how to maintain Data Areas.

### Data Area - Main

Use this page to define basic information about a data area. Open this page using **Admin > System > Data Area**.

#### Description of Page

Enter a unique **Data Area** name and **Description**. Use the **Detailed Description** to describe what this data area defines in detail. **Owner** indicates if the data area is owned by the base package or by your implementation (**Customer Modification**).

---

**CAUTION:** Important! If you introduce a new data area, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

Click the **View Schema** to view the data area's expanded schema definition. Doing this opens the [schema viewer](#) window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

To extend another data area, reference that data area in the **Extended Data Area** field. By extending a data area you can add additional elements to a base product data area.

Here's an example of an extended data area:

- The product releases with data area A, which contains elements a, b, and c.
- Your implementation creates data area CM-A, which contains element z, and references data area A as the extended data area.
- At run time, everywhere data area A is included it will contain elements a, b, c, and z.

#### Where Used

Follow this link to open the data dictionary to view the tables that reference [FI\\_DATA\\_AREA](#).

## Data Area - Schema

Use this page to maintain a Data Area's schema and to see where the data area is used in the system. Open this page using **Admin > System > Data Area** and then navigate to the **Schema** tab.

#### Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays the main attributes of the data area.

The [Schema Designer](#) zone allows you to edit the data area's schema. The purpose of the schema is to describe the structure and elements of the data area.

---

**NOTE: Schema Definition Tips.** A context sensitive "Schema Tips" zone is associated with this page. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

---

---

**NOTE: View UI Rendering.** A context sensitive "View UI Rendering" zone is associated with this page. The zone is useful for data areas that are to be included in [business objects that define the user interface detail](#) using schema attributes and UI Hints. The buttons allow you to view the automatically rendered display and input maps.

---

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

## Schema Designer

The Schema Designer is a user-friendly interface for performing the following common schema editing tasks:

- Displaying existing schemas.
- Creating schema elements.
- Moving elements within a schema.
- Adding attribute values.

The designer provides the ability to toggle between **Design View** and **Source View** views by clicking the icons on the **Schema Designer** title bar:



The **Source View** shows the schema elements and their attributes written in the proper syntax.


In the graphical **Design View** mode, the zone is split into two areas. The left pane displays the elements of the schema in a tree-like presentation, while the right pane displays all valid attributes and values for selected schema elements.

The following points highlight some functionality available in the design view.

- If the schema definition refers to another schema using an “include” statement, one can expand that schema in the left panel to view the elements within that schema. Clicking an element in that expanded view shows information about the element’s definition in the right panel. Changes cannot be made to elements in this view.
- To move an existing element within the schema in the left panel, simply drag and drop.
- Right clicking an element in the left panel causes a pop-up to display which offers options based on the position of the element clicked on and based on the type of element.
  - **Add element above | below**
  - **Delete this element**
  - **Move this element up | down**
  - **Add child element**

When adding a new element, you are prompted for the element type. The following lists the possible element types. Most are self explanatory and represent standard schema options.

- Comment. This adds a comment to the schema.
- Embedded HTML. This is specific to a schema enabled for UI Hints. It is used to include a UI map fragment.
- Field
- Flattened Field
- Flattened List
- Group
- Include BO Schema
- Include BS Schema
- Include DA Schema
- Include Map Schema
- Include SS Schema
- Input Map Title. This is specific to a schema enabled for UI Hints. It is used to define a title element for the map.
- List
- Nested Flattened Field. This is a flattened field from a child table.
- Raw Element. This element is used to capture text as is. It is typically used to capture an XML structure without any details of the definition of the individual nodes.
- Section. This is specific to a schema enabled for UI Hints. It is used to define a section within the map.
- Simple Field. This is a special type of Field element that is used to define an element that is mapped to an XML column.

Context-sensitive embedded help is provided for fields and controls in the right pane by clicking the Help icon 

You can also use links in the **Schema Tips** zone in the dashboard area to open a navigable Help window containing schema-related topics and tips. To search topic content, press `Ctrl+F` when the Schema Tips Help window has focus.

The **Schema Designer** is available by choosing the **Schema** tab on the [Business Object](#), [Data Area](#), [UI Map](#), [Business Service](#), and [Script](#) pages.

## Schema Viewer

The schema viewer shows a tree-view presentation of a schema in its expanded form.

The schema illustrates the structure to be used when communicating with the schema's associated object. The following takes place when a schema is expanded:

- If the schema definition includes references to other schemas, these references are replaced with the corresponding schema definitions.
- Also, if the schema definition contains **private** elements, they are omitted from this view.

Clicking on any node on the tree populates the text box on the top with the node's absolute XPath expression. You will find this feature very useful when writing scripts interacting with schema-based objects. [Scripting](#) often involves referencing elements in a schema-based XML document using their absolute XPath expression. You can use this feature on the schema viewer to obtain the XPath expression for an element and copy it over to your script.

## Business Event Log

Business Event Log may be viewed as a tool designed to capture any type of business event worth noting. You configure business objects to represent the various types of events your application calls for. The following type of details may be captured for each event:

- The business object representing the type of event.
- The date and time the event took place and who initiated it.
- The business entity for which this event is logged.
- Standard application message to describe the event.
- Additional context information that is available at the time of the event and varies for each type of event. The Business Event Log maintenance object supports a standard characteristics collection as well as an XML storage (CLOB) field. The event's business object determines where each piece of information resides. Refer to [Business Objects](#) for more information.

One common type of event may be the audit of changes made to sensitive data, for example, tracking an address change. Whenever an entity associated with a business object is added, changed, or deleted the system summarizes the list of changes that took place in that transaction and hands them over to **Audit** business object algorithms to process. You may design such an algorithm to audit the changes as business event logs. Refer to [a business object may define business rules](#) for more information.

You can also allow users to initiate business event logs to capture important notes about a business entity by exposing a [BPA Script](#) to invoke the event's corresponding business object.

Bottom line is that any process can create a business event log by invoking the business object representing the appropriate type of event.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [FI\\_BUS\\_EVT\\_LOG](#).

# To Do Lists

---

Certain events that occur within the system will trigger messages describing work that requires attention. For example, if a bill segment has an error, the system generates a To Do message to alert the person responsible for correcting such errors.

Each type of message represents a To Do list. For example, there are To Do lists for bill segment errors, payment errors, customer contact reminder, etc.

We refer to each message as a **To Do Entry**. Each To Do entry is assigned a specific **To Do Role**. The role defines the users who may work on the entry. A To Do entry has a **To Do log** that maintains record of the progress on the To Do entry. For example, the To Do log indicates when the To Do entry was created, when it was assigned to a user and to whom it was assigned, and when and by whom it was completed.

---

**FASTPATH:** Refer to [To Do Processing](#) for a description of end-user queries and tools assisting in reviewing, assigning and processing To Do entries.

---

## The Big Picture of To Do Lists

The topics below provide more information about To Do configuration.

## To Do Entries Reference A To Do Type

Every [To Do entry](#) references a To Do type. The To Do type controls the following functions:

- The To Do list on which the entry appears.
- The page into which a user is taken when they drill down on an entry.
- The message that appears in the user's To Do list. Note this message can be overridden for specific To Do messages by specifying a different message number in the process that creates the specific To Do entry. For example, the process that creates To Do entries associated with bill segments that are in error displays the error message rather than a generic "bill segment is in error" message.
- The To Do list's sort options. Sort options may be used on the To Do list page to sort the entries in a different order. For example, when a user looks at the bill segment error To Do list, they have the option of sorting it in error number order, account name order, or in customer class order. Note the default sort order is also defined on To Do type.
- Whether (and how) the To Do entry is downloaded to an external system (e.g., an email system).
- The roles to which an entry may be reassigned.
- The default priority of the To Do list in respect of other To Do lists.
- The To Do list's usage, which indicates whether a To Do of that type may be created manually by a user.
- The algorithms used to perform To Do list specific business rules.
- The characteristics applicable to the To Do list.

## To Do Entries Reference A Role

Every [To Do entry](#) references a role. The role defines the users who may be assigned to **Open** entries.

The permissible roles that may be assigned to a To Do entry are defined on the entry's To Do type. After an entry is created, its role may be changed to any role defined as valid for the entry's To Do type.



An entry's initial role is assigned by the background process or algorithm that creates the entry. Because you can create your own processes and algorithms, there are an infinite number of ways to default an entry's role. However, the base package processes and algorithms use the following mechanisms to default an entry's role:

- The system checks if an entry's message category / number is suppressed (i.e., not created). If so, the entry is not created. Refer to [To Do Entries Can Be Rerouted Or Suppressed Based On Message Number](#) for more information.
- The system checks if an entry's message category / number is rerouted to a specific role. If so, it defaults this role. Refer to [To Do Entries Can Be Rerouted Or Suppressed Based On Message Number](#) for more information.
- Your specific product may introduce additional criteria for assigning a role, for example perhaps important accounts are assigned to a kind of account management group and the account management group includes configuration for special role for certain To Do types. Refer to [Set Additional Information Before a To Do is Created](#) for more information.
- If a Role wasn't determined in one of the previous steps and a Role is provided by the initiating process, the entry is created with that Role.
- If the entry does not have a role after the above takes place, the entry's To Do type's default role is assigned to the entry.

---

**NOTE:**

At installation time, the system provides a default role assigned to the system To Do types when first installed called **F1\_DFLT**. This is done to allow testing of the system prior to implementing of appropriate To Do roles for your organization. The recommendation is to configure all the To Do Types with appropriate business oriented To Do roles once they are defined.

---

**CAUTION:** Important! Most organizations have the notion of a supervisor who is responsible for all entries assigned to a given role. It's important for this user (or users) to be part of all such roles. Refer to [To Do Supervisor Functions](#) for information about transactions that can be used by supervisors to review and assign work to users.

---

## To Do Entries Can Be Rerouted (Or Suppressed) Based On Message Number

Consider the To Do type used to highlight bill segments that are in error. To Do entries of this type reference the specific bill segment error number so that the error message can be shown when the Bill Segments in Error To Do list is displayed.

---

**NOTE: Message Category / Message Number.** Every error in the system has a unique message category / number combination. Refer to [The Big Picture of System Messages](#) for more information about message categories and numbers.

---

If you want specific types of errors to be routed to specific users, you can indicate such on the To Do type. For example, if certain bill segment errors are always resolved by a given rate specialist, you can indicate such on the To Do type. You do this by updating the [To Do type's message overrides](#). On this page you specify the message category / number of the error and indicate the To Do role of the user(s) who should work on such errors. Once the To Do type is updated, all new To Do entries of this type that reference the message number are routed to the desired role.

---

**NOTE: Reroute versus suppression.** Rather than reroute an entry to a specific role, you can indicate that an entry with a given message number should be suppressed (i.e., not created). You might want to do this if you have a large volume of certain types of errors and you don't want these to clutter your users' To Do lists.

---

Obviously, you would only reroute those To Do types that handle many different types of messages. In other words, if the To Do type already references a specific message category / number rerouting is not applicable.

We do not supply documentation of every possible message that can be handled by a given To Do type. The best way to build each To Do type's reroute list is during the pre-production period when you're testing the system. During this period, compile a list of the messages that should be routed to specific roles and add them to the To Do type.

Keep in mind that if a message number / category is not referenced on a To Do type's reroute information, the entry is routed as described under [To Do Entries Reference A Role](#).

---

**NOTE: Manually created To Do entries cannot be rerouted or suppressed.** The rerouting occurs as part of the batch process or algorithm processing when the To Do is created. The role or user to whom a manual To Do should be assigned is specified when the To Do is created online. A manually created To Do may also be forwarded to another user or role.

---

## The Priority Of A To Do Entry

Some To Do entries may be more urgent to resolve than others. A To Do entry is associated with a priority level representing its relative processing order compared to other entries.

Priority level is initially assigned as follows:

- If a **Calculate Priority** plug-in is defined on the To Do entry's type, the system calls it to determine the entry's priority. You may want to use this method if an entry's priority is based on context or time-based factors. For example, when priority takes into consideration account specific attributes. When applicable, you may design a process that triggers priority recalculation of To Do entries "at will". For example, when priority is reassessed periodically based on factors external to the To Do entry's information. Refer to [To Do Type](#) for more information on priority calculation algorithms.
- If a priority value has not been determined by a plug-in, the system defaults a To Do entry's initial priority to the value specified on its type.

A user may manually override a To Do entry's priority at any time. Notice that once a To Do entry's priority is overridden, **Calculate Priority** plug-ins are no longer called so as to not override the value explicitly set by the user.

---

**NOTE:** The system does not use priority values to control order of assignment nor processing of To Do entries. Priority is available to assist your organization with supporting a business practice that ensures higher priority issues are worked on first.

---

## Working On A To Do Entry

A user can drill down on a To Do entry. When a user drills down on an entry, the user is transferred to the transaction associated with the entry. For example, if a user drills down on a bill segment error entry, the user is taken to the Bill Segment - Main page. Obviously, the page to which the user is taken differs depending on the type of entry.

It is also possible to configure the To Do type to launch a [script](#) when a user drills down on an entry rather than taking the user to a transaction. The script would walk the user through the steps required to resolve the To Do entry. Refer to [Launching Scripts When A To Do Is Selected](#) for more information.

After finishing work on an entry, the user can mark it as **Complete**. Completed entries do not appear on the To Do list queries (but they are retained on the database for audit purposes). If the user cannot resolve the problem, the user can forward the To Do to another user.

## Launching Scripts When A To Do Is Selected

Users can complete many To Do entries without assistance. However, you can set up the system to launch a [script](#) when a user selects a To Do entry. For example, consider a To Do entry that highlights a bill that's in error due to an invalid mailing address. You can set up the system to execute a script when this To Do entry is selected by a user. This script might prompt the user to first correct the customer's default mailing address and then re-complete the bill.

A script is linked to a To Do type based on its message number using the [To Do type's message overrides](#). Refer to [Executing A Script When A To Do Is Selected](#) for more information.

## To Do Entries Have Logs

Each [To Do entry](#) has a To Do log that maintains a record of the To Do's progress in the system. For example, the To Do log indicates when the To Do entry was created, when it was assigned to a user and to whom it was assigned, and when and by whom it was completed. Users can view the log to see who assigned them a particular To Do and whether any work has already been done on the To Do.

A log entry is created for all actions that can be taken on a To Do entry. Log entries are created for the following events:

- A To Do entry is created (either by the system or by a user)
- A To Do entry is completed (either by the system or by a user)
- A user takes an open To Do entry
- A supervisor assigns a To Do entry
- A user forwards an entry to another user or role
- A user sends back a To Do to the user who forwarded it
- A user manually adds a log entry to record details about the To Do's progress
- A user manually overrides the To Do entry's priority

---

**FASTPATH:** For information about the contents of log entries for each of the events, refer to [Log Entry Events](#).

---

## How Are To Do Entries Created?

A To Do Entry may be created in the following ways:

- A [background process](#) can create To Do Entries.
- An [algorithm](#) can create entries of a given type. Because the use of algorithms is entirely dependent on how you configure the control tables, the number of types of such entries is indeterminate.
- A user can create entries of To Do types that have a **Manual** usage. Refer to [To Do Entries Created Manually](#) for information about setting up manual To Do types.

For any base product process that includes logic to create a To Do entry, the system supplies a sample To Do type that may be used. Although the To Do types provided by the product are system data, the following information related to each To Do type may be customized for an implementation and is not overwritten during an upgrade:

- The creation process. If the To Do is created by a background process where the background process is referenced on a To Do type. Refer to [To Do Entries Created By Background Processes](#) for more information.
- The routing process. Refer to [To Do Entries May Be Routed Out of the System](#) for more information.
- The priority. Refer to [To Do Type - Main](#) for more information.
- The [roles](#) that may be associated with the To Do type. Refer to [To Do Entries Reference a Role](#) for more information.
- The [message override](#) information. Refer to [To Do Entries Can Be Rerouted \(Or Suppressed\)](#) and [Launching Scripts When a To Do Is Selected](#) for more information.

## To Do Entries Created By Background Processes

There are different types of To Do entries created by background processes:

- To Do entries created by dedicated To Do background processes
- To Do entries created for object-specific errors detected in certain background processes

- To Do entries created based on a specific condition

## Dedicated To Do Background Processes

There are To Do entries that are created by system background processes whose main purpose is to create To Do entries based on a given condition. For these background processes, the To Do Type indicates the creation background process.

---

**NOTE: If you don't schedule the background process, the entries will NOT be created!** The To Do entries of this type will only be created if you have scheduled the associated background process. Therefore, if you want the system to produce a given entry, schedule the background process.

---

## To Dos Created for Object-Specific Error Conditions

A system background process may create a To Do entry when an error is detected during object-specific processing and it is not possible to create an exception record. (I.e., either no exception record exists for the process or the error is not related to the entity reported in the exception record.)

For these background processes, the To Do Type must reference the creation background process. To have the system create To Do entries for some or all of the errors generated by one of these processes, you must do the following:

- If you want the system to generate To Do entries for errors detected by one of the background processes below, go to the appropriate To Do type and populate the creation background process.
- If you want the system to generate To Do entries for some errors for the process, but not for all errors, populate the creation background process and then proceed to the [message overrides](#) tab to [suppress](#) certain messages. To this by indicating the message category and message number you want to suppress. Any error that is suppressed is written to the [batch run tree](#).

If you do not populate the creation background process, the errors are written to the [batch run tree](#).

---

**NOTE: Errors received while creating a To Do entry.** If the background process cannot successfully create a To Do entry to report an object-specific error, the error is written to the batch run tree along with the error encountered while attempting to create the To Do entry.

---

**NOTE: System errors are not included.** To Do entries are not created for a system error, for example an error related to validation of input parameter. These errors are written to the [batch run tree](#). Refer to [Processing Errors](#) for more information.

---

## To Dos Created by Background Processes for Specific Conditions

There are some system background processes that create a To Do entry when the process detects a specific condition that a user should investigate. For each background process, the To Do type is an input parameter to the process. The system provides To Do types for each base package background process that may create a To Do entry.

---

**NOTE: No Creation Process.** These To Do types do not need (and should not have) a **Creation Process** specified.

---

## To Do Entries Created By Algorithms

There are To Do entries that are created by algorithm types supplied with the base package. The system supplies a To Do Type for each of these To Do entries that you may use.

If you want to take advantage of these types of entries for system algorithm types, you must do the following:

- Create an [algorithm](#):

- This algorithm must reference the appropriate Algorithm Type.
- These algorithms have a parameter of the To Do Type to be created. You should specify the To Do Type indicated in the table.
- Plug the algorithm into the respective control table.

## To Do Entries Created Manually

You must set up manual To Do entry types if you want your users to be able to create To Do entries online. Users may create a manual To Do entry as a reminder to themselves to complete a task. Online To Do entries may also be used like electronic help tickets in the system. For example, if a user is having a problem starting service, the user can create a To Do that describes the problem. The To Do can be assigned to a help resolution group that could either resolve the problem or send the To Do back to the initiating user with information describing how to resolve the problem.

If you want to take advantage of manual To Do entries, create a To Do type and specify the following information.

### On the Main tab:

- Set the **To Do Type Usage** flag to **Manual**.
- Set the **Navigation Option** to **toDoEntryMaint** (To Do entry maintenance).
- Set the **Message Category** and **Message Number** to the message you want to be used for To Do entries of this type. The system will populate the message parameter with the Subject. To show only the subject in the To Do's message, use a message with "%1" as its text.

### On the Roles tab:

- Specify the [To Do roles](#) that may be assigned to To Do entries of this type.
- Indicate the To Do role that should be defaulted when you create To Do entries of this type.

### On the Sort Keys tab:

When a user adds a manual To Do entry, the system creates an entry with three sort key values. (Sort keys may be used on the To Do list page to sort the entries in a different order.) The To Do type should be set up to reference the sort keys as follows:

Sequence	Description
1	Created by user ID
2	Created by user name
3	Subject

We recommend that the keys have an **Ascending** sort order and that the Subject is made the default sort key.

**NOTE:** It is possible to define additional sort keys and use a To Do Post Processing algorithm to populate the values. In this case, the base sort keys defined above should still be defined.

### On the Drill Keys tab:

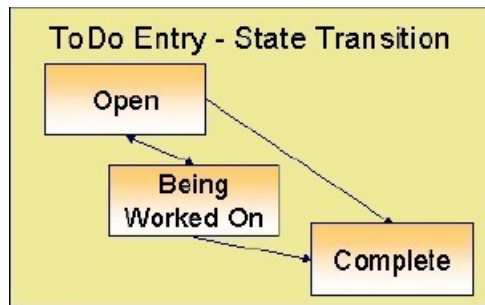
When a user adds a manual To Do entry, it is created with a drill key value equal to the To Do entry's ID. When the user clicks the Go To button next to the message in the To Do list, the system uses the drill down application service (defined on the main tab) and the drill key to display the associated To Do entry.

The To Do type must be set up with a drill key that reference the To Do entry table and the To Do entry ID:

Sequence	Table	Field
1	CI_TD_ENTRY	TD_ENTRY_ID

## The Lifecycle Of A To Do Entry

The following state transition diagram will be useful in understanding the lifecycle of a To Do entry.



A To Do entry is typically created in the **Open** state. Entries of this type can be viewed by all users belonging to the entry's role. Refer to [How Are To Do Entries Created?](#) for information about how entries are created.

An **Open** entry becomes **Being Worked On** when it is assigned to a specific user or when a user proactively assumes responsibility for the entry. While an entry is **Being Worked On**, it is visible on the To Do Summary page only by the user who is assigned to it.

---

**NOTE:** To Do entries may be created in the **Being Worked On** state. Some To Do background processes may create To Do entries in the **Being Worked On** state. When a user adds a To Do entry online and assigns the To Do to a user (as opposed to a role), the To Do entry is also created in the **Being Worked On** state.

---

A **Being Worked On** entry may be forwarded to a different user or role. If the entry is forwarded to a role, it becomes **Open** again.

When an entry becomes **Complete**, it is no longer visible in the To Do list queries (but it remains on the database for audit purposes). There are two ways an entry can become **Complete**:

- A user can manually indicate it is **Complete** (there are several ways to do this).
- For To Do entries that are logically associated with the state of some object, the system automatically marks the entry **Complete** when the object is no longer in the respective state. For example, an entry that's created when an account doesn't have a bill cycle is completed when the account has a bill cycle.

---

**CAUTION:** Important! The automatic completion of To Do entries occurs when the background process responsible for creating entries of a given type is executed. Therefore, if you only run these processes once per day, these entries remain **Being Worked On** even if the object is no longer in the respective state.

---

## Linking Additional Information To A To Do Entry

Additional information may be linked to a To Do entry using characteristics. For example, when creating a manual To Do entry, a user may define the account related to the To Do.

When creating an automatic To Do entry, the program that generates the To Do may link related data to the To Do using characteristics. Use system [algorithm](#) to link related entities. For manually created To Dos, the valid characteristic types that may be linked to the To Do entry must be defined on the [To Do type](#) for that To Do entry.

If your To Do entries reference characteristics that are related to your global context data, you may want to configure an [Alerts](#) to display an alert if a related entry is **Open** or **Being Worked On**.

## Implementing Additional To Do Entry Business Rules

If your business practice calls for additional validation rules or processing steps to take place after a To Do Entry is created or updated, you may want to take advantage of the **To Do Post Processing** plug-ins defined on [To Do type](#).

For example, you may want to validate that To Do entries are only assigned to users with the proper skill levels needed to resolve them. Refer to [F1-VAL-SKILL](#) for a sample algorithm handling such validation.

## To Do Entries May Be Routed Out Of The System

A To Do type can be configured so that its entries are interfaced to another system.

For example, a given To Do type can be configured to create an email message whenever a new To Do entry is created. The following points describe how to do this:

- Define the name of the background process responsible for interfacing the new To Do entries to another system on the respective To Do type. The base package contains a batch process called [F1-TDEER](#) that can be used for most situations. This batch process invokes the **External Routing algorithms** defined on each entry's To Do type.
- Plug in an appropriate **External Routing** algorithm on the respective To Do type. The logic in this type of algorithm performs the interface efforts for a specific To Do entry. For example, if an email message should be created for a To Do entry, the logic in the algorithm would compose and send the email message(s) for a specific To Do entry.

Click [here](#) to see the algorithm types available for this system event.

## Periodically Purging To Do Entries

**Completed** To Do entries should be periodically purged from the system by executing the [F1-TDPG](#) background process. This background process offers you the following choices:

- You can purge all To Do entries older than a given number of days.
- You can purge To Do entries for a specific list of To Do types that are older than a given number of days.
- You can purge all To Do entries except for a specific list of To Do types that are older than a given number of days.

We want to stress that there is no system constraint as to the number of **Completed** To Do entries that may exist. You can retain these entries for as long as you desire. However, you will eventually end up with a very large number of **Completed** entries and these entries will cause the various To Do background processes to degrade over time. Therefore, you should periodically purge **Completed** To Do entries as they exist only to satisfy auditing and reporting needs.

---

**NOTE: Different retention periods for different types of To Do entries.** Keep in mind that the purge program allows you to retain different types of entries for different periods of time.

---

## Setting Up To Do Options

The topics in this section describe how to set up To Do management options.

## Installation Options

The following section describes configuration setup on the installation options.



## To Do Information May Be Formatted By An Algorithm

A **To Do Information** algorithm may be plugged in on the [installation record](#) to format the standard To Do information that appears throughout the system. This algorithm may be further overridden by a corresponding plug-in on the [To Do Type](#).

## Set Additional Information Before A To Do Is Created

A **To Do Pre-creation** algorithm may be plugged in on the [installation record](#) to set additional information for a To Do entry before it is created. Algorithms of this type are used for two common purposes:

- Linking context specific data to the To Do entry using characteristics. For example, Oracle Utilities Customer Care and Billing provides an algorithm that attempts to link a related person, account, premise, service agreement or service point to a To Do entry based on its drill key value. Note, before you can set up this algorithm, you must define the characteristic types that you'll use to hold each of these entities. Also note that it is not necessary to define these characteristics as valid characteristic types on the To Do type.
- Overriding the Role of a To Do entry based on specific configuration related to the To Do's context data. For example, Oracle Utilities Customer Care and Billing provides an algorithm to determine a To Do role based on overrides related to the account's account management group or division.

## Alerts

If your To Do entries reference characteristics related to your global context data and your product supports dashboard alerts generated by algorithms, you may want configure an algorithm to display an alert if the entry is **Open** or **Being Worked On** for the data currently in context.

Refer to your product's documentation to determine if these types of alerts are supported.

## Next Assignment Algorithm

If your organization opts to use the next assignment feature supported by the Current To Do dashboard zone, you need to plug-in a **Next To Do Assignment** algorithm into the [installation options](#) to determine the next To Do entry the user should work on. Make sure you provide users with security access rights to the zone's next assignment action.

---

**FASTPATH:** Refer to the [Current To Do](#) zone for more information.

---

## Messages

You need only set up new messages if you use algorithms to create To Do entries or prefer different messages than those associated with the base package's To Do types.

## Feature Configuration

The base package is provided with a generic **Activity Queue Management Feature Configuration** type. You may want to set up a feature configuration of this type to define any To Do management related options supporting business rules specific to your organization.

For example, the base package provides the following plug-ins to demonstrate a business practice where To Do entries are only assigned to users with the proper skill levels to work on them.



- The base **To Do Post Processing** To Do Type algorithm [F1-VAL-SKILL](#) validates that a user has the necessary skill levels required to work on a given To Do entry.
- The base **Next To Do Assignment** installation options algorithm [F1-NEXT-ASSG](#) only assigns To Do entries to users that have the proper skills to work on them. This plug-in is only applicable if your organization practices [work distribution](#) "on demand."

You must set up such an **Activity Queue Management** feature configuration if you want to use any of the above base package plug-ins.

The following points describe the various **Option Types** provided with the base package:

- **Skill.** This option provides a reference to a skill category. For example, if you were using characteristics to represent skill categories then you should reference each characteristic type using this option.
- **Override Skill.** This option provides an override skill information reference for a specific message. For example, if you were using a To Do Type characteristic to specify an override skill category and level for a specific message category / number then you would reference this characteristic type using this option.

---

**NOTE: Skill Setup.** Refer to the description of the above base package algorithms for further details on how to setup skill level information.

---



---

**NOTE: More Options.** Your implementation may define additional options types. You do this by add new lookup values to the lookup field **F1QM\_OPT\_TYP\_FLG**.

---



---

**NOTE: Only one.** The system expects only one **Activity Queue Management** feature configuration to be defined.

---

## Defining To Do Roles

This section describes the control table used to maintain To Do roles.

### To Do Role - Main

The **Main** page is used to define basic information about a To Do role.

To maintain this information, select **Admin > General > To Do Role**.

#### Description of Page

Enter a unique **To Do Role** and **Description** for the To Do role.

The grid contains the ID of each **User** that may view and work on entries assigned to this role. The First Name and Last Name associated with the user is displayed adjacent.

---

**NOTE: System Default Role.** The system supplies a default role **F1\_DFLT** linked to each system To Do type. This is done so that To Do functionality may be tested prior to the creation of appropriate business oriented To Do roles.

---

#### Where Used

Follow this link to view the tables that reference [CI\\_ROLE](#) in the data dictionary schema viewer.

In addition, various "type" objects or algorithms may reference a To Do role to use when creating a To Do for a given business scenario. This is dependent on your specific product.

## To Do Role - To Do Types

The **To Do Types** page defines the To Do types that may be viewed and worked on by users belonging to a given To Do role.

To maintain this information, select **Admin > To Do Role > Search** and navigate to the **To Do Types** page.

### Description of Page

Enter the ID of each **To Do Type** whose entries may be viewed and worked on by the role.

**Use As Default** is a display-only field that indicates if the role is assigned to newly created entries of this type. You may define the default role for a given To Do type on the To Do Type maintenance page.

---

**CAUTION:** If you remove a To Do type where this role is the default, you must define a new role as the default for the To Do type. You do this on the To Do Type maintenance page.

---

## Defining To Do Types

This section describes the control table used to maintain To Do types.

### To Do Type - Main

The **Main** page is used to define basic information about a To Do type.

---

**FASTPATH:** Refer to [The Big Picture Of To Do Lists](#) for more information about To Do types and To Do lists in general.

---

To maintain this information, select **Admin > General > To Do Type**.

---

**CAUTION:** Important! If you introduce a To Do type, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

### Description of Page

Enter a unique **To Do Type** and **Description** for the To Do type.

**Owner** indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

Use the **Detailed Description** to provide further details related to the To Do Type.

Enter the default **Priority** of To Do entries of this type in respect of other To Do types. Refer to [The Priority Of A To Do Entry](#) for more information.

For **To Do Type Usage**, select **Automatic** if To Dos of this type are created by the system (i.e., a background process or algorithm). Select **Manual** if a user can create a To Do of this type online.

Define the **Navigation Option** for the page into which the user is transferred when drilling down on a To Do entry of this type.

Use **Creation Process** to define the background process, if any, that is used to manage (i.e., create and perhaps complete) entries of this type. A **Creation Process** need only be specified for those To Do types whose entries are created by a background process. Refer to [To Do Entries Created By Background Processes](#) for more information.

Use **Routing Process** to define the background process that is used to download entries of a given type to an external system, if any. A **Routing Process** need only be specified for those To Do types whose entries are routed to an external system (e.g., an Email system or an auto-dialer). Refer to [To Do Entries May Be Routed Out Of The System](#) for more information.

Use **Message Category** and **Message Number** to define the message associated with this To Do type's entries. Note: this message will only be used if the process that creates the To Do entry does not supply a specific message number. For example, the process that creates To Do entries that highlight bill segments that are in error would not use this message; rather, the entries are marked with the message associated with the bill segment's error.

Use the characteristics collection to define a **Characteristic Type** and **Characteristic Value** common to all To Do entries of this type. You may enter more than one characteristic row for the same characteristic type, each associated with a unique **Sequence** number. If not specified, the system defaults it to the next sequence number for the characteristic type.

#### Where Used

Follow this link to view the tables that reference [CI\\_TD\\_TYPE](#) in the data dictionary schema viewer.

## To Do Type - Roles

The **Roles** page defines the roles who may view and work on entries of a given To Do type.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Roles** page.

#### Description of Page

Enter each **To Do Role** that may view and work on entries of a given type. Turn on **Use as Default** if the role should be assigned to newly created entries of this type. Only one role may be defined as the default per To Do type.

---

**FASTPATH:** Refer to [To Do Entries Reference A Role](#) for more information about roles and To Do entries.

---

## To Do Type - Sort Keys

The **Sort Keys** page defines the various ways a To Do list's entries may be sorted. For example, when you look at the bill segment error To Do List, you have the option of sorting the entries in error number order, account name order, or in customer class order.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Sort Keys** page.

---

**CAUTION:** Do not change this information unless you are positive that the process / algorithm that creates entries of a given type stores this information on the entries.

---

#### Description of Page

The following fields display for each sort key.

**Sequence** is the unique ID of the sort key.

**Description** is the description of the sort key that appears on the To Do list.

**Use as Default** indicates the default sort key (the one that is initially used when a user opens a To Do list). Only one sort key may be defined as the default per To Do type.

**Sort Order** indicates whether the To Do entries should be sorted in **Ascending** or **Descending** order when this sort key is used.

**Owner** indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

## To Do Type - Drill Keys

The **Drill Keys** page defines the keys passed to the application service (defined on the Main page) when you drill down on an entry of a given type.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Drill Keys** page.

---

**CAUTION:** Do not change this information unless you are positive that the process / algorithm that creates entries of a given type stores this information on the entries.

---

### Description of Page

**Navigation Option** shows the page into which the user is transferred when drilling down on a To Do entry of this type.

The following fields display for each drill key.

**Sequence** is the unique ID of the drill key.

**Table** and **Field** are passed to the application service when you drill down on an entry of a given type.

**Owner** indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

## To Do Type - Message Overrides

The **Message Overrides** page is used if you want To Do entries that reference a given message category / number to be routed to a specific To Do role (or suppressed altogether) or if you want to associate a script to a given message category / number.

---

**FASTPATH:** Refer to [To Do Entries Reference A Role](#) and [To Do Entries Can Be Rerouted Or Suppressed](#) for more information.

---

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Message Overrides** page.

### Description of Page

The following fields display for each override.

**Message Category** and **Number** allow the message to be overridden.

**Exclude To Do Entry** indicates if a To Do entry of this type that references the adjacent **Message Category** and **Number** should NOT be created.

**Override Role** indicates the to do role to which a To Do entry of this type that references the adjacent **Message Category** and **Number** should be addressed. This field is protected if **Exclude To Do Entry** is on.

**Script** indicates the script that should execute when a user drills down on a To Do entry of this type that references the adjacent **Message Category** and **Number**. This field is protected if **Exclude To Do Entry** is on. Refer to [Working On A To Do Entry](#) for more information.

## To Do Type - To Do Characteristics

The **To Do Characteristics** page defines characteristics that can be defined for To Do entries of this type. The characteristic types for characteristics that are linked to the To Do entry as a result of a pre-creation algorithm do not need to be defined here.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **To Do Characteristics** page.

Turn on the **Required** switch if the **Characteristic Type** must be defined on To Do entries of this type.

Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

## To Do Type - Algorithms

The **To Do Algorithms** page defines the algorithms that should be executed for a given To Do type.

To maintain this information, select **Admin > To Do Type > Search** and navigate to the **Algorithms** page.

## Description of Page

The grid contains **Algorithms** that control important To Do functions. If you haven't already done so, you must [set up the appropriate algorithms](#) in your system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

<b>System Event</b>	<b>Optional / Required</b>	<b>Description</b>
<b>Calculate Priority</b>	Optional	<p>Algorithms of this type may be used to calculate a To Do entry's priority. They are called initially when a To Do entry is created and each time it gets updated so long as the To Do entry's priority has not been manually overridden. Once overridden, these algorithms are not called anymore.</p> <p>Note that it is not the responsibility of the algorithms to actually update the To Do entry with the calculated priority value but rather only return the calculated value. The system carries out the update as necessary.</p> <p>If more than one algorithm is plugged-in the system calls them one by one until the first to return a calculated priority.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>External Routing</b>	Optional	<p>Algorithms of this type may be used to route a To Do entry to an external system.</p> <p>The base package <a href="#">F1-TDEER</a> background process invokes the algorithms for every To Do entry that its type references the process as the <b>Routing Process</b> and that the entry was not already routed. The background process marks an entry as routed by updating it with the batch control's current run number.</p> <p>If more than one algorithm is plugged-in the batch process calls them one by one until the first to indicate the To Do entry was routed.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>To Do Information</b>	Optional	<p>We use the term "To Do information" to describe the basic information that appears throughout the system to describe a To Do entry. The data that appears in "To Do information" is constructed using this algorithm.</p> <p>Plug an algorithm into this spot to override the "To Do information" algorithm on installation options or the system default "To Do information" if no such algorithm is defined on installation options.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>
<b>To Do Post-Processing</b>	Optional	<p>Algorithms of this type may be used to validate and/or further process a To Do entry that has been added or updated.</p> <p>Click <a href="#">here</a> to see the algorithm types available for this system event.</p>

## List of System To Do Types

The To Do types available to use with the product are found in the To Do Type page. In addition, they may be viewed in the [application viewer's To Do type](#) viewer. If your implementation adds To Do types, you may [regenerate](#) the application viewer to see your additions reflected there.

## Implementing The To Do Entries

To enable the To Do entries visible in the To Do Type page and application viewer, you must configure the system as follows:

- Define the To Do roles associated with each To Do type and link the appropriate users to them. Once you have defined the roles appropriate for your organization's To Do types, remove the reference to this system default role **F1\_DFLT**. Refer to [To Do Entries Reference A Role](#) for more information.
- For any To Do Type that is provided for a specific background process, the To Do simply needs to reference the appropriate Creation Background Process. When the background process is scheduled, To Dos are created based on the logic of the related background process. This applies to [To Dos Created for Object-Specific Error Conditions](#) and [Dedicated To Do Background Processes](#).
- For any To Do Type that is provided for creation by an algorithm or other process, there may be configuration required to populate that To Do type as an algorithm parameter or as an attribute on a control table.

---

**NOTE:** Refer to the description of the To Do type for more information.

---

## Background Processes

---

This chapter covers various topics related to background processes. Besides providing an overview of background process functionality, the various tools available within the application to define, submit and monitor background processes are covered.

---

**NOTE:** Your specific source application may have additional background process topics. Please refer to the documentation section that applies to your source application for more information.

---

## The Big Picture of Background Processes

This section describes various topics related to the background processes that perform many important functions throughout your product such as:

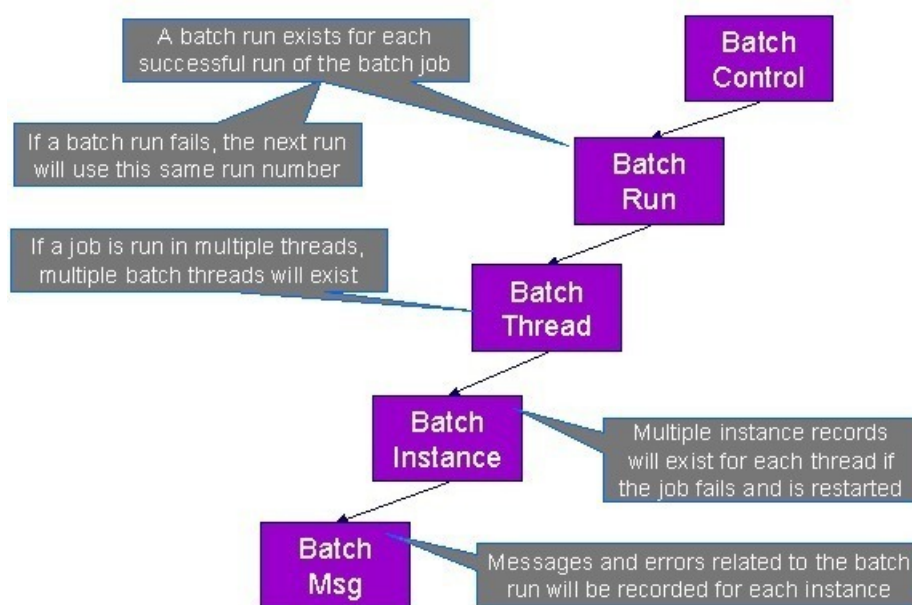
- Processing To Do Entries
- Monitor processes that select records in a given state to progress them to their next state in their lifecycle
- Processes that purge data
- Processes that extract data
- And many more...

## Background Processing Concepts

While the system relies on a scheduler to secure and execute its background processes, there are additional issues that you should be familiar with:

- Batch control records are used for the following purposes:
  - Define the code that executes the logic associated with the background process.
  - For processes that extract information, the batch control record defines the next batch number to be assigned to new records that are eligible for extraction. For example, the batch control record associated with the process that routes To Do entries to an external system defines the next batch number to be assigned to new To Do entries that are configured with this batch control. When this To Do external routing process next runs, it selects all To Do entries marked with the current batch number (and increments the next batch number).
  - The batch control record for each background process organizes audit information about the historical execution of the background process. The system uses this information to control the restart of failed processes. You can use this information to view error messages associated with failed runs.
  - Many processes have been designed to run in parallel in order to speed execution. For example, the process that applies updates for a migration data set import for CMA can be executed so that multiple "threads" are processing a different subset of records (and multiple threads can execute at the same time). Batch control records associated with this type of process organize audit information about each thread in every execution. The system uses this information to control the restart of failed threads. Refer to [Parallel Background Processes](#) for more information.
  - Some processes define extra parameters. These parameters are defined with the batch control. Default values may also be captured for each parameter. They will be used when the [background process is submitted on-line](#).

The following diagram illustrates the relationships that exist for batch control records.



Results of each batch run can be viewed using the [Batch Run Tree](#) page.

## Parameters Supplied To Background Processes

All background processes receive the following parameters.



- **Batch code.** Batch code is the unique identifier of the background process.
- **Batch thread number.** Thread number is only used for background processes that can be run in multiple parallel threads. It contains the relative thread number of the process. For example, if the billing process has been set up to run in 20 parallel threads, each of the 20 instances receives its relative thread number (1 through 20). Refer to [Optimal Thread Count for Parallel Background Processes](#) for more information.
- **Batch thread count.** Thread count is only used for background processes that can be run in multiple parallel threads. It contains the total number of parallel threads that have been scheduled. For example, if the billing process has been set up to run in 20 parallel threads, each of the 20 instances receives a thread count of 20. Refer to [Optimal Thread Count for Parallel Background Processes](#) for more information.
- **Batch rerun number.** Rerun number is only used for background processes that download information that belongs to given run number. It should only be supplied if you need to download an historical run (rather than the latest run).
- **Batch business date.** Business date is only used for background processes that use the current date in their processing. For example, billing using the business date to determine which bill cycles should be downloaded. If this parameter is left blank, the system date is used. If supplied, this date must be in the format YYYY-MM-DD. Note: this parameter is only used during QA to test how processes behave over time.
- **Override maximum records between commits.** This parameter is optional and overrides each background process's Standard Commit. You would reduce this value, for example, if you were submitting a job during the day and you wanted more frequent commits to release held resources. You might want to increase this value when a background process is executed at night (or weekends) and you have a lot of memory on your servers.
- **Override maximum minutes between cursor re-initiation.** This parameter is optional and overrides each background process's Standard Cursor Re-Initiation Minutes. You would reduce these values, for example, if you were submitting a job during the day and you wanted more frequent commits to release held resources (or more frequent cursor initiations). You might want to increase these values when a background process is executed at night (or weekends) and you have a lot of memory on your servers.

---

**NOTE:** The maximum minutes between cursor re-initiation is relevant for Oracle implementations only. Most of the system background processes contain an outermost loop / cursor. The cursor is opened at the beginning of the process and closed at the end. If Oracle feels that the cursor is open for too long, it may incorrectly interpret this as a problem and may issue an error that the snapshot is too old. The commit processing for the background processes is designed to refresh the cursor based on the minutes between cursor re-initiation in order to prevent this error.

---

- **User ID.** Please be aware of the following in respect of user ID:
  - The user ID is a user who should have access to all application services in the system. This is because some batch processes update records via services that may be check security.
  - Any batch process that stamps a user ID on a record it creates or updates uses this user ID.
  - This user ID's [display profile](#) controls how dates and currency values are formatted in messages.
- **Password.** Password is not currently used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace program at start (Y/N), trace program exit (Y/N), trace SQL (Y/N) and output trace (Y/N).** These switches are only used during QA and benchmarking. If trace program start is set to Y, a message is displayed whenever a program is started. If trace program at exist is set to Y, a message is displayed whenever a program is exited. If trace SQL is set to Y, a message is displayed whenever an SQL statement is executed. If output trace is set to Y, special messages formatted by the background process are written.

---

**NOTE:** The information displayed when the output trace switch is turned on depends on each background process. It is possible that a background process displays no special information for this switch.

---



## Override Maximum Errors in Batch Process Parameter

Each of the batch processes has, as part of its run parameters, a preset constant that determines how many errors that batch process may encounter before it is required to abort the run. A user can override this constant with an optional additional parameter (defined as either **MAX-ERRORS** or **maxErrors**). If a user chooses not to enter a value for the parameter, the process uses its own preset constant.

The input value must be an integer that is greater than or equal to zero. The maximum valid value for this parameter is 999,999,999,999,999.

The syntax for entering this additional parameter when submitting the batch process is "MAX-ERRORS|maxErrors=PARAM VALUE", where the PARAM VALUE is the desired value (e.g., **MAX-ERRORS=50** or **maxErrors=50**).

## Extra Parameters

Some background processes receive additional parameters that are specific to their functionality. When a process receives additional parameters, they are defined and documented in the batch control entry in the application. They are also visible in the [batch control](#) viewer (part of the [application viewer](#)).

The syntax for entering these parameters when submitting the batch process is "PARAM-NAME=PARAM VALUE", where PARAM-NAME is the name of the parameter as indicated in the batch control record. For example, if the batch control includes the **ADD-WORK-DAYS**, with possible values of **Y** and **N**, and you want to pass a value of **N**, enter the following when prompted: **ADD-WORK-DAYS=N**.

## Indicating a File Path

Some of the system background processes use extra parameters to indicate a File Path and/or File Name for an input file or an output file. For example, most extract processes use File Path and File Name parameter to indicate where to place the output file.

When supplying a FILE-PATH variable, the directory specified in the FILE-PATH must already exist and must grant write access to the administrator account for the product. You may need to verify a proper location with your systems administrator.

The syntax of the FILE-PATH depends on the platform used for the product application server. Contact your system administrator for verification. For example, if the platform is UNIX, use forward slashes and be sure to put a trailing slash, for example [/spltemp/filepath/](#).

## Processing Errors

When a background process detects an error, the error may or may not be related to a specific object that is being processed. For example, if the program finds an error during batch parameter validation, this error is not object-specific. However, if the program finds an error while processing a specific bill, this error is object-specific. The system reports errors in one of the following ways:

- Errors that are not object-specific are written to the error message log in the [Batch Run Tree](#).
- Some batch processes create entries in an "exception table" for certain object-specific errors. For example, an error detected in the creation of a bill in Oracle Utilities Customer Care and Billing may be written to the bill exception table. If an error is written to an exception table, it does not appear in the batch run tree. For each exception table, there is an associated to do entry process that creates a To Do Entry for each error to allow a user to correct the problem on-line.
- For some background processes, errors that do not result in the creation of an exception record may instead generate a To Do entry directly. For these processes, if you wish the system to directly create a To Do entry, you must configure the To Do type appropriately. Refer to [To Do entry for object-specific errors](#) for information about configuring the To Do

type. If the background process detects an object specific error AND you have configured the system to create a To Do entry, the error is not written to the batch run tree. If you have configured your To Do type to not create To Do entries for certain errors, these errors are written to the [batch run tree](#).

---

**NOTE: Some processes create exceptions and To Do entries.** It is possible for a background process to create entries in an exception table and create To Do entries directly, depending on the error. Consider batch billing in Oracle Utilities Customer Care and Billing; any conditions that cause a bill or bill segment to be created in **error** status result in a record added to the bill exception table or the bill segment exception table. However, any object-specific error that is not related to a specific bill or bill segment or any error that prevents a bill or bill segment from being created may result in a To Do entry for the object-specific error.

---

## Parallel Background Processes

Many processes have been designed to run in parallel in order to speed execution. For example, the billing process in Oracle Utilities Customer Care and Billing can be executed so that bills are produced in multiple "threads" (and multiple threads can execute at the same time).

---

**NOTE:** The detailed description in the metadata for each batch control provided with the system should indicate if it may be run in parallel.

---

By default the system distributes the data for the multiple threads using the primary key of the main table of the process. For example, if you are running BILLING in Oracle Utilities Customer Care and Billing with multiple threads, the batch process distributes the accounts across the multiple threads. The Account ID is 10 digits long so if you run it with 4 threads, the records are processed as follows:

- Thread 1: Account IDs 0000000001 - 2500000000
- Thread 2: Account IDs 2500000001 - 5000000000
- Thread 3: Account IDs 5000000001 - 7500000000
- Thread 4: Account Ids 7500000001 - 9999999999

Note that the multi-threading logic relies on the fact that primary keys for master and transaction data are typically system generated random keys.

---

**NOTE: Overriding the thread ranges.** Your implementation has the ability to override the thread ranges if certain data in your system takes longer to process. For example, imagine you have a single account in Oracle Utilities Customer Care and Billing that has thousands of service agreements (maybe the account for a large corporation or a major city). You may want to set up the thread ranges to put this large account into its own thread and distribute the other accounts to the other threads. To do this, you should create the appropriate batch thread records ahead of time in a status of **Thread Ready ( 50)** with the key ranges pre-populated. Note that the base product does not provide the ability to add batch thread records online. If you are interested in more information about this technique, contact Customer Support.

---

## Optimal Thread Count

Running a background process in multiple threads is almost always faster than running it in a single thread. The trick is determining the number of threads that is optimal for each process.

---

**NOTE:** A good rule of thumb is to have one thread for every 100 MHz of application server CPU available. For example if you have four 450 MHz processors available on your application server, you can start with 18 threads to begin your testing:  $(450 * 4) / 100 = 18$ .

---

This is a rule of thumb because each process is different and is dependent on the data in your database. Also, your hardware configuration (i.e., number of processors, speed of your disk drives, speed of the network between the database server and

the application server) has an impact on the optimal number of threads. Please follow these guidelines to determine the optimal number of threads for each background process:

- Execute the background process using the number of threads dictated by the rule of thumb (described above). During this execution, monitor the utilization percentage of your application server, database server and network traffic.
- If you find that your database server has hit 100% utilization, but your application server hasn't one of the following is probably occurring:
  - There may be a problematic SQL statement executing during the process. You must capture a database trace to identify the problem SQL.
  - It is also possible that your commit frequency may be too large. Commit frequency is a parameter supplied to every background process. If it is too large, the database's hold queues can start swapping. Refer to [Parameters Supplied to Background Processes](#) for more information about this parameter.
- It is normal if you find that your application server has hit 100% utilization but your database server has not. This is normal because, in general, all processes are CPU bound and not IO bound. At this point, you should decrease the number of threads until just under 100% of the application server utilization is achieved. And this will be the optimal number of threads required for this background process.
- If you find that your application server has NOT hit 100% utilization, you should increase the number of threads until you achieve just under 100% utilization on the application server. And remember, the application server should achieve 100% utilization before the database server reaches 100% utilization. If this proves not to be true, something is probably wrong with an SQL statement and you must capture an SQL trace to determine the culprit.

Another way to achieve similar results is to start out with a small number of threads and increase the number of threads until you have maximized throughput. The definition of "throughput" may differ for each process but can be generalized as a simple count of the records processed in the batch run tree. For example, in the Billing background process in Oracle Utilities Customer Care and Billing, throughput is the number of bills processed per minute. If you opt to use this method, we recommend you graph a curve of throughput vs. number of threads. The graph should display a curve that is steep at first but then flattens as more threads are added. Eventually adding more threads will cause the throughput to decline. Through this type of analysis you can determine the optimum number of threads to execute for any given process.

## Timed Batch Processes

Most batch jobs are submitted via a batch scheduler. In the absence of a scheduler, a batch control may be configured as "timed" triggering the framework to monitor and schedule these batch jobs as defined by the timer interval. The timer interval defines the desired interval between starts (in seconds). The system schedules new batch runs at each interval if the last instance of the job has completed.

When configuring a batch control as "timed", other default information must be provided, including the User ID and Language to use for submitting the job and the email address for notification, if desired.

Timed batch controls also include an Active setting, allowing for an implementation to temporarily stop further executions of the batch job (but retain the other timer settings).

Timed jobs are controlled by the default threadpool and not by a scheduler. When the **DEFAULT** threadpoolworker starts it will start executing any job for a Batch Control configured as **Timed** with the Timer Active set to **Yes**. This is whether the batch daemon or batch server is enabled or not.

## How to Re-extract Information

If you need to recreate the records associated with an historical execution of an extract process, you can - simply supply the desired batch number when you request the batch process.

## How to Submit Batch Jobs

Most batch jobs are submitted via a batch scheduler.

In addition, you can manually submit your adhoc background processes or submit a special run for one of your scheduled background processes using the online [batch job submission](#) page.

## How to Track Batch Jobs

You can track batch jobs using the batch process pages, which show the execution status of batch processes. For a specified batch control id and run id, the tree shows each thread, the run-instances of each thread, and any messages (informational, warnings, and errors) that might have occurred during the run.

---

**FASTPATH:** For more information, refer to [Tracking Batch Processes](#).

---

## How to Restart Failed Jobs and Processes

Every process in the system can be easily restarted if it fails (after fixing the cause of the failure). All you have to do is resubmit the failed job; the system handles the restart.

## Assessing Level of Service

For some background processes, an implementation may wish to supply an algorithm that checks some conditions to assess whether or not the process is performing as expected. This algorithm could be simply verifying that the batch process is running as frequently as expected. Or it could be used to check the performance of the job to see if it is running as efficiently as expected. Or it could analyze the data processed by the background process to assess whether there may be some problem with the quality of the data.

The system provides a Level of Service plug-in spot on [batch control](#) to configure the appropriate algorithm for a given background process, if desired. The algorithm is expected to return a value to indicate the 'level of service' determined along with a message indicating the reason for the value. The following Level of Service values are supported:

- **Normal.** Indicates that the algorithm did not detect any issues.
- **Warning.** Indicates that the algorithm found some issues that may or may not indicate a problem.
- **Error.** Indicates that the algorithm found some issues that should be investigated.
- **Disabled.** Indicates that the algorithm could not properly execute the level of service logic.

When viewing a [batch control](#) record, if there is a level of service algorithm configured, its logic is executed and the results are displayed. The level of service is also part of the [Health Check](#) service.

## System Background Processes

---

**NOTE: List of system background processes.** The list of background processes provided in the base product may be viewed in the [application viewer's batch control](#) viewer. In addition if your implementation adds batch control records, you may [regenerate](#) the application viewer to see your additions reflected there.

---

# Defining Batch Controls

The system is delivered with all necessary batch controls. Implementations may define default values for parameters. In addition, implementations may define their own background processes.

To view background processes, open **Admin > System > Batch Control**. Refer to [Background Processing Concepts](#) for more information.

---

**CAUTION:** Important! If you introduce a new batch process, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

## Description of Page

Enter an easily recognizable **Batch Process** and **Description** for each batch process.

**Owner** indicates if this batch control is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a batch control. This information is display-only.

Use the **Detailed Description** to describe the functionality of the batch process in detail.

Use **Batch Control Type** to define the batch process as either **Timed** or **Not Timed**. A [Timed batch process](#) will be automatically initialized on a regular basis. A Not Timed process needs to be run manually or through a scheduler.

Use **Batch Control Category** to categorize the process for documentation purposes. The base values provided are as follows:

- **Ad Hoc.** Processes of this type are run on an ad hoc basis, only when needed. For example, if there is a process to do a mass cancel / correction of data, it would only be run when a situation occurs requiring this.
- **Extract.** Extract processes extract information that is interfaced out of the system. Processes of this type typically extract records marked with a given run number. If the requester of the process does not supply a specific run number, the system assumes that the latest run number should be extracted. If you need to re-extract an historical batch, you simply supply the respective run number when you request the batch process.
- **ILM. Information Lifecycle Management** jobs are crawler background processes that are associated with the ILM based storage solution.
- **Monitor.** Processes of this type are processes related to business objects with a lifecycle state that defines [monitor algorithms](#). The monitor process selects records in a given state and executes its algorithms, which may cause the record to transition to another state or may trigger some other logic to occur. Using configuration, the monitor process may target only specific records. Refer to [Monitoring Batch Processes](#) for more information. Note that these types of background processes can be considered a subset of **Process What's Ready**
- **Process What's Ready.** Processes of this type create and update records that are “ready” for processing. The definition of “ready” differs for every process. For example, a payment upload process creates payments for every record that is **pending**. An overdue event monitor activates pending overdue events that have reached their trigger date.
- **To Do Entry.** Processes of this type are used to detect a given situation and create a To Do Entry. Refer to [To Do Entries Created by Background Processes](#) for more information.
- The following categories are related to the data conversion / migration processes, which are not applicable to all products:
  - **Conversion.** Processes of this type are dedicated to converting or migrating data from external applications into the product.
  - **Object Validation.** Processes of this type are dedicated to validate data within objects for conversion or migration purposes.
  - **Referential Integrity.** Processes of this type are dedicated to validate referential integrity within objects for conversion or migration purposes.

---

**NOTE:** Additional categories may be introduced by your specific product.

---

If the batch process is Timed, then the following fields are available:

- **Timer Interval** is the number of seconds between batch process submissions. The system will start the next run this many seconds from the start time of the previous run.
- **User ID** is the ID under which the batch process will run.
- **Email Address** is the Email address to be used for notification if the batch process fails.
- **Timer Active** allows you to temporarily switch off the timer while retaining the other settings for the timed job.
- **Batch Language** is the language associated with the batch process.

Use **Program Type** to define if the batch process program is written in **Java** or **Java (Converted)**, meaning that the program has been converted into Java.

---

**NOTE:** **Java (Converted)** program types are not applicable to all products.

---

Use **Program Name** to define the Java class / program associated with your batch process:

---

**NOTE: View the source.** If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the [Java docs viewer](#).

---

**Level of Service** shows the output of the [level of service](#) algorithm for the batch control. It shows the level of service lookup value along with a message indicating the reason for the output value. Note that if no level of service algorithm is found, then the value **Disabled** is shown with a message indicating that no algorithm is provided for this batch control.

The **Last Update Timestamp**, **Last Update Instance** and **Next Batch Nbr** are used for audit purposes.

Turn on **Accumulate All Instances** to control how this batch control is displayed in the [Batch Run Tree](#). If checked, the run statistics (i.e., "Records Processed" and "Records in Error") for a thread are to be accumulated from all the instances of the thread. This would include the original thread instance, as well as any restarted instances. If this is not turned on, only the ending (last) thread instance's statistics are used as the thread's statistics. This may be preferred for certain types of batch processes where the accumulation would create inaccurate thread statistics, such as those that process flat files and, therefore, always start at the beginning, even in the case of a restart.

The following fields are default values that are used when a batch job is submitted for the batch control:

- Use **Thread Count** to control whether a background process is run single threaded or in multiple parallel threads. This value defines the total number of threads that have been scheduled.
- Select **Trace Program Start** if you want a message to be written whenever a program is started.
- Select **Trace SQL** if you want a message to be written whenever an SQL statement is executed.
- Use **Override Nbr Records to Commit** to define the default number of records to commit. This is used as the default value for timed jobs as well as online submission of jobs that are not timed.
- Select **Trace Program Exit** if you want a message to be written whenever a program is exited.
- Select **Trace Output** if you want a message to be displayed for special information logged by the background process.

For more information about these fields, see [Batch Job Submission - Main](#)

The parameter collection is used to define additional parameters required for a particular background process. The following fields should be defined for each parameter:

**Sequence.** Defines the relative position of the parameter.

**Parameter Name.** The name of the parameter as defined by the background process program.

**Description.** A description of the parameter.

**Detailed Description.** A more detailed description of the parameter.

**Required.** Indicates whether or not this is a required parameter.

**Parameter Value.** The default value, if applicable. Note that an implementation may define a default value for base provided batch controls.

**Security.** Indicates whether the system should **Encrypt** the parameter value or not. A value of **Encrypt** means that the parameter value is stored in the database and written to the log files using encryption. In addition, the parameter is written to the log files with asterisks. The setting applies to values entered here as well as in the online [Batch Submission](#). If there is no need to secure the parameter value, use the default setting of **None**.

**Owner** Indicates if this batch process is owned by the base package or by your implementation (Customer Modification). The system sets the owner to **Customer Modification** when you add a batch process. This information is display-only.

## Batch Control - Algorithms

Use this page to maintain a batch control's algorithms. Open this page using **Admin > System > Batch Control** and then navigate to the **Algorithms** tab.

### Description of Page

The **Algorithms** grid contains algorithms that control important functions for instances of this batch control. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes the **System Events**.

System Event	Optional / Required	Description
Level of Service	Optional	Algorithms of this type are called to determine the Level of Service provided by a batch control. Refer to <a href="#">Assessing Level of Service</a> for more information.  Click <a href="#">here</a> to see the algorithm types available for this system event.

## On-Line Batch Submission

The on-line [batch submission](#) page enables you to request a specific background process to be run. When submitting a background process on-line, you may override standard system parameters and you may be required to supply additional parameters for your specific background process. After submitting your background process, you may use this page to review the status of the submission.

The following topics further describe logic available for on-line submission of background processes.

## Batch Submission Creates a Batch Run

When you request a batch job to be submitted from on-line, the execution of the desired background process will result in the creation of a batch run. Just as with background processes executed through your scheduler, you may use the [Batch Run Tree](#) page to view the status of the run, the status of each thread, the run-instances of each thread, and any messages that might have occurred during the run.



---

**NOTE:** Your on-line submission record is assigned a status value so that you may know whether your job has been submitted and whether or not it has ended, however, it will not contain any information about the results of the background process itself. You must navigate to the [Batch Run Tree](#) page to view this detail.

---

## Jobs Submitted in the Background

When you save a record on the batch job submission page, the batch job does not get submitted automatically. Rather, it saves a record in the batch job table. A special background process will periodically check this table for pending records and will execute the batch job. This background process will update the status of the batch job submission record so that a user can determine when their job is complete.

---

**NOTE:** At installation time, your system administrator will set up this special background process to periodically check for pending records in the batch job submission table. Your administrator will define how often the system will look for pending records in this table.

---

It should be noted that this special background process only submits one pending batch job submission record at a time. It submits a job and waits for it to end before submitting the next pending job.

---

**NOTE:** If you request a batch job to be run multi-threaded, the special background process will submit the job as requested. It will wait for all threads to complete before marking the batch job submission record as **ended**. Refer to [Running Multi-threaded Processes](#) for more information.

---

## Email Notification

If you wish the system to inform you when the background process completes, you may supply your email address. The email you receive will contain details related to the batch job's output; similar to the job results you would see from your batch scheduler.

---

**NOTE:** This assumes that during the installation process, your system administrator configured the system to enable email notification. Your administrator may also override the amount of detail included in the email notification.

---

## Running Multi-Threaded Processes

Many of the system background processes may be run multi-threaded. When submitting a background process on-line, you may also run a multi-threaded process or run a single thread of a multi-threaded process. The fields Thread Count and Thread Number on the [batch submission](#) page control the multi-threaded process requests:

- To run a multi-threaded process, indicate the number of threads in **Thread Count** and enter **0** in the **Thread Number**.
- To run a single thread in a multi-threaded process, indicate the number of threads in Thread Count and indicate the Thread Number you would like to run.
- To run a process as a single thread, enter Thread Count = **1** and Thread Number = **1**. This will execute the background process single-threaded.

---

**NOTE:** When running a multi-threaded process, the special background process will wait until all threads are complete before marking the batch job submission record as **Ended**.

---



## Batch Jobs May End in Error

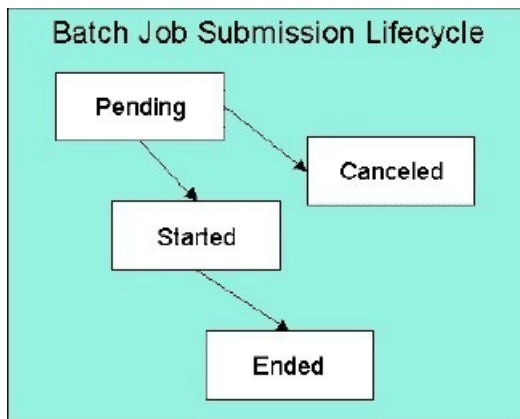
It is possible for your background process to end with an error. When this occurs, your batch job submission record will still be marked as **Ended**. You will need to navigate to the [Batch Run Tree](#) page to determine the status of the batch run.

## Submitting Jobs in the Future

If you wish to request a batch job to be submitted in the future, you may do so when creating your batch job submission record by entering a future submission date. The special background process, which looks for pending records in the batch job submission table, will only submit batch jobs that do not have a future submission date.

## Lifecycle of a Batch Job Submission

The following diagram illustrates the lifecycle of a batch job submission record.



**Pending** — Records are created in **Pending** status. Records in this state are put in a queue to be submitted.

**Canceled**— Users may cancel a pending record to prevent the batch job from being submitted.

**Started**— Once a pending record has been submitted for processing, its status will be changed to **Started**. Records in this status may not be canceled.

**Ended**— When the batch job has finished processing, its status will be changed to **Ended**. Note that records in **Ended** status may have ended in error. Refer to [Batch Jobs May End in Error](#) for more information.

## Batch Job Submission - Main

This page allows you to submit a batch job on-line. Navigate to this page using **Menu > Batch > Batch Job Submission**.

### Description of Page

The **Batch Job ID** is a system generated random number that identifies a particular submission.

To submit a batch job, choose the **Batch Code** for the process you wish to submit.

---

**NOTE: List of system background processes.** The list of background processes provided in the base product may be viewed in the [application viewer's batch control](#) viewer.

---

The following parameters are provided with each background process:

**Thread Number** is used to control whether a background processes is run single threaded or in multiple parallel threads. It contains the relative thread number of the process. For example, if the process has been set up to run in 20 parallel threads, each of the 20 instances receives its relative thread number (1 through 20). Refer to [Running Multi-threaded Processes](#) for more information about populating this field.

---

**NOTE:** Not all processes may be run multi-threaded. Refer to the description of a batch control to find out if it runs multi-threaded.

---

**Thread Count** is used to control whether a background processes is run single threaded or in multiple parallel threads. It contains the total number of threads that have been scheduled. For example, if the process has been set up to run in 20 parallel threads, each of the 20 instances receives a thread count of 20. Refer to [Running Multi-threaded Processes](#) for more information about populating this field.

**Batch Rerun Number** is only used for background processes that download information that belongs to given run number. It should only be supplied if you need to download an historical run (rather than the latest run).

**Batch Business Date** is only used for background processes that use a date in their processing. In Oracle Utilities Customer Care and Billing, for example, a billing process can use the business date to determine which bill cycles should be downloaded. If this parameter is left blank, the system date is used at the time the background process is executed.

---

**NOTE:** Saving a record on this page does not submit the batch job immediately. A special background process will run periodically to find pending records and submit them. Depending on how often the special process checks for pending records and depending on how many other pending records are in the 'queue', there may be a slight lag in submission time. If the desired execution date/time is close to midnight, it is possible that your batch job will run on the day after you submit it. If you have left the business date blank in this case, keep in mind that your business date would be set to the day after you submit the job.

---

**Override Nbr Records To Commit** and **Override Max Timeout Minutes**. These parameters are optional and override each background process's Standard Commit Records and Standard Cursor Re-Initiation Minutes. (Each background process's Standard Commit Records / Standard Cursor Re-Initiation Minutes should be documented in the detailed description of the batch control record). Note that Max Timeout Minutes corresponds to the Cursor Re-initiation Minutes.

---

**FASTPATH:** Refer to [Parameters Supplied to Background Processes](#) for more information.

---

**User ID** is the user ID associated with the run of the background process. Refer to [Parameters Supplied to Background Processes](#) for more information about the significance of the user id.

---

**NOTE:** This field defaults to the id of the current user.

---

**Password** is not currently used.

**Language Code** is used to access language-specific control table values. For example, error messages are presented in this language code.

If you wish the system to notify you when the batch job is complete, enter your **Email ID**. Refer to [Email Notification](#) for more information.

---

**NOTE:** This field defaults to the email address for the current user, if populated on the [user](#) record.

---

The **Desired Execution Date/Time** defaults to the current date and time. Override this information if you wish the background process to be executed at some future date and time. Refer to [Submitting Jobs in the Future](#) for more information.

The **Batch Job Status** indicates the current status of the batch job. Refer to [Lifecycle of a Batch Job Submission](#) for more information.

The **Program Name** associated with the batch control code is displayed.

The following trace parameters may also be supplied to a background process and are only used during QA and benchmarking.

- **Trace Program Start** Turn on this switch if you wish a message to be written whenever a program is started.
- **Trace Program Exit** Turn on this switch if you wish a message to be written whenever a program is exited.
- **Trace SQL** Turn on this switch if you wish a message to be written whenever an SQL statement is executed.
- **Trace Output** Turn on this switch if you wish a message to be displayed for special information logged by the background process.

---

**NOTE:** The information displayed when the trace output switch is turned on depends on each background process. It is possible that a background process displays no special information for this switch.

---

**NOTE:** The location of the output of this trace information is defined by your system administrator at installation time.

---

If additional parameters have been defined for this background process on the Batch Control page, the **Parameter Name**, **Description**, **Detailed Description** and an indicator of whether or not the parameter is **Required** are displayed.

If a default parameter value is configured on the batch control configuration, that value is shown and may be overridden. Confirm or enter the appropriate **Parameter Value** for each parameter. Note that if the parameter value is configured to be Encrypted on the batch control configuration, the value here will be shown encrypted.

Once you have entered all the desired values, **Save** the record in order to include it in the queue for background processes.

If you wish to duplicate an existing batch job submission record, including all its parameter settings, display the submission record you wish to duplicate and use the **Duplicate and Queue** button. This will create a new Batch Job Submission entry in pending status. The new submission entry will be displayed.

If you wish to cancel a **Pending** batch job submission record use the **Cancel** button. The button is disabled for all other status values.

## Tracking Batch Processes

The batch process pages show the execution status of batch processes. For a specified batch control id and run id, the tree shows each thread, the run-instances of each thread, and any messages (informational, warnings, and errors) that might have occurred during the run. Refer to [Defining Batch Controls](#) for more information on how batch control codes are defined.

## Batch Run Tree - Main

This page allows you to view the status of a specific execution of a batch job. Navigate to this page using **Menu > Batch > Batch Run Tree**.

### Description of Page

Select a **Batch Control** process and **Batch Number** to view information and statistics on the batch run's "threads". The following points should help understand this concept:

- Many batch jobs cannot take advantage of your hardware's processing power when they are run singularly. Rather, you'll find that a large percentage of the CPU and/or disk drives are idle.
- In order to minimize the amount of idle time (and increase the throughput of your batch processes), we allow you to set up your batch processes so that multiple instances of a given batch job are executed at the same time. For example, in Oracle Utilities Customer Care and Billing when you schedule the **billing** process, you can indicate that multiple parallel instances should be executed (rather than just one instance). You'd do this so that the processing burden of creating bills for your customers can be spread over multiple processes.
- We refer to each parallel execution of a batch process as a "thread".

- Statistics and information messages are displayed in respect of each thread. Why? Because each thread is a separate execution and therefore can start and end at different times.

The tree includes a node that displays the total number of records processed for the batch run, the total number of records in error for the batch run and the batch run elapsed time. The elapsed time is the longest elapsed time among the batch thread(s). The message is red if there are any records in error.

If the background process has been enabled to create [To Do entries for object specific errors](#), information about the To Do entries are displayed in the tree. This information is not displayed for each thread, but rather all the To Do entries created for the batch run are grouped together. The To Do entries are grouped by their status.

The messages that appear under a thread always show the start and end times of the execution instance. If errors are detected during the execution of the thread, these error messages may also appear in the tree. Refer to [Processing Errors](#) for information about the types of errors that appear in the batch run tree.

## Batch Run Tree - Run Control

By default, if a batch process fails, it will restart. This notebook tab allows you to modify the restart status of a failed run.

Navigate to this page using **Menu > Batch > Batch Run Tree** search for the desired batch control and then navigate to the **Run Control** page.

### Description of Page

On the main page, you must select a **Batch Control**, **Batch Number**, and **Batch Rerun Number** to view a tree of the batch run. On this page, the following information is displayed:

- **Date Time** contains the date and time this batch run last start or last completed.
- **Batch Business Date** is the business date that was supplied to the background process (this date is used as the "system date" by the process).

**Run Status** indicates the status of the batch run. Valid values are: **In Progress**, **Error**, and **Complete**.

If the **Run Status** is **Error**, the system will attempt to restart this run when you attempt to execute the **Batch Control**.

In most situations, this is exactly what you want to happen. However, there are rare situations where you do not want the system to execute a given batch run (e.g., if this run is somehow corrupt and you cannot correct the data for whatever reasons). If you want the system to skip the execution of a batch run (and proceed to the next run), turn on **Do Not Attempt Restart**.

## Service Health Check

The system provides a service that returns an overall assessment of the system's health. The overall response is expressed using an HTTP code. The codes supported by the service are defined in the lookup **HEALTH\_RESPONSE\_FLG** and are as follows:

- A value of 500 (One or More Critical Functions Degraded) is returned if there is any critical issue detected by the service.
- A value of 203 (Non-Critical Function Degraded) is returned if no critical issues are found and there is at least one non-critical issue detected by the service.
- Otherwise a value of 200 (All Checks Successful) is returned.

This health check service is accessible through the business service **F1-HealthCheck**. Also note that the system provides an [Inbound Web Service](#) for this business service (also called **F1-HealthCheck**) allowing external systems to use a web service to retrieve this information.

In addition, the product provides a portal that allows a user to view the detailed results of the health check service.

Navigate using **Admin > System > Health Check** to view this portal.

The zone on the portal displays the following:

**Overall Response** is the HTTP response code returned by the business service as described above.

The grid displays the detail of each individual component checked as part of the system health check. See below for more detail about what components are checked.

- **Health Component** indicate the type of health check performed. Currently the system supports **Batch Level of Service** component type.
- **Health Component Details** provides information about the individual entity checked. The information displayed here depends on the type of health component. If supported by the type of component, the column may be hypertext, allowing the user to drill into the entity itself.
- **Status** displays the status returned by the health component check for this individual entity. The information displayed here depends on the type of health component.
- **Status Reason** provides more detail about why the individual entity is in the indicated status. The information displayed here depends on the type of health component.
- **Response** shows the health check response code assigned to this individual entity's health check response. It is mapped based on the status returned by the health component check.

## Health Component Type Details

The details returned by the business service for this portal depends on the health component type. The system supports the **Batch Level of Service** health component type.

This health component type finds all the batch controls that are configured with a [level of service](#) algorithm and invokes the algorithm for each batch control. The business service populates the output for this health service for each batch control as follows:

- The **Health Component Detail** is populated with the Batch Control code and description. In addition, the navigation information for being able to drill into the batch control are provided and used to build the column as hypertext.
- The **Status** is populated with the description of the Level of Service lookup value returned by the level of service algorithm for this batch control.
- The **Status Reason** is populated with the expanded text of the status reason returned by the level of service algorithm for this batch control.
- The **Response** is populated based on the value of the Level of Service status. It is set to **All Checks Successful** (200) when the Level of Service is **Normal** or **Disabled; Non-Critical Function Degraded** (203) when the Level of Service is **Warning** and **One or More Critical Functions Degraded** (500) when the Level of Service is **Error**.

## The Big Picture of Requests

Requests enable an implementer to design an ad-hoc batch process using the configuration tools.

An example of such a process might be to send an email to a group of users that summarizes the To Do entries that are assigned to them. This is just one example. The request enables many types of diverse processing.

## Request Type Defines Parameters

For each type of process that your implementation wants, you must configure a request type to capture the appropriate parameters needed by the process.

## Previewing and Submitting a Request

To submit a new request, go to **Menu > Batch > Request > Add**. You must select the appropriate request type and then enter the desired parameter values, if applicable.

After entering the parameters, the following actions are possible:

- Click **Save** to submit the request.
- Click **Cancel** to cancel the request.
- Click **Preview** to see a sample of records that satisfy the selection criteria for this request. This information is displayed in a separate map. In addition, the map displays the total number of records that will be processed when the request is submitted. From this map you can click **Save** to submit the request, **Back** to adjust the parameters, or **Cancel** to cancel the request.

When a request is saved, the job is not immediately submitted for real time processing. The record is saved with the status **Pending** and a monitor process for this record's business object is responsible for transitioning the record to **Complete**.

As long as the record is still **Pending**, it may be edited to adjust the parameters. The preview logic described above may be repeated when editing a record.

The actual work of the request, such as generating an email, is performed when transitioning to **Complete** (using an enter processing algorithm for the business object).

## To Do Summary Email

The base product includes a sample request process that sends an email to users that have incomplete To Dos older than a specified number of days.

The following configuration tasks are required to use this process:

- Define an Outbound Message Type. The business object usually defined for the Outbound Message Type is **F1-EmailMessage**.
- Define an External System that contains the Outbound Message Type in one of its steps. In the configuration determine if the communication is through SOA, batch, or real-time processing method when sending the email notification. Refer to [Outbound Messages](#) for more information about configuration needed for the different processing methods.
- Create a Request Type that includes the Outbound Message Type and the corresponding External System.
- Create a Request for the created Request Type.

## Exploring Request Data Relationships

Use the following links to open the application viewer where you can explore the physical tables and data relationships behind the request functionality:

- Click [F1-REQ-TYPE](#) to view the request type maintenance object's tables.
- Click [F1-REQ](#) to view the request maintenance object's tables.

## Defining a New Request

To design a new ad-hoc batch job that users can submit via Request, first create a new Request Type business object. The base product BO for request type, **F1-TodoSumEmailTyp**, may be used as a sample.

The business object for the request includes the functionality for selecting the records to process, displaying a preview map for the user to review, and for performing the actual processing. The base product BO for request, **F1-TodoSumEmailReq**, may be used as a sample. The following points highlight the important configuration details for this business object:

- Special BO options are available for request BOs to support the Preview Request functionality.
  - **Request Preview Service Script.** This script retrieves the information that is displayed when a user asks for a preview of a request.
  - **Request Preview Map.** This map displays the preview of a request.
- The enter algorithm plugged into the Complete state is responsible for selecting the records that satisfy the criteria and processing the records accordingly.

## Setting Up Request Types

Use the Request Type portal to define the parameters to capture when submitting a request. Open this page using **Admin > Request Type > Add**.

This topic describes the base-package zones that appear on the Request Type portal.

**Request Type List.** The Request Type List zone lists every request type. The following functions are available:

- Click a **broadcast** icon to open other zones that contain more information about the request type.
- Click **Add** in the zone's title bar to add a new request type.

**Request Type.** The Request Type zone contains display-only information about a request type. This zone appears when a request type has been broadcast from the Request Type List zone or if this portal is opened via a drill down from another page. The following functions are available:

- Click **Edit** to start a business process that updates the request type.
- Click **Delete** to start a business process that deletes the request type.
- Click **Deactivate** start a business process that deactivates the request type.
- Click **Duplicate** to start a business process that duplicates the request type.
- State transition buttons are available to transition the request type to an appropriate next state.

Please see the zone's help text for information about this zone's fields.

## Maintaining Requests

Use the Request transaction to view and maintain pending or historic requests.

Open this page using **Menu > Batch > Request > Search**. This topic describes the base-package portals and zones on this page.

**Request Query.** Use the query portal to search for an existing request. Once a request is selected, you are brought to the maintenance portal to view and maintain the selected record.

**Request Portal.** This portal appears when a request has been selected from the Request Query portal. The following base-package zones appear on this portal:

- **Actions.** This is a standard actions zone.
- **Request.** The Request zone contains display-only information about a request. Please see the zone's help text for information about this zone's fields.
- **Request Log.** This is a standard log zone.

# Defining Algorithms

---

In this section, we describe how to set up the user-defined algorithms that perform many important functions including:

- Validating the format of a phone number entered by a user.
- Validating the format of a latitude/longitude geographic code entered by a user.
- In products that support payment and billing:
  - Calculating late payment charges.
  - Calculating the recommended deposit amount.
  - Constructing your GL account during the interface of financial transactions to your GL
- And many other functions...

## The Big Picture Of Algorithms

Many functions in the system are performed using a user-defined algorithm. For example, when a CSR requests a customer's recommended deposit amount, the system calls the deposit recommendation algorithm. This algorithm calculates the recommended deposit amount and returns it to the caller.

---

**NOTE: Algorithm = Plug-in.** We use the terms plug-in and algorithm interchangeably throughout this documentation.

---

So how does the system know which algorithm to call? When you set up the system's control tables, you define which algorithm to use for each component-driven function. You do this on the control table that governs each respective function. For example:

- You define the algorithm used to validate a phone number on your phone types.
- You define the algorithm in Oracle Utilities Customer Care and Billing used to calculate late payment charges on each Service Agreement Type that has late payment charges.
- The list goes on...

The topics in this section provide background information about a variety of algorithm issues.

## Algorithm Type Versus Algorithm

You have to differentiate between the type of algorithm and the algorithm itself.

- An **Algorithm Type** defines the program that is called when an algorithm of this type is executed. It also defines the types of parameters that must be supplied to algorithms of this type.
- An **Algorithm** references an **Algorithm Type**. It also defines the value of each parameter. It is the algorithm that is referenced on the various control tables.

---

**FASTPATH:** Refer to [How to Add A New Algorithm](#) for an example that will further clarify the difference between an algorithm and an algorithm type.

---



## How To Add A New Algorithm

Before you can add a new algorithm, you must determine if you can use one of the sample algorithm types supplied with the system. Refer to [List of Algorithm Types](#) for a complete list of algorithm types.

If you can use one of the sample algorithm types, simply add the algorithm and then reference it on the respective control table. Refer to [Setting Up Algorithms](#) for how to do this.

If you have to add a new algorithm type, you may have to involve a programmer. Let's use an example to help clarify what you can do versus what a programmer must do. Assume that you require an additional geographic type validation algorithm. To create your own algorithm type you must:

- Write a new program to validate geographic type in the appropriate manner. Alternatively, you may configure a plug-in script to implement the validation rules. The advantage of the latter is that it does not require programming. Refer to [plug-in script](#) for more information.
- Create an Algorithm Type called **Our Geographic Format** (or something appropriate). On this algorithm type, you'd define the name of the program (or the plug-in script) that performs the function. You'd also define the various parameters required by this type of algorithm.
- After creating the new Algorithm Type, you can reference it on an Algorithm.
- And finally, you'd reference the new Algorithm on the Geographic Type that requires this validation.

## Minimizing The Impact Of Future Upgrades

The system has been designed to use algorithms so an implementation can introduce their own logic in a way that's 100% upgradeable (without the need to retrofit logic). The following points describe strong recommendations about how to construct new algorithm type programs so that you won't have to make program changes during future upgrades:

- Do not alter an algorithm type's hard parameters. For example, you might be tempted to redefine or initialize parameters defined in an algorithm type's linkage section. Do not do this.
- Follow the naming conventions for the new algorithm type code and your source code, i.e., both the source code and the algorithm type should be prefixed with "CM". The reason for this naming convention is to make it impossible for a new, base-package algorithm type from overwriting your source code or algorithm type meta-data (we will never develop a program or introduce meta-data beginning with CM).
- Avoid using inline SQL to perform insert/update/delete. Rather, invoke the base-package's object routines or common routines.
- Avoid using base messages (outside of common messages, i.e., those with a message number < 1000) as we may deprecate or change these messages in future releases. The most common problem is caused when an implementation clones a base package algorithm type program because they need to change a few lines of logic. Technically, to be 100% upgradeable, you should add new messages in the "90000" or greater category (i.e., the category reserved for implementation-specific messages) for every message in your new program even though these messages may be duplicates of those in the base package.

## Setting Up Algorithm Types

The system provides many algorithm types to support base product functionality. If you need to introduce a new type of algorithm, open **Admin > System > Algorithm Type**.

---

**FASTPATH:** Refer to [The Big Picture Of Algorithms](#) for more information.

---

---

**CAUTION:** Important! If you introduce a new algorithm type, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

### Description of Page

Enter an easily recognizable **Algorithm Type** and **Description**.

**Owner** indicates if this algorithm type is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an algorithm type. This information is display-only.

Enter a **Detailed Description** that describes, in detail, what algorithms of this type do.

Use **Algorithm Entity** to define where algorithms of this type can be "plugged in". If a detailed description about an algorithm entity is available, a small help icon is visible adjacent to the dropdown. Click the icon to view the information.

---

**NOTE:** The values for this field are customizable using the [lookup](#) table. This field name is **ALG\_ENTITY\_FLG**.

---

Use **Program Type** to define if the algorithm's program is written using **Java**, a **Plug-In Script**, or **Java (Converted)**, meaning the program has been converted to Java.

---

**NOTE:** **Java (Converted)** program types are not applicable to all products.

---

Use **Program Name** to define the program to be invoked when algorithms of this type are executed:

- If the Program Type is **Java (Converted)**, enter the name of the converted program.
- If the Program Type is **Java**, enter the Java class name.
- If the Program Type is **Plug-In Script**, enter the plug-in [script](#) name. Only plug-in scripts defined for the algorithm entity may be used.

---

**NOTE: View the source.** If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the [Java docs viewer](#). For plug-in scripts, drill into the plug-in script to view the details.

---

Use the **Parameter Types** grid to define the types of parameters that algorithms of this type use. The following fields should be defined for each parameter:

- Use **Sequence** to define the relative position of the **Parameter**.
- Use **Parameter** to describe the verbiage that appears adjacent to the parameter on the Algorithm page.
- Indicate whether the parameter is **Required**. This indicator is used when parameters are defined on algorithms that reference this algorithm type.
- **Owner** indicates if the parameter for this algorithm type is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an algorithm type with parameters. This information is display-only.

---

**NOTE:** When adding a new algorithm type that is for a Java program, the parameters are automatically generated based on the Java code. Once an algorithm type exists, any additional parameters defined in the Java code should be manually added to the algorithm type. For other program types, algorithm type parameters must be manually defined.

---

**NOTE:** When a new algorithm type parameter is added for any program type, existing algorithms for the algorithm type do not automatically get updated with the new parameter. The algorithms must be manually updated.

---

### Where Used

An Algorithm references an Algorithm Type. Refer to [Setting Up Algorithms](#) for more information.

## List of Algorithm Types

The algorithm types available to use with the product may be viewed in the algorithm type page and in the [application viewer's algorithm viewer](#). If your implementation adds algorithm types or adds algorithms to reference existing algorithm types, you may [regenerate](#) the application viewer to see your additions reflected there.

## Setting Up Algorithms

If you need to introduce a new algorithm, open **Admin > System > Algorithm**. Refer to [The Big Picture Of Algorithms](#) for more information.

### Description of Page

Enter an easily recognizable **Algorithm Code** and **Description** of the algorithm. **Owner** indicates if this algorithm is owned by the base package or by your implementation (**Customer Modification**).

---

**CAUTION:** Important! If you introduce a new algorithm, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

Reference the **Algorithm Type** associated with this algorithm.

---

**FASTPATH:** Refer to [Algorithm Type Versus Algorithm](#) for more information about how an algorithm type controls the type of parameters associated with an algorithm.

---

The parameters available for an algorithm are defined on the algorithm type. The system allows a set of parameter values to change over time. Use the parameter scroll to view parameter values for a given **Effective Date**. The **Owner** of the collection of parameters is displayed. The collection shows the **Parameter** description, the **Sequence** and the **Value** for each parameter.

---

**NOTE:** If the product delivers an algorithm with parameter values defined, an implementation may override the base provided parameter values by adding an additional effective dated collection of parameters.

---

**NOTE:** If an algorithm is defined and subsequently a new parameter is added to the algorithm type, existing algorithms for the algorithm type should be updated as follows: Click the “+” to add a new effective dated entry to the parameter collection. At this point the latest list of parameters for the algorithm type are visible. Configure the parameters accordingly.

---

### Where Used

Algorithms are plugged in on control tables throughout the system. Each algorithm type's Algorithm Entity indicates the name of the control table where it is plugged in. The [algorithm viewer](#) in the application viewer may also be used to see a list of plug-in spots along with their algorithm types and algorithms.

## Defining Script Options

---

We use the term "script" to define processing rules that your implementation sets up to control both front-end and back-end processing:

- Rules that control front-end processing are defined using [Business Process Assistant \(BPA\)](#) scripts. For example, your implementation could set up a BPA script to guide a user through your organization's payment cancellation process.

- Rules that control back-end processing are defined using [Server-based](#) scripts. For example, your implementation could set up a server-based script to control the processing that executes whenever a given type of adjustment is canceled.

The topics in this section describe how to configure your scripts.

## The Big Picture Of Scripts

This section describes features and functions that are shared by both BPA scripts and server-based scripts.

### Scripts Are Business Process-Oriented

To create a script, you must analyze the steps used to implement a given business process. For example, you could create a "stop auto pay" BPA script that:

- Asks the user to select the customer / taxpayer using an appropriate search page
- Asks the user to define the date on which the person would like to stop making automatic payments
- Invokes a server-based script that populates the end-date on the account's latest automatic payment instructions.

After you understand the business process, you can set up a script to mimic these steps. If the business process is straightforward (e.g., users always perform the same steps), the script configuration will be straightforward. If the business process has several logic branches, the composition of the script may be more challenging.

### A Script Is Composed Of Steps

A script contains one or more steps. For example, a "stop auto pay" BPA script might have three steps:

- Ask the user to select the customer / taxpayer using an appropriate search page
- Ask the customer the date on which they'd like to stop making automatic payments (and default the current date)
- Invoke a server-based script that, in turn, updates the account's auto pay options.

Each step references a step type. The step type controls what happens when a step executes. It might be helpful to think of a script as a macro and each step as a "line of code" in the macro. Each step's step type controls the function that is executed when the step is performed.

---

**FASTPATH:** Refer to [How To Set Up Each Step Type](#) for a detailed description of all available step types and how to set them up.

---

## Designing And Developing Scripts

Constructing a script is similar to writing a computer program. We recommend that you follow the approach outlined below when you construct scripts:

- Thoroughly understand the business process to be scripted
- Thoroughly understand how the transactions and services that your script uses work
- Design the steps for the script "on paper"
- Determine the most maintainable way to set up your scripts. Rather than creating complex, monolithic scripts, we recommend dividing scripts into smaller sections. For example:
  - For BPA scripts,

- Determine if several scripts have similar steps. If so, set up a script that contains these common steps and invoke it from the main scripts. For example, if the main script were a BPA script, this would be invoked via a **Perform script** step.
- Determine if a large script can be divided into logical sections. If so, set up a small script for each section and create a "master script" to invoke each. For example, if the main script were a BPA script, this would be invoked via a **Transfer control** step.
- For server-based script, you can segregate reusable steps into "service scripts" and then use **Invoke service script** steps to execute the common logic.
- Add the script using [Script Maintenance](#)
- Thoroughly test the script

## A Script May Declare Data Areas

Both BPA and server-based scripts may have one or more [data areas](#):

- If the script contains steps that exchange XML documents, you must declare a data area for each type of XML document. For example, if a BPA script has a step that invokes a service script, the BPA script must declare a data area that holds the XML document that is used to pass information to and from the service script.
- You can use a data area as a more definitive way to declare your temporary storage. For example, you can describe your script's temporary storage variables using a [stand-alone data area](#) schema and associate it with your script.

Various step types involve referencing the script's data areas as well as support the ability to compare and move data to and from field elements residing in the data areas.

An [Edit Data](#) step supports the syntax to dynamically declare data areas as part of the step itself. This technique eliminates the need to statically declare a data area. Refer to [Designing Generic Scripts](#) for an example of when this technique may be useful.

## Designing Generic Scripts

Scripts may be designed to encapsulate an overall procedure common across different business objects.

For example, BPA scripts may implement a standard procedure to maintain business entities in which the first step obtains the entity's data, the second step presents its associated UI map to the user for update, and the last step updates the entity. Notice that in this case the only difference from one object to another is the data to capture. Rather than designing a dedicated BPA script with static data areas and invocation steps for each business object, you can design a single generic script that dynamically invokes a business object and its associated UI map.

This functionality is available only within an [Edit Data](#) step. With this technique, the name of the schema-based object is held in a variable or an XPath schema location and is used to both declare and invoke the object.

---

**NOTE: Tips.** Refer to the tips context zone associated with the script maintenance page for more information on edit data commands and examples.

---

## Securing Script Execution

The system supports the ability to secure the execution of scripts by associating the script with an Application Service. Refer to [The Big Picture of Application Security](#) for more information. Application security is optional and applies to service scripts and user-invocable BPA scripts only. If a script is not associated with an application service, all users may execute the script. Otherwise, only users that have **Execute** access to the application service may execute the script.

# The Big Picture Of BPA Scripts

---

**FASTPATH:** Refer to [The Big Picture Of Scripts](#) to better understand the basic concept of scripts.

---

Users may require instructions in order to perform certain tasks. The business process assistant allows you to set up scripts that step a user through your business processes. For example, you might want to create scripts to help users do the following:

- Add a new person to an existing account
- Set up a customer to pay automatically
- Modify a customer who no longer wants to receive marketing information
- Modify a customer's web password
- Record a trouble order
- Merge two accounts into one account
- Fix a bill with an invalid rate
- ... (the list is only limited by your time and imagination)

Users execute these scripts via the [business process assistant](#) (BPA). Users can also define their favorite BPA scripts in their [user preferences](#). By doing this, a user can execute a script by pressing an accelerator key (Ctrl + Shift + a number).

Don't think of these scripts as merely a training tool. BPA scripts can also reduce the time it takes to perform common tasks. For example, you can write a script that reduces the "number of clicks" required to add a new person to an existing account.

---

**CAUTION:** Future upgrade issues. Although we make every effort not to remove fields or tab pages between releases, there may be times when changes made by the base-package will necessitate changes to your scripts. Please refer to the release notes for a list of any removed fields or tab pages.

---

---

**CAUTION:** Scripts are not a substitute for end-user training. Scripts minimize the steps required to perform common tasks. Unusual problems (e.g., a missing meter exchange) may be difficult to script as there are many different ways to resolve such a problem. However, scripts can point a user in the right direction and reduce the need to memorize obscure business processes.

---

The topics in this section describe background topics relevant to BPA scripts.

## How To Invoke Scripts

Refer to [Initiating Scripts](#) for a description of how end-users initiate scripts.

## Developing and Debugging Your BPA Scripts

You may find it helpful to categorize the step types into two groups: those that involve some type of activity in the [script area](#), and those that don't. The following step types cause activity in the script area: **Height**, **Display text**, **Prompt user**, **Input data**, **Input Map**, **Set focus to a field**. The rest of the step types are procedural and involve no user interaction. For debugging purposes, you can instruct the system to display text in the script area for the latter step types. Also note, for debugging purposes, you can display an entire data area (or a portion thereof) in the script area by entering `%+...+%` where ... is the name of the node whose element(s) should be displayed.

---

**NOTE: Time saver.** When you develop a new BPA script, change your [user preferences](#) to include the script as your first "favorite". This way, you can press `Ctrl+Shift+1` to invoke the script (eliminating the need to select it from the [script menu](#)).

---

## Launching A Script From A Menu

You can create menu items that launch BPA scripts rather than open a page. To do this, create a [navigation option](#) that references your script and then add a menu item that references the navigation option.

If the navigation option is referenced on a [context menu](#) and the navigation option has a "context field", a temporary storage variable will be created and populated with the unique identifier of the object in context. For example, if you add a "script" navigation option to the bill context menu and this navigation option has a context field of `BILL_ID`, the system will create a temporary storage variable called `BILL_ID` and populate it with the respective bill id when the menu item is selected.

## Launching A Script When Starting The System

You can set the system to launch a script upon startup.

For example, imagine that through an interactive voice response system, a customer has keyed in their account ID and has indicated that they would like to stop an automatic payment. At the point when the IVR system determines that the customer must speak to a user, the interface can be configured to launch the application. When launched it passes the script name and account ID. It can also pass a [navigation option](#) to automatically load the appropriate page (if this information is not part of the script).

To do this, parameters are appended to the standard system URL. The following parameters may be supplied:

- `script=<scriptname>`
- `ACCT_ID=<account id>`
- `location=<navigation option>`

Parameters are added to the standard system URL by appending a question mark (?) to the end and then adding the "key=value" pair. If you require more than one parameter, use an ampersand (&) to separate each key=value pair.

For example, the following URLs are possible ways to launch the **StopAutoPay** script at startup, assuming your standard URL for launching the system is `http://system-server:1234/cis.jsp`:

- `http://system-server:1234/cis.jsp?script=StopAutoPay`
- `http://system-server:1234/cis.jsp?script=StopAutoPay&ACCT_ID=1234512345`
- `http://system-server:1234/cis.jsp?script=StopAutoPay&ACCT_ID=1234512345&location=accountMaint`

It doesn't matter in which order the parameters are provided. The system processes them in the correct order. For example, the following examples are processed by the system in the same way:

- `http://system-server:1234/cis.jsp?ACCT_ID=1234512345&script=StopAutoPay&location=accountMaint`
- `http://system-server:1234/cis.jsp?ACCT_ID=1234512345&location=accountMaint&script=StopAutoPay`

These parameters are kept in a common area accessible by any script for the duration of the session. To use these parameters on a script you may reference the corresponding `%PARM-<name>` [global variables](#). In this example, after the system is launched any script may have access to the above account ID parameter value by means of the `%PARM-ACCT_ID` global variable. Also note, these parameters are also loaded into temporary storage (to continue the example, there'd also be a temporary storage variable called `ACCT_ID` that holds the passed value).



## Executing A Script When A To Do Entry Is Selected

The system creates [To Do entries](#) to highlight tasks that require attention (e.g., records in error). Users can complete many of these tasks without assistance. However, you can set up the system to automatically launch a script when a user selects a To Do entry. For example, consider a To Do entry that highlights a bill that's in error due to an invalid mailing address. You can set up the system to execute a script when this To Do entry is selected by a user. This script might prompt the user to first correct the customer's default mailing address and then re-complete the bill.

The following points provide background information to help you understand how to implement this functionality:

- Every To Do entry references a [To Do type](#) and a [message category and number](#). The To Do type defines the category of the task (e.g., bill errors). The message number defines the specific issue (e.g., a valid address can't be found.). Refer to [The Big Picture of System Messages](#) for more information about message categories and numbers.
- When a user drills down on a To Do entry, either a script launches OR the user is transferred to the transaction associated with the entry's To Do type. You control what happens by configuring the To Do type accordingly:
  - If you want to launch a script when a user drills down on an entry, you link the script to the [To Do type and message number](#). Keep in mind that you can define a different script for every message (and some To Do types have many different messages).
  - If the system doesn't find a script for an entry's To Do type and message number, it transfers the user to the To Do type's default transaction.

---

**NOTE: How do you find the message numbers?** We do not supply documentation of every To Do type's message numbers (this list would be impossible to maintain and therefore untrustworthy). The best way to determine which message numbers warrant a script is during pre-production when you're testing the system. During this period, To Do entries will be generated. For those entries that warrant a script, simply display the entry on [To Do maintenance](#). On this page, you will find the entry's message number adjacent to the description.

---

- These types of scripts invariably need to access data that resides on the selected To Do entry. Refer to [How To Use To Do Fields](#) for the details.

## The Big Picture Of Script Eligibility Rules

You can configure [eligibility criteria](#) on the scripts to limit the scripts that a user sees in the [script search](#). For example, you could indicate a script should only appear on the script menu if the user belongs to the level 1 customer service representative group. You may also indicate that a script should only appear if the data a user is viewing has certain criteria. For example, if you are using Oracle Utilities Customer Care and Billing, you can indicate that a script should only appear if the current account's customer class is residential. By doing this, you avoid presenting the user with scripts that aren't applicable to the current data in context or the user's role.

The topics in this section describe eligibility rules.

## Script Eligibility Rules Are Not Strictly Enforced

The [script search](#) gives a user a choice of seeing all scripts or only scripts that are eligible (given the current data in context and their user profile). This means that it's possible for a script that isn't eligible for the given context data / user to be executed via this search. In other words, the system does not strictly enforce a script's eligibility rules.

It might be more helpful to think of eligibility rules as "highlight conditions". These "highlight conditions" simply control whether the script appears in the [script search](#) when a user indicates they only want to see eligible scripts.



## You Can Mark A Script As Always Eligible

If you don't want to configure eligibility rules, you don't have to. Simply indicate that the script is always eligible.

## You Can Mark A Script As Never Eligible

If you have scripts that you do not want a user to select from the script menu, indicate that it is never eligible. An example of a script that you wouldn't want a user to select from the menu is one that is [launched when a To Do entry is selected](#). These types of scripts most likely rely on data linked to the selected To Do entry. As a result, a user should only launch scripts of this type from the To Do entry and not from the script menu.

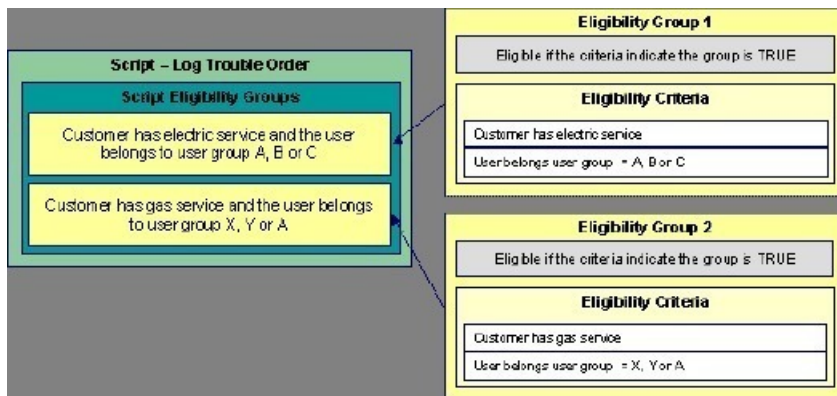
## Criteria Groups versus Eligibility Criteria

Before we provide concrete examples of eligibility criteria, we need to explain two concepts: Criteria Groups and Eligibility Criteria. A script's criteria groups control whether a user is eligible to choose a script. At a high level, it works like this:

- A criteria group has one or more eligibility criteria. A group's criteria control whether the group is considered true or false.
- When you create a group, you define what should happen if the group is true or false. You have the following choices:
  - The user is eligible to choose the script
  - The user is not eligible to choose the script
  - The next group should be checked

We'll use the following example from Oracle Utilities Customer Care and Billing to help illustrate these points. Assume a script is only eligible if:

- The customer has electric service and the user belongs to user group A, B or C
- OR, the customer has gas service and the user belongs to user group X, Y or A



This script requires two eligibility groups because it has two distinct conditions:

- IF (Customer has electric service AND (User belongs to user group A, B or C))
- IF (Customer has gas service AND (User belongs to user group X, Y or A))

If either condition is true, the script is eligible.

You would need to set the following criteria groups in order to support this requirement:

Group No.	Group Description	If Group is True	If Group is False
-----------	-------------------	------------------	-------------------

1	Customer has electric service and the user belongs to user group A, B or C	<b>Eligible</b>	<b>Check next group</b>
2	Customer has gas service and the user belongs to user group X, Y or A	<b>Eligible</b>	<b>Ineligible</b>

The following criteria are required for each of the above groups:

Group 1: Customer has electric service and the user belongs to user group A, B or C				
Seq	Logical Criteria	If Eligibility Criteria is True	If Eligibility Criteria is False	If Insufficient Data
10	Customer has electric service	<b>Check next condition</b>	<b>Group is false</b>	<b>Group is false</b>
20	User belongs to user group A, B or C	<b>Group is true</b>	<b>Group is false</b>	<b>Group is false</b>

Group 2: Customer has gas service and the user belongs to user group X, Y or A				
Seq	Logical Criteria	If Eligibility Criteria is True	If Eligibility Criteria is False	If Insufficient Data
10	Customer has gas service	<b>Check next condition</b>	<b>Group is false</b>	<b>Group is false</b>
20	User belongs to user group X, Y or A	<b>Group is true</b>	<b>Group is false</b>	<b>Group is false</b>

The next section describes how you might configure the specific logical criteria in each of the groups.

## Defining Logical Criteria

When you set up an eligibility criterion, you must define two things:

- The field to be compared
- The comparison method

You have the following choices in respect of identifying the *field to be compared* :

- You can execute an algorithm to retrieve a field value from somewhere else in the system.
- Some products may support choosing a characteristic linked to an appropriate object in the system (such as an account or person).

You have the following choices in respect of identifying the *comparison method*:

- You can choose an operator (e.g., >, <, =, BETWEEN, IN, etc.) and a comparison value.
- You can execute an algorithm that performs the comparison (and returns True, False or Insufficient Data). This is also a very powerful feature, but it's not terribly intuitive. We'll present a few examples later in this section to illustrate the power of this approach.

The [Examples Of Script Eligibility Rules](#) provide examples to help you understand this design.

## Examples Of Script Eligibility Rules

The topics in this section provide examples about how to set up script eligibility rules.

### A Script With A Time Span Comparison

A script that is only eligible for senior citizens has the following eligibility rules:

- Customer class = Residential

- Birth date equates to that of a senior citizen

These rules require only one eligibility group on the script. It would look as follows:

Group No.	Group Description	If Group is True	If Group is False
1	Residential and Senior Citizen	<b>Eligible</b>	<b>Ineligible</b>

The following criteria will be required for this group:

Group 1: Residential, Calif, Senior					
Seq	Field to Compare	Comparison Method	If True	If False	If Insufficient Data
10	Algorithm: retrieve account's customer class	= R	<b>Check next condition</b>	<b>Group is false</b>	<b>Group is false</b>
30	Person characteristic: Date of Birth	Algorithm: True if senior	<b>Group is true</b>	<b>Group is false</b>	<b>Group is false</b>

The first criterion is easy; it calls an algorithm that retrieves a field on the current account. This value, in turn, is compared to a given value. If the comparison results in a True value, the next condition is checked. If the comparison doesn't result in a True value, the **Group is false** (and, the group indicates that if the group is false, the script isn't eligible). Refer to SECF-ACCTFLD in the product documentation for an example of an algorithm type that retrieves a field value from an account.

The last criterion contains a time span comparison. Time span comparisons are used to compare a date to something. In our example, we have to determine the age of the customer based on their birth date. If the resultant age is > 65, they are considered a senior citizen. To pull this off, you can take advantage of a comparison algorithm supplied with the base script as described below.

- Field to Compare. The person characteristic in which the customer's birth date is held is selected.
- Comparison Method. We chose a comparison algorithm that returns a value of **True** if the related field value (the customer's date of birth) is greater than 65 years (refer to [SECC-TIMESPN](#) for an example of this type of algorithm).

You'll notice that if a value of **True** is returned by the **True if senior** algorithm, the group is true (and we've set up the group to indicate a true group means the script is eligible).

**NOTE: The time span algorithm can be used to compare days, weeks, months, etc.** Refer to [SECC-TIMESPN](#) for more information about this algorithm.

## A Script With Service Type Comparison

Imagine a script that is only eligible if the current customer has gas service and the user belongs to user groups A, B or C. This script would need the following eligibility rules:

- Customer has gas service
- User belongs to user group A, B, or C

These rules require only one eligibility group on the script. It would look as follows:

Group No.	Group Description	If Group is True	If Group is False
1	Has gas service and user is part of user group A, B or C	<b>Eligible</b>	<b>Ineligible</b>

The following criteria are required for this group:

Group 1: Has gas service and user is part of user group A, B, or C					
Seq	Field to Compare	Comparison Method	If True	If False	If Insufficient Data
10	Algorithm: check if customer has gas service	= True	<b>Check next condition</b>	<b>Group is false</b>	<b>Group is false</b>

Algorithm: check if  
user belongs to user  
group A, B or C

= True

Group is true

Group is false

Group is false

---

Both criteria are similar - they call an algorithm that performs a logical comparison. These algorithms are a bit counter intuitive (but understanding them provides you with another way to implement complex eligibility criteria):

The first criterion works as follows:

- **Field to Compare.** We chose a "field to compare" algorithm that checks if the current account has service agreements that belong to a given set of service types. It returns a value of **True** if the customer has an active service agreement that matches one of the service types in the algorithm. In our example, the "check if customer has gas service" algorithm returns a value of **True** if the customer has at least one active service agreement whose SA type references the gas service type. The "check if customer has electric service" algorithm is almost identical, only the service type differs.
- **Comparison Method.** We simply compare the value returned by the algorithm to True and indicate the appropriate response.

The second criterion works similarly:

- **Field to Compare.** We chose a "field to compare" algorithm that checks if the user belongs to any user group in a set of user groups. It returns a value of **True** if the user belongs to at least one user group defined in parameters of the algorithm. Refer to [SECF-USRNGRP](#) for an example of this type of algorithm.
- **Comparison Method.** We simply compare the value returned by the algorithm to True and indicate the appropriate response.

---

**NOTE: Bottom line.** The "field to compare" algorithm isn't actually returning a specific field's value. Rather, it's returning a value of **True** or **False**. This value is in turn, compared by the "comparison method" and the group is set to true, false or check next accordingly.

---

## The Big Picture Of Server-Based Scripts

---

**FASTPATH:** Refer to [The Big Picture Of Scripts](#) to better understand the basic concept of scripts.

---

Server-based scripts allow an implementation to configure backend business processes. The system supports two types of server-based scripts, **Plug-In** scripts and **Service** scripts.

- **Plug-in scripts** allow an implementation to develop routines that are executed from the system's various plug-in spots without coding. For example, an implementation could configure a plug-in script that is executed every time an adjustment of a given type is frozen.
- **Service scripts** allow an implementation to develop common routines that are invoked from both front-end and back-end services. For example, an implementation could create a service script that terminates an account's automatic payment preferences. This service script could be invoked from a BPA script initiated by an end-user when a customer asks to stop paying automatically, and it could also be executed from a plug-in script if a customer fails to pay their debt on time.

The topics in this section describe background topics relevant to server-based scripts.

## Plug-In Scripts

---

**NOTE:** This section assumes you are familiar with the notion of plug-in spots (algorithm entities) and plug-ins. See [The Big Picture Of Algorithms](#) for more information.

---

Rather than write a java program for a plug-in spot, you can create a plug-in using the scripting "language". In essence, this is the only difference between a program-based plug-in and a script-based one. Obviously, this is a significant difference as it allows you to implement plug-ins without programming (and compilers).

The following topics describe basic concepts related to plug-in scripts.

## A Plug-In Script's API

Like program-based plug-ins, plug-in scripts:

- Run on the application server
- Have their API (input / output interface) defined by the plug-in spot (i.e., plug-in scripts don't get to declare their own API)
- Can only be invoked by the "plug-in spot driver"

The best way to understand a plug-in script's API is to use the **View Script Schema** hyperlink to view its parameters data area schema.

```
<schema>
  <parm type="group">
    <soft type="list">
      <value/>
    </soft>
    <hard type="group">
      <action use="input"/>
      <businessObject type="group" use="input">
        <id/>
      </businessObject>
    </hard>
  </parm>
</schema>
```

Notice the two groups: soft and hard. If you are familiar with plug-in spots, you'll recognize these as the classic soft and hard parameters:

- The **soft** parameters are the values of the parameters defined on the algorithm. Notice they are not named - if you want to reference them in your plug-in script, you must do it by position (this is equivalent to what a Java programmer does for plug-ins written in Java).
- The **hard** parameters are controlled by the plug-in spot (i.e., the algorithm entity). Notice that this plug-in spot has a single input parameter called "**businessObject/id**". Also notice the **use=** attribute - this shows that this parameter is input-only (i.e., you can't change it in the plug-in script).

---

**NOTE: XPath.** You can click on an element name to see the XPath used to reference the element in your script.

---

## Setting Up Plug-In Scripts

You can write plug-in scripts for all plug-in spots that have been Java-enabled. The following points describe how to implement a plug-in script:

- Compose your plug-in [script](#), associating it with the appropriate algorithm entity (plug-in spot).
- Create a new algorithm type for the respective algorithm entity, referencing your plug-in script as the program to carry out the algorithm type's function. Only plug-in scripts associated with the algorithm entity may be referenced on the algorithm type.
- Set up an algorithm for the respective algorithm type and plug it in where applicable. Refer to [Setting Up Algorithm Types](#) for more information.

---

**NOTE:** No plug-in scripts are shipped with the base-package.

---

## Service Scripts

BPA scripts run on the client's browser and guide the end-users through business processes. Service scripts run on the application server and perform server-based processing for [BPA scripts](#), [zones](#) and more. You may want to think of a service script as a common routine that is set up via the scripting (rather than programming).

The following topics describe basic concepts related to service scripts.

### A Service Script's API

As with any common routine, a service script must declare its input / output parameters (i.e., its API). A service script's API is defined on its [schema](#).

---

**NOTE: Schema Definition Tips.** A context sensitive "Script Tips" zone appears when you open the [Script](#) page to assist you with the schema definition syntax. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

---

### Invoking Service Scripts

Any type of script may [invoke a service script](#):

- A BPA script may invoke a service script to perform server-based processing.
- Plug-in scripts may invoke a service script (like a "common routine").
- A service script may call another service script (like a "common routine").

Map zones may be configured to invoke service scripts to obtain the data to be displayed. Refer to [Map Zones](#) for more information.

[Inbound web services](#) support interaction with service scripts allowing the outside world to interact directly with a service script.

You can also invoke a service script from a Java class.

### Debugging Server-Based Scripts

The server can create log entries to help you debug your server scripts. These logs are only created if you do the following:

- Start the system in `?debug=true` mode
- Turn on the **Global debug** checkbox in the upper corner of the browser. When you click this check box, a pop-up appears asking you what you want to trace - make sure to check box that causes script steps to be traced.

The logs contain a great deal of information including the contents of the referenced data area for **Move data**, **Invoke business object**, **Invoke business service** and **Invoke service script** steps.

You can view the contents of the logs by pressing the **Show User Log** button appears at the top of the browser.

Please note that all log entries for your user ID are shown (so don't share user IDs).

### Maintaining Scripts

The script maintenance transaction is used to maintain your scripts. The topics in this section describe how to use this transaction.

---

**FASTPATH:** Refer to [The Big Picture Of Scripts](#) for more information about scripts.

---

## Script - Main

Use this page to define basic information about a script. Open this page using **Admin > System > Script**.

---

**NOTE: Script Tips.** A context sensitive "Script Tips" zone is associated with this page. The zone provides useful tips on various topics related to setting up scripts.

---

### Description of Page

Enter a unique **Script** code and **Description** for the script. Use the **Detailed Description** to describe the purpose of this script in detail. **Owner** indicates if the script is owned by the base package or by your implementation (**Customer Modification**).

---

**CAUTION:** Important! If you introduce a new script, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

**Script Type** indicates if this is a **BPA Script**, **Plug-In Script** or **Service Script**. Refer to [The Big Picture Of BPA Scripts](#) and [The Big Picture Of Server Based Scripts](#) for more information.

**Accessibility Option** appears only for BPA scripts. Set this value to **Accessible from Script Menu** for any script that may be launched as a stand-alone script. Scripts with this configuration may be linked to a navigation option so that they may be invoked from a menu and may be configured by a user as a favorite script. Set this value to **Not Accessible from Script Menu** for any script that cannot be launched on its own. For example, any script that is invoked as a sub-script from another script should have this setting. In addition, any script that is designed to be launched from within a specific portal where certain data is provided to the script should include this setting.

Enter an **Application Service** if the execution of the script should be secured. The application service should include **Execute** as one of its access modes. Refer to [Securing Script Execution](#) for more information.

**Algorithm Entity** appears only for **plug-in scripts**. Use this field to define the **algorithm entity** into which this script can be plugged in.

**Business Object** appears only for business object related plug-in scripts. Enter the **Business Object** whose elements are to be referenced by the plug-in script.

**Script Engine Version** defines the version of the XML Path Language (XPath) to be used for the script. Script engine versions 2 and 3 use the XPath 2 engine supplied by the XQuery team. This is the same engine used inside the Oracle database. The current script engine version 3 is a modified version that offers performance improvements without impacting existing version 2 scripts.

**Notes regarding Script engine version 1:** The XPath library used is Jaxen; for BPA scripts, use IE's built-in MSXML parser; Xpath 1 (and even JavaScript) uses floating point arithmetic, which means that adding a collection of numbers with two decimal places might end up with a value of 10779.079999999998 instead of 10779.08.

Click the **View Script Schema** to view the [script's data areas](#) on the [schema viewer](#) window.

Click the **View XSD** hyperlink to view a service script's schema definition in XSD format.

The **View Script As Text** hyperlink appears for server-based scripts only. Click this link to view the internal scripting commands in a separate window. The presented script syntax is valid within [edit data](#) steps.

The [tree](#) summarizes the script's steps. You can use the hyperlink to transfer you to the **Step** tab with the corresponding step displayed.



## Script - Step

Use this page to add or update a script's steps. Open this page using **Admin > System > Script** and then navigate to the **Step** tab.

---

**NOTE: Time saver.** You can navigate to a step by clicking on the respective node in the tree on the Main tab.

---

### Description of Page

The **Steps** accordion contains an entry for every step linked to the script. When a script is initially displayed, its steps are collapsed. To see a step's details, simply click on the step's summary bar. You can re-click the bar to collapse the step's details. Please see [accordions](#) for the details of other features you can use to save time.

Select the **Step Type** that corresponds with the step. Refer to [How To Set Up Each Step Type](#) for an overview of the step types.

---

**CAUTION:** The Step Type affects what you can enter on other parts of this page. The remainder of this section is devoted to those fields that can be entered regardless of Step Type. The subtopics that follow describe those fields whose entry is contingent on the Step Type.

---

**StepSequence** defines the relative position of this step in respect of the other steps. The position is important because it defines the order in which the step is executed. You should only change a Step Sequence if you need to reposition this step. But take care; if you change the Step Sequence and the step is referenced on other steps, you'll have to change all of the referencing steps.

---

**NOTE: Leave gaps in the sequence numbers.** Make sure you leave space between sequence numbers so that you can add new steps between existing ones in the future. If you run out of space, you can use the **Renumber** button to renumber all of the steps. This will renumber the script's steps by 10 and change all related references accordingly.

---

**Display Step** is only enabled on BPA scripts for step types that typically don't cause information to be displayed in the [script area](#) (i.e., step types like **Conditional Branch**, **Go to a step**, **Height**, etc). If you turn on this switch, information about the step is displayed in the script area to help you debug the script.

---

**CAUTION:** Remember to turn this switch off when you're ready to let users use this script.

---

**NOTE:** If **Display Step** is turned on and the step has **Text**, this information will be displayed in the script area. If **Display Step** is turned on and the step does NOT have **Text**, a system-generated messages describing what the step does is displayed in the script area.

---

**Display Icon** controls the [icon](#) that prefixes the **Text** that's displayed in the script area. Using an icon on a step is optional. This field is only applicable to BPA scripts.

**Text** is the information that displays in the [script area](#) when the step executes. You need only define text for steps that cause something to display in the script area.

---

**FASTPATH:** Refer to [How To Substitute Variables In Text](#) for a discussion about how to substitute variables in a text string.

---

**FASTPATH:** Refer to [How To Use HTML Tags And Spans In Text](#) for a discussion about how to format (with colors and fonts) the text that appears in the script area.

---

The other fields on this page are dependent on the **Step Type**. The topics that follow briefly describe each step type's fields and provide additional information about steps.



Click on the **View Script Schema** hyperlink to view the script's data areas. Doing this opens the [schema viewer](#) window.

The **View Script As Text** hyperlink appears for server-based scripts only. Click this link to view the internal scripting commands in a separate window. The presented script syntax is valid within [edit data](#) steps.

## How To Set Up Each Step Type

The contents of this section describe how to set up each type of step.

## Common Step Types To All Script Types

The contents of this section describe common step types applicable to all script types.

## How To Set Up Conditional Branch Steps

**Conditional branch** steps allow you to conditionally jump to a different step based on logical criteria. For example, you could jump to a different step in a script if the customer is residential as opposed to commercial. In addition, several fields are required for **Conditional Branch** steps:

**Compare Field Type** and **Compare Field Name** define the first operand in the comparison. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Current To Do Information.** Use this field type when the field being compared resides on the current To Do entry. Refer to [How To Use To Do Fields](#) for instructions on how to define the appropriate **Field Name**.
- **Data Area.** Use this field type when the field being compared is one that you put into one of the scripts data areas in an earlier step. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to [How To Reference Fields In Data Areas](#) for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model.** Use this field type when the field being compared resides on one of the tab pages in the [object display area](#). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Predefined Value.** Use this field type when the field being compared is a [global variable](#).
- **Temporary Storage.** Use this field type when the field being compared is one that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field.** Use this field type when the field being compared resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

**Condition** defines the comparison criteria:

- Use **>**, **<**, **=**, **>=**, **<=**, **<>** (not equal) to compare the field using standard logical operators. Enter the comparison value using the following fields.
- Use **IN** to compare the first field to a list of values. Each value is separated by a comma. For example, if a field value must equal **1**, **3** or **9**, you would enter a comparison value of **1,3,9**.
- Use **BETWEEN** to compare the field to a range of values. For example, if a field value must be between **1** and **9**, you would enter a comparison value of **1,9**. Note, the comparison is inclusive of the low and high values.

**Comparison Field Type**, **Comparison Field Name** and **Comparison Value** define what you're comparing the first operand to. The following points describe each field type:

- **Current To Do Information.** Use this field type when the comparison value resides on the current To Do entry. Refer to [How To Use To Do Fields](#) for instructions on how to define the appropriate **Field Name**.

- **Data Area.** Use this field type when the comparison value resides in one of the scripts data areas. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to [How To Reference Fields In Data Areas](#) for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model.** Use this field type when the comparison value resides on one of the tab pages in the [object display area](#). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Predefined Value.** Use this field type when the field being compared is a constant value defined in the script. When this field type is used, use **Comparison Value** to define the constant value. Refer to [How To Use Constants In Scripts](#) for instructions on how to use constants.
- **Temporary Storage.** Use this field type when the comparison value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field.** Use this field type when the comparison value resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

---

**NOTE: Conditional field types.** The field types **Current To Do Information**, **Page Data Model** and **User Interface Field** are only applicable to BPA scripts.

---

The above fields allow you to perform a comparison that results in a value of **TRUE** or **FALSE**. The remaining fields control the step to which control is passed given the value:

- **If TRUE, Go to** defines the step that is executed if the comparison results in a **TRUE** value.
- **If FALSE, Go to** defines the step that is executed if the comparison results in a **FALSE** value.

---

**NOTE: Numeric Comparison.** Comparison of two values may be numeric or textual (left-to-right). Numeric comparison takes place only when values on both side of the comparison are recognized as numeric by the system. Otherwise, textual comparison is used. Fields for **Current To Do Information**, **Data Area**, **Page Data Model**, and **User Interface Field** types are explicitly associated with a data type and therefore can be recognized as numeric or not. This is not the case for fields residing in **Temporary Storage** or those set as **Predefined Values**. A **Temporary Storage** field is considered numeric if it either holds a numeric value moved to it from an explicitly defined numeric value (see above) or it is a resultant field of mathematical operation. A **Predefined Value** field is considered numeric if the other field it is compared to is numeric. For example, if a numeric field is compared to a **Predefined Value** the latter is considered numeric as well resulting in numeric value comparison. However, if the two fields are defined as **Predefined Values** the system assumes their values are text strings and therefore applies textual comparison.

---

## How To Set Up Edit Data Steps

**Edit data** steps provide a free format region where you can specify commands to control your script processing.

In general, the syntax available within edit data mimics the commands available within the explicit step types. However, there are a few commands that are available only within edit data. For example, the two structured commands: **For**, and **If**.

For server-based scripts, you may find it useful to create a few explicit step types and then use the **View Script as Text** hyperlink on the [Script - Step](#) page to better understand the edit data syntax.

---

**NOTE:** Not all BPA step types are supported using the edit data syntax. **Tips.** Refer to the tips context zone associated with the script maintenance page for more information on edit data commands and examples.

---

Additional field required for **Edit data** steps:

Enter your scripting commands in the **Edit Data Text** field.

## How To Set Up Go To Steps

**Go to** steps allow you to jump to a step other than the next step. Additional fields required for **Go To** steps:

**Next Step** defines the step to which the script should jump.

## How To Set Up Invoke Business Object Steps

**Invoke business object** steps allow you to interact with a [business object](#) in order to obtain or maintain its information.

The following additional fields are required for **Invoke business object** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

**Group Name** references the [data area](#) to be passed to and from the server when communicating with the **Business Object**. Indicate the **Action** to be performed on the object when invoked. Valid values are **Add, Delete, Fast Add (No Read), Fast Update (No Read), Read, Replace, Update**.

---

**NOTE: Performance note.** The actions **Fast Add** and **Fast Update** should be used when the business object's data area does not need to be re-read subsequent to the **Add** or **Update** action. In other words, the **Add** and **Update** actions are equivalent to **Fast Add + Read** and **Fast Update + Read**.

---

The business object call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script, to determine the step to which control is passed given the outcome of the call.

**If Success, Go To** defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

**If Error, Go To** defines the step that is executed if the call returns on error. Please note that the error information is held in [global variables](#). This field is only applicable to BPA scripts.

---

**NOTE: Error technique.** Let's assume a scenario where a business object is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

---

## How To Set Up Invoke Business Service Steps

**Invoke business service** steps allow you to interact with a [business service](#).

The following additional fields are required for **Invoke business service** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

**Group Name** references the [data area](#) to be passed to and from the server when the **Business Service** is invoked.

The business service call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script, to determine the step to which control is passed given the outcome of the call.

**If Success, Go To** defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

**If Error, Go To** defines the step that is executed if the call returns on error. Please note that the error information is held in [global variables](#). This field is only applicable to BPA scripts.

---

**NOTE: Error technique.** Let's assume a scenario where a business service is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

---

## How To Set Up Invoke Service Script Steps

**Invoke service script** steps allow you to execute a [service script](#).

The following additional fields are required for **Invoke service script** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

**Group Name** references the [data area](#) to be passed to and from the server when the **Service Script** is invoked.

The service script call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script to determine the step to which control is passed given the outcome of the call.

**If Success, Go To** defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

**If Error, Go To** defines the step that is executed if the call returns on error. Please note that the error information is held in [global variables](#). This field is only applicable to BPA scripts.

---

**NOTE: Error technique.** Let's assume a scenario where a service script is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

---

## How To Set Up Label Steps

**Label** steps allow you to describe what the next step(s) are doing. Steps of this type are helpful to the script administrators when reviewing or modifying the steps in a script, especially when a script has many steps. When designing a script, the label steps enable you to provide a heading for common steps that belong together. The script tree displays steps of this type in a different color (green) so that they stand out from other steps.

There are no additional fields for **Label** steps.

## How To Set Up Move Data Steps

**Move data** steps allow you to move data (from a source to a destination). The following additional fields are required for **Move data** steps:

**Source Field Type**, **Source Field Name** and **Source Field Value** define what you're moving. The following points describe each field type:

- **Context Variable.** Use this field type in a plug-in or service script if the source value is a variable initiated in a higher level script.
- **Current To Do Information.** Use this field type when the source value resides on the current To Do entry. Refer to [How To Use To Do Fields](#) for instructions on how to define the appropriate **Field Name**.
- **Data Area.** Use this field type when the field being compared is one that you put into one of the script's data areas in an earlier step. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to [How To Reference Fields In Data Areas](#) for instructions on how to construct the appropriate **Field Name**.

- **Global Context.** Use this field type in a BPA script when the source value is a global variable.
- **Page Data Model.** Use this field type in a BPA script when the source value resides on any of the tab pages in the [object display area](#) (i.e., the source field doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Portal Context.** Use this field type when the source value is a variable in the portal context.
- **Predefined Value.** Use this field type when the source value is a constant value defined in the script. When this field type is used, use **Source Field Value** to define the constant value. Refer to [How To Use Constants In Scripts](#) for instructions on how to use constants.

---

**NOTE: Concatenating fields together.** You can also use **Predefined Value** if you want to concatenate two fields together. For example, let's say you have a script that merges two persons into a single person. You might want this script to change the name of the person being merged out of existence to include the ID of the person remaining. In this example, you could enter a **Source Field Value** of **%ONAMEmerged into person %PERID** (where **ONAME** is a field in temporary storage that contains the name of the person being merged out of existence and **PERID** contains the ID of the person being kept). Refer to [How To Substitute Variables In Text](#) for a description of how you can substitute field values to compose the field value.

---

- **Temporary Storage.** Use this field type when the source value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field.** Use this field type when the source value resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

**Destination Field Type** and **Destination Field Name** define where the source field will be moved. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Context Variable.** Use this field type in your plug-in or service script if you use a variable to communicate information to a lower level service script or schema.
- **Data Area.** Use this field type when the destination field resides on one of the scripts data areas. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to [How To Reference Fields In Data Areas](#) for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model.** Use this field type when the destination field resides on any of the tab pages in the [object display area](#) (i.e., the field populated doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Portal Context.** Use this field type in a BPA script when the destination to be updated is in the current portal context.
- **Temporary Storage.** Use this field type when the destination field resides in temporary storage. Use **Field Name** to name the field in temporary storage. Use **Field Name** to name the field in temporary storage. Refer to [How To Name Temporary Storage Fields](#) for more information.
- **User Interface Field.** Use this field type when the destination field resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

---

**NOTE: Conditional field types.** The field types **Current To Do Information**, **Page Data Model** and **User Interface Field** are only applicable to BPA scripts.

---

## How To Set Up Terminate Steps

**Terminate** steps cause a server-based script to end processing successfully or issue an error.

The following additional fields are required for **Terminate** steps:

**Error** indicates whether an error should be thrown or not. If error, **Error Data Text** must be specified, indicating the error message and any message substitution parameters. Refer to the tips zone associated with the Script page for the actual syntax of initiating an error message.

---

**NOTE:** The ability to terminate a step in error is only supported for server-based scripts.

---

## Step Types Applicable to BPA Scripts only

The contents of this section describe step types that are only applicable to BPA scripts.

### How To Set Up Display Text Steps

**Display text** steps cause a text string to be displayed in the script area. Steps of this type can be used to provide the user with guidance when manual actions are necessary. In addition, they can be used to provide confirmation of the completion of tasks.

The information you enter in the **Text** field is displayed in the **script area** when the step is executed.

The text string can contain **substitution variables** and **HTML formatting commands**. Also note that for debugging purposes, you can display an entire data area (or a portion thereof) by entering **%+...+%** where **...** is the name of the node whose element(s) should be displayed.

---

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

### How To Set Up Height Steps

**Height** steps are used to change the height of the script area to be larger or smaller than the standard size.

The following additional fields are required for **Height** steps:

**Script Window Height** defines the number of **Pixels** or the **Percentage** (according to the **Height Unit**) that the script window height should be adjusted. The percentage indicates the percentage of the visible screen area that the script area uses. For example, a percentage value of **100** means that the script area will use the entire area.

---

**NOTE: Standard Number of Pixels.** The default number of pixels used by the script area is **75**.

---

**NOTE: Adjust script height in the first step.** If you want to adjust the height of the script area, it is recommendation to define the **height** step type as your first step. Otherwise, the script area will open using the standard height and then readjust, causing the screen to redisplay.

---

**NOTE: Hide script area.** You could use this type of step to set the height to **0** to hide the script area altogether. This is useful if the script does not require any prompting to the user. For example, perhaps you define a script to take a user to a page and with certain data pre-populated and that is all.

---

**NOTE: Automatically close script area.** If you want the script area to close when a script is completed, you could define the final step type with a height of **0**.

---

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---



## How To Set Up Input Data Steps

**Input data** steps cause the user to be prompted to populate an input field in the script area. The input value can be saved in a field on a page or in temporary storage. A **Continue** button always appears adjacent to the input field. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Input Data** steps:

**Destination Field Type** and **Destination Field Name** define where the input field will be saved. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Page Data Model.** Use this field type to put the input field into a field that resides on any of the tab pages in the [object display area](#) (i.e., the field populated doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Temporary Storage.** Use this field type to put the input field into temporary storage. Use **Field Name** to name the field in temporary storage. Refer to [How To Name Temporary Storage Fields](#) for more information.
- **User Interface Field.** Use this field type to put the input field into a field that resides on the currently displayed tab page. Note, if you want to execute underlying default logic, you must populate a **User Interface Field**. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

The **Prompt Values** grid may be used to define additional buttons. A separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button. Refer to [How To Substitute Variables In Text](#) for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons.
- **Next Script Step** defines the step to execute if the user clicks the button.

---

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

## How To Set Up Invoke Function Steps

---

**NOTE:** Functions were implemented prior to the introduction of business services (BS), service scripts (SS) and business objects (BO). The functionality is still supported, but the recommendation for implementations going forward is to use a step that invokes one of the above configuration tool objects in a script rather than defining a function.

---

**Invoke function** steps may be used to retrieve or update data independent of the page currently being displayed. For example, if you design a script that takes different paths based on the customer's customer class, you could invoke a function to retrieve the customer's customer class.

---

**FASTPATH:** You must set up a function before it can be referenced in a script. Refer to [Maintaining Functions](#) for the details.

---

The following additional fields are required for **Invoke Function** steps:

**Function** defines the name of the function. The function's **Long Description** is displayed below.

When a function is invoked, it will either be successful or return an error. The next two fields control the step to which control is passed given the outcome of the function call:

- **If Success, Go to** defines the step that is executed if the function is successful.

- **If Error, Go to** defines the step that is executed if the function returns on error. Refer to [How To Use Constants In Scripts](#) for a list of the global variables that are populated when a function returns an error.

---

**NOTE: Error technique.** If a function returns an error, we recommend that you invoke a step that transfers control to a script that displays the error message information and stops (note, the error information is held in [global variables](#)). You would invoke this script via a **Transfer Control**.

---

The **Send Fields** grid defines the fields whose values are sent to the function and whose field value source is not **Defined On The Function**. For example, if the function receives an account ID, you must define the name of the field in the script that holds the account ID.

- **Field** contains a brief description of the field sent to the function.
- **Source Field Type** and **Mapped Field / Value** define the field sent to the function. Refer to the description of Source Field under [How To Set Up Move Data Steps](#) for a description of each field type.
- **Comments** contain information about the field (this is defined on the function).

The **Receive Fields** grid defines the fields that hold the values returned from the function. For example, if the function returns an account's customer class and credit rating, you must set up two fields in this grid.

- **Field** contains a brief description of the field returned from the function.
- **Destination Field Type** and **Mapped Field** define the field returned from the function. Refer to the description of Destination Field under [How To Set Up Move Data Steps](#) for a description of each field type.
- **Comments** contain information about how the field (this is defined on the function).

---

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

## How To Set Up Invoke Map Steps

**Invoke map** steps are used to invoke a [UI Map](#) to display, capture and update data using an HTML form. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Invoke map** steps:

**Group Name** references the [data area](#) to be passed to and from the server when rendering the HTML form associated with the **Map**.

Use **Target Area** to designate where the map will be presented.

- Select **BPA Zone** if the map should be presented within the [script area](#).
- Select **Page Area** if the map should be presented in the [object display area](#), i.e. the frame typically used to house a maintenance page.
- Select **Pop-up Window** if the map should be launched in a separate window.

The **Returned Values** grid contains a row for every button defined on the map.

- **Returned Value** is the value returned when the user clicks the button.
- **Use as Default** can only be turned on for one entry in the grid. If this is turned on, this value's Next Script Step will be executed if the returned value does not match any other entry in the grid. For example, if the user closes a pop-up (rather than clicking a button), the default value will be used.
- **Next Script Step** defines the step to execute if the user clicks the button.

---

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---



## How To Set Up Mathematical Operation Steps

**Mathematical operation** steps allow you to perform arithmetic on fields. You can also use this type of step to add and subtract days from dates. For example, you could calculate a date 7 days in the future and then use this value as the customer's next credit review date. The following additional fields are required for **Mathematical Operation** steps:

**Base Field Type** and **Base Field Name** define the field on which the mathematical operation will be performed. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Page Data Model.** Use this field type when the field resides on any of the tab pages in the [object display area](#). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Temporary Storage.** Use this field type when the field resides in temporary storage. You must initialize the temporary storage field with a Move Data step before performing mathematical operations on the field. Refer to [How To Set Up Move Data Steps](#) for more information.
- **User Interface Field.** Use this field type when the field resides on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

**Math Operation** controls the math function to be applied to the **Base Field**. You can specify +, -, /, and \*. Note, if the base field is a date, you can only use + or -.

**Math Field Type**, **Math Field Name** and **Math Field Value** define the field that contains the value to be added, subtracted, divided, or multiplied. The following points describe each field type:

- **Current To Do Information.** Use this field type when the value resides on the current To Do entry. Refer to [How To Use To Do Fields](#) for instructions on how to define the appropriate **Field Name**.
- **Page Data Model.** Use this field type when the value resides on any of the tab pages in the [object display area](#). Refer to [How To Find The Name Of Page Data Model Fields](#) for instructions on how to find the appropriate **Field Name**.
- **Predefined Value.** Use this field type when the value is a constant. When this field type is used, use **Source Field Value** to define the constant value. Refer to [How To Use Constants In Scripts](#) for more information. Note, if you are performing arithmetic on a date, the field value must contain the number and type of **days/ months/ years**. For example, if you want to add 2 years to a date, the source field value would be **2 years**.
- **Temporary Storage.** Use this field type when the value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field.** Use this field type when the value resides in a field on the current tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

---

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

## How To Set Up Navigate To A Page Steps

**Navigate to a page** steps cause a new page (or tab within the existing page) to be displayed in the object display area. Steps of this type are a precursor to doing anything on the page. The following additional field is required for **Navigate to a page** steps:

**Navigation Option** defines the transaction, tab, access mode (add or change) and any context fields that are passed to the transaction in change mode. For example, if you want a script to navigate to Person - Characteristics for the current person being displayed in the dashboard, you must set up an appropriate navigation option. Refer to [Defining Navigation Options](#) for more information.

---

**NOTE: Navigating to a page in update mode.** Before you can navigate to a page in change mode, the page data model must contain the values to use for the navigation option's context fields. If necessary, you can move values into the page data model using a [Move Data step](#) first. For example, before you can navigate to a page in change mode with an

account ID in context, you may need to move the desired account ID into the ACCT\_ID field in the page data model. The actual field name(s) to use are listed as context fields on the [navigation option](#).

---

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

## How To Set Up Perform Script Steps

**Perform script** steps cause another BPA script to be performed. After the performed script completes, control is returned to the next step in the original script. You might want to think of the scripts referred to on steps of this type as "subroutines". This functionality allows you to encapsulate common logic in reusable BPA scripts that can be called from other BPA scripts. This simplifies maintenance over the long term.

The following additional field is required for **Perform script** steps:

**Subscript** is the name of the script that is performed.

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

## How To Set Up Press A Button Steps

**Press a button** steps cause a button or link text to be 'pressed' in the [object display area](#), the [application toolbar](#) or the [page title area](#). For example, you could use this type of step to add a new row to a person's characteristic (and then you could use a **Move Data** step to populate the newly added row with a given char type and value). The following additional fields are required for **Press a button** steps:

**Button Name** is the name of the button to be pressed. This button must reside on the currently displayed tab page (or in the application toolbar or page actions toolbar). Refer to [How To Find The Name Of A Button](#) for more information.

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

## How To Set Up Prompt User Steps

**Prompt user** steps cause the user to be presented with a menu of options. The options can be presented using either buttons or in the contents of a drop down. You can also use steps of this type to pause a script while the user checks something out (and when the user is ready to continue with the script, they are instructed to click a prompt button). The following additional fields are required for **Prompt User** steps:

**Prompt Type** controls if the prompt shown in the script area is in the form of **Button(s)** or a **Dropdown**. Note, if you use a **Dropdown**, a Continue button appears adjacent to the dropdown in the script area when the step executes. The user clicks the Continue button when they are ready for the script to continue.

The **Prompt Values** grid contains a row for every value that can be selected by a user. Note, if you use a **Prompt Type** of **Button(s)**, a separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button or in the dropdown entry. Refer to [How To Substitute Variables In Text](#) for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons or dropdown entries.
- **Use As Default** can only be turned on for one entry in the grid. If this is turned on for a dropdown entry, this value is defaulted in the grid. If this is turned on for a button, this button becomes the default (and the user should just have to press Enter (or space) rather than click on it).
- **Next Script Step** defines the step to execute if the user clicks the button or selects the dropdown value.

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

## How To Set Up Set Focus To A Field Steps

**Set focus to a field** steps cause the cursor to be placed in a specific field on a page. A **Continue** button always appears in the script area when this type of step executes. The user may click the **Continue** button when they are ready for the script to continue. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Set focus to a field** steps:

**Destination Field Name** defines the field on which focus should be placed. This field must reside on the currently displayed tab page. Refer to [How To Find The Name Of User Interface Fields](#) for instructions on how to find the appropriate **Field Name**.

The **Prompt Values** grid may be used to define additional buttons. A separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button. Refer to [How To Substitute Variables In Text](#) for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons.
- **Next Script Step** defines the step to execute if the user clicks the button.

---

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

## How To Set Up Transfer Control Steps

**Transfer control** steps cause the current BPA script to terminate and the control to pass to another BPA script. You might want to construct a BPA script with steps of this type when the script has several potential logic paths and you want to segregate each logic path into a separate BPA script (for ease of maintenance).

The following additional fields are required for **Transfer control** steps:

**Subscript** is the name of the script to which control is transferred.

---

**NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

---

## Additional Topics

The contents of this section provide additional information about steps.

## How To Find The Name Of User Interface Fields

Follow these steps to find the name of a field that resides on a page:

- Navigate to the page in question.
- Right click in the body of the page (but not while the pointer is in an input field). Note, if the field in question resides in a grid, you must right click while the pointer is in the section that contains the grid (but not while the pointer is in an input field in the grid) - this is because there's a separate HTML document for each grid on a page.
- Select **View Source** from the pop-up menu to display the source HTML.
- Scroll to the Widget Info section (towards the top of the HTML document). It contains a list of all of the objects on a page. For example, the following is an example from the Account - Main page:

```

widget Info:
widget_ID , Element Type - label info - label
ENTITY_NAME, IL - $ENTITY_NAME - Name
ACCT_ID, IT - $ACCT_ID - Account ID
ACCT_CHECK_DIGIT, IL - $ACCT_CHECK_DIGIT - Account Check Digit
IM_ACCT_ID, IM - $SEARCH_FOR_ACC_LBL - Search for Account
COLL_CL_CD, HD - $COLL_CL_CD - Collection Class
SETUP_DT, IT - $SETUP_DT - Set Up Date
CURRENCY_CD, IS - $CURRENCY_CD - Currency Code
CIS_DIVISION, IS - $CIS_DIVISION - CIS Division
PROTECT_DIV_SW, CB - $CI_ACCT$PROTECT_DIV_SW - Protect CIS Division
CUST_CL_CD, IS - $CUST_CL_CD - Customer Class
ACCESS_GRP_CD, IT - $ACCESS_GRP_CD - Access Group
IM_ACCESS_GRP_CD, IM - $SRCH_ACC_GRP_CD - Search for Access Group
ACCESS_DESCR, IL - $DESCR - Description
ACCT_MGMT_GRP_CD, IT - $ACCT_MGMT_GRP_CD - Account Management Group
IM_ACCT_MGMT_GRP_CD, IM - $SEARCH_FOR_TDR_LBL - Search for To Do Role
MGMT_DESCR, IL
ALERT_INFO, IA - $ALERT_INFO - Alert Information
BILL_CYC_CD, IS - $BILL_CYC_CD - Bill Cycle
BILL_AFTER_DT, IT - $BILL_AFTER_DT - Bill After
PROTECT_CYC_SW, CB - $PROTECT_CYC_SW - Protect Bill Cycle
BILL_PRT_INTERCEPT, IT - $BILL_PRT_INTERCEPT - Bill Print Intercept
IM_BILL_PRT_INTERCEPT, IM - $FOR_BILL_PRINT_LBL - Search for User
MAILING_PREM_ID, IT - $MAILING_PREM_ID - Mailing Premise
IM_MAILING_PREM_ID, IM - $SEARCH_FOR_MAI_LBL - Search for Mailing Premise
PREM_INFO, IL
PROTECT_PREM_SW, CB - $PROTECT_PREM_SW - Protect Mailing Premise
dataframe, GD

```

The field names that you'll reference in your scripts are defined on the left side of the HTML (e.g., ENTITY\_NAME, ACCT\_ID, CUST\_CL\_CD, etc.).

The names of fields that reside in scrolls are in a slightly different format. The following is an example of the HTML for the persons scroll that appears on Account - Person. Notice that the fields in the scroll are prefixed with the name of the scroll plus a \$ sign. For example, the person's ID is called ACCT\_PER\$PER\_ID.

```

widget Info:
widget_ID , Element Type - label info - label
ENTITY_NAME, IL - $ENTITY_NAME - Name
PREM_INFO, HD
ACCT_ID, IT - $ACCT_ID - Account ID
ACCT_CHECK_DIGIT, IL - $ACCT_CHECK_DIGIT - Account Check Digit
IM_ACCT_ID, IM - $SEARCH_FOR_ACC_LBL - Search for Account
ACCT_PER$recordCount, SN - $OF_LBL - of
ACCT_PER$PER_ID, IT - $PER_ID - Person ID
IM_ACCT_PER$PER_ID, IM - $FOR_PERSON_LBL - Search for Person
ACCT_PER$ENTITY_NAME, IL - $ENTITY_NAME - Name
ACCT_PER$MAIN_CUST_SW, CB - $MAIN_CUST_SW - Main Customer
ACCT_PER$FIN_RESP_SW, CB - $FIN_RESP_SW - Financially Responsible
ACCT_PER$THRD_PTY_SW, CB - $THRD_PTY_SW - Third Party Guarantor
ACCT_PER$ACCT_REL_TYPE_CD, IS - $CI_ACCT_PER$ACCT_REL_TYPE_CD - Relationship Type
ACCT_PER$WEB_ACCESS_FLG, IS - $WEB_ACCESS_FLG - Web self service Access Flag
ACCT_PER$PFX_SFX_FLG, IS - $PFX_SFX_FLG - Prefix/suffix
ACCT_PER$NAME_PFX_SFX, IT - $NAME_PFX_SFX - Pfx/sfx Name
ACCT_PER$RECEIVE_COPY_SW, CB - $RECEIVE_COPY_SW - Receives Copy of Bill
ACCT_PER$BILL_RTE_TYPE_CD, IS - $BILL_RTE_TYPE_CD - Bill Route Type
ACCT_PER$BILL_RTE_TYPE_INFO, IL
ACCT_PER$BILL_RTG METH_FLG, HD
ACCT_PER$BILL_FORMAT_FLG, IS - $BILL_FORMAT_FLG - Bill Format

```

The names of fields that reside in grids are in a slightly different format. The following is an example of the HTML for the names grid that appears on Person - Main. Notice that the fields in the grid are prefixed with the name of the grid plus a :x \$. For example, the person's name is called PER\_NAME:x\$ENTITY\_NAME. When you reference such a field in your script, you have the following choices:

- Substitute x with the row in the grid (and keep in mind, the first row in a grid is row 0 (zero); this means the second row is row 1).
- If you want to reference the "current row" (e.g., the row in which the cursor will be placed), you can keep the x notation ( x means the "current row").

```

widget Info:
widget_ID , Element Type - label info - label
PER_NAME:x$NAME_TYPE_FLG, IS - $NAME_TYPE_FLG - Name Type
PER_NAME:x$ENTITY_NAME, IT - $CI_PER_NAME$ENTITY_NAME - Person Name

```

## How To Find The Name Of Page Data Model Fields

You find the name of a **Page Data Model** field in the same way described under [How To Find The Name Of User Interface Fields](#). The only restriction is that you cannot refer to hidden / derived fields. However, you can refer to ANY of the object's fields regardless of the tab page on which they appear. For example, if you position the object display area to the Main tab of the Account transaction, you can reference fields that reside on all of the tab pages.

---

**CAUTION:** If you populate a **Page Data Model** field, none of the underlying default logic takes place. For example, if you populate a customer contact's contact type, none of the characteristics associated with the customer contact type are defaulted onto the customer contact. If you want the underlying defaulting to take place, you must populate a **User Interface Field**.

---

## How To Find The Name Of A Button

If you want a **Press a button** step to press a button or click a link in the application toolbar, use one of the following names:

- IM\_GOBACK
- IM\_HISTORY
- IM\_GOFORWARD
- IM\_menuButton
- IM\_USER\_HOME
- IM\_MY\_PREF
- IM\_helpButton
- IM\_aboutButton

If you want a **Press a button** step to press a button in the page actions toolbar, use one of the following names:

- IM\_SAVE
- IM\_REFRESH
- IM\_CLEAR
- IM\_COPY
- IM\_DELETE
- IM\_ScrollBack
- IM\_ScrollForward

The following buttons are also supported:

- IM\_TO\_DO. This brings you to the [To Do Summary](#) page.
- IM\_PrevTo Do. This simulates clicking the **Previous To Do** button in the [Current To Do Zone](#).
- IM\_NextTo Do. This simulates clicking the **Next To Do** button in the [Current To Do Zone](#).
- IM\_CurrentTo Do. This navigates to the [To Do Entry](#) page for the user's current To Do. Refer to [A User's Current To Do](#) for more information.
- IM\_MINIMIZE\_DASHBOARD. Pressing this will collapse the dashboard.
- IM\_MAXIMIZE\_DASHBOARD. Pressing this will expand the dashboard.

Follow these steps to find the name of other buttons that reside in the object display area:

- Navigate to the page in question.

- Right click in the body of the page (but not while the pointer is in an input field). Note, if the field in question resides in a grid, you must right click while the pointer is in the section that contains the grid (but not while the pointer is in an input field in the grid) - this is because there's a separate HTML document for each grid on a page.
- The option to select may differ based on the browser you are using. For example, for some browsers, the option may be **View Source**. For others, the option may be **This Frame** and then **Frame Source**
- Scroll to the Widget Info section (towards the top of the HTML document). It contains a list of all of the objects on a page, including buttons.
- Iconized buttons (e.g., search buttons) are represented as HTML images and their field names are prefixed with **IM**. The following is an example of the HTML on the To Do Entry - Main page (notice the **IM** fields for the iconized buttons).

```
* Widget Info:
*   Widget_ID , Element Type - label info - label
*   TD_ENTRY_INFO, IL - $TD_ENTRY_INFO - To Do Info
*   TD_ENTRY_ID, IT - CI_TD_ENTRY$TD_ENTRY_ID - To Do ID
*   IM_TD_ENTRY_ID, IM - $TD_ENTRY_ID_SRCH - Search for Entry Id
*   TD_TYPE_CD, IL - $TD_TYPE_CD - To Do Type
*   TYPE_DESCR, IL
*   ROLE_ID, IL
*   ROLE_DESCR2, IL - $DESCR - Description
*   FULL_MSG, PL - $GOTO_TD_ACTION_LBL - Work on To Do
*   IM_EXP_MSG_LONG, IM - $DISPLAY_MESSAG_LBL - Display Message Explanation
```

- Transaction-specific actions buttons (e.g., the buttons use to complete or forward a To Do) are represented as switches. The following is an example of the HTML on the To Do Entry - Main page (notice the **SW** fields for the buttons). Note, if you want to **Set focus** to such a field, you would move a **Predefined Value** of **TRUE** to the switch.

```
* COMPLETE_SW, BU - $COMPLETE_SW - Complete
* FORWARD_SW, BU - $FORWARD_SW - Forward
* SEND_BACK_SW, BU - $SEND_BACK_SW - Send Back
```

## How To Substitute Variables In Text

You can substitute field values into a step's text string. You do this by prefixing the field name whose value should be substituted in the string with a **%**. For example, the message, "On **%COMPLETION\_DTTM** this bill was completed, it's ending balance was **%ENDING\_BALANCE**" contains two substitution variables (the bill's completion date / time and the bill's ending balance).

To substitute the value of an element from a data area you need to reference its XPath location as follows: **%=XPath=%**. If you want to substitute the whole XML node, not just the value, you need to reference it as follows **%+XPath+%**.

Only fields linked to the [current To Do](#) and fields that reside in [temporary storage](#) and [global variables](#) can be substituted into a text string.

---

**NOTE:** You can substitute fields that reside in the User Interface or Page Data Model by first moving them into temporary storage (using a **Move data** step).

---

You can also substitute field values into the verbiage displayed in [prompts](#) using the same technique.

## How To Use HTML Tags And Spans In Text Strings and Prompts

You can use HTML tags in a step's text string. For example, the word "Continue" will be italicized in the following text string "Press<i>Continue</i> after you've selected the customer" (the **<i>** and **</i>** are the HTML tags used to indicate that the surrounded text should be italicized).

The following are other useful HTML tags:

- **<br>** causes a line break in a text string. If you use **<br><br>** a blank line will appear.
- **<font color=red>** text **</font>** causes the surrounded text to be colored as specified (in this case, red). You can also use hex codes rather than the color name.



Please refer to an HTML reference manual or website for more examples.

You can also use "spans" to customize the look of the contents of a text string. For example, your text string could be "Press `<span style="font-family:Courier; font-size:large; font-weight:bold;">Continue</span>` after you've selected the customer". This would make the word "Continue" appear as large, bold, Courier text. Please refer to a Cascading Style Sheets (CSS) reference manual or website for more examples.

## How To Use Constants In Scripts

Some steps can reference fields called **Predefined Values**. For example, if you want to compare an input value to the letter "Y", the letter **Y** would be defined as a Predefined Value's field value.

Special constants are used for fields defined as switches. When you move **TRUE** to a switch, it turns it on. When you move **FALSE** to a switch, it turns it off.

You can use a [global variable](#) as a Predefined Value. For example, if you wanted to move the current date to a field, you'd indicate you wanted to move a Predefined Value named **%CURRENT\_DATE**.

## How To Use Global Variables

As described above, some steps can reference fields called **Predefined Values**. In addition to referencing an ad hoc constant value (e.g., the letter **Y**), you can also reference a global variable in such a field value. A global variable is used when you want to reference system data. The following global variables exist:

- **%PARM-`<name>`** is the value of a parameter of that name passed in to the application when launched via the standard system URL. Refer to [Launching A Script When Starting the System](#) for more information on these parameters.
- **%PARM-NOT-SET** is to be used to compare against **%PARM-`<name>`** parameters to check if the parameter has been set or not when the application was launched. A parameter that has not been set would test as equal to this global variable. It is recommended to test parameters against this global variable before using them for the first time.
- **%CONTEXT-PERSONID** is a constant that contains the ID of the current person
- **%CONTEXT-ACCOUNTID** is a constant that contains the ID of the current account
- **%CONTEXT-PREMISEID** is a constant that contains the ID of the current premise
- **%BLANK** is a constant that contains a blank value, i.e. no value
- **%SPACE** is a constant that contains a single space value
- **%CURRENT-DATE** is the current date (as known by the browser, not the server)
- **%SAVE-REQUIRED** is a flag that contains an indication of whether the data on a page has been changed (and this requires saving). You may want to interrogate this flag to force a user to save their work before executing subsequent steps. This flag will have a value of **TRUE** or **FALSE**.
- **%NEWLINE** is a constant that contains a new line character (carriage return). Upon substitution, a line break is inserted in the resultant text.

---

**NOTE:** The constant **%NEWLINE** does not have the desired effect when the resultant text is HTML. For example, a step's text and prompt strings. This is because HTML ignores special characters such as new lines. Refer to [How To Use HTML Tags And Spans In Text](#) to learn how to cause a line break in an HTML text.

---

In addition, if an **Invoke Function** step returns an error, the following global variables contain information about the error:

- **%ERRMSG-CATEGORY** and **%ERRMSG-NUMBER** contain the unique identifier of the error message number.
- **%ERRMSG-TEXT** contains the brief description of the error.
- **%ERRMSG-LONG** contains the complete description of the error.

## How To Name Temporary Storage Fields

**Input Data** and **Move Data** steps can create fields in temporary storage. You specify the name of the temporary storage field in the step's **Field Name**. The name of the field must NOT begin with % and must not be named the same as the [global variables](#). Besides this restriction, you can use any **Field Name** that's acceptable to JavaScript (i.e., you can name a field in temporary storage almost anything). Keep in mind that field names are case-sensitive.

## How To Work With Dates

Before we discuss how to work with dates in your scripts, we need to point out that there are two types of date fields: date-only and date-time. Date-only fields only contain a date. Date-time fields contain both a date and a time. The following topics describe how to work with dates on the various step types.

---

**NOTE:** If you're working with a field that resides on the database (as opposed to a temporary storage field), the database field name will tell you what type of date it is: date-only fields are suffixed with **DT**, and date-time fields are suffixed with **DTTM**.

---

## Move Data

If you intend to use a **Move data** step to populate a *date-time* field, please be aware of the following:

- If the destination field resides in the *page data model*, the source field value must be in the format YYYY-MM-DD-HH.MM.SS or YYYY-MM-DD. If the field is in the format YYYY-MM-DD, the time of 12:00 am will be defaulted.
- If the destination field resides in the *user interface*, you must use two steps if you want to populate both date and time. To explain this, we'll assume the field you want to populate is called EXPIRE\_DTTM:
  - First, you populate the date portion of the field. To do this, you'd move a date (this value can be in any valid date format that a user is allowed to enter) to a field called EXPIRE\_DTTM\_FWDDTM\_P1. In other words, you suffix **\_FWDDTM\_P1** to the field name.
  - If you want to populate the time, you'd move the time (again, the field value can be in any format that a user could use to enter a time) to a field called EXPIRE\_DTTM\_FWDDTM\_P2. In other words, you suffix **\_FWDDTM\_P2** to the field name.

If you intend to use a **Move data** step to populate a *date-only* field, please be aware of the following:

- If the destination field resides in the *page data model*, the source field value must be in the format YYYY-MM-DD.
- If the destination field resides in the *user interface*, the source field can be in any valid date format that a user is allowed to enter.

---

**NOTE:** **%CURRENT-DATE**. Keep in mind that the [global variable](#) **%CURRENT-DATE** contains the current date and you can move this to either a page data model, user interface, or temporary storage field. If you move **%CURRENT-DATE** to a temporary storage fields, it is held in the format YYYY-MM-DD.

---

## Mathematical Operation

If you intend to use a **Mathematical operation** step to calculate a date, you can reference both date-only and date-time fields. This is because mathematical operations are only performed against the date portion of date-time fields.

Mathematical operations are limited to adding or subtracting days, months and years to / from a date.



---

**NOTE:** A useful technique to perform date arithmetic using the current date is to move the [global variable](#) %CURRENT-DATE to a temporary storage field and then perform the math on this field.

---

## Input Data

If you intend to use an **Input data** step on a *date-time* field, please be aware of the following:

- If the field resides in the *page data model*, the user must enter a value in the format YYYY-MM-DD-HH.MM.SS (and therefore we do not recommend doing this).
- If the field resides in the *user interface*, you must use two steps if you want to populate both date and time. To explain this, we'll assume the field you want to populate is called EXPIRE\_DTTM:
  - First, you populate the date portion of the field. To do this, you'd input the date (this value can be in any valid date format that a user is allowed to enter) in a field called EXPIRE\_DTTM\_FWDDTM\_P1. In other words, you suffix **\_FWDDTM\_P1** to the field name.
  - If you want to populate the time, you'd input the time (again, the field value can be in any format that a user could use to enter a time) in a field called EXPIRE\_DTTM\_FWDTTM\_P2. In other words, you suffix **\_FWDDTM\_P2** to the field name.

If you intend to use an **Input data** step to populate a *date-only* field, please be aware of the following:

- If the field resides in the *page data model*, the user must enter a value in the format YYYY-MM-DD (and therefore we do not recommend doing this).
- If the field resides in the *user interface*, the user can enter any valid date format.

## How To Use To Do Fields

As described under [Executing A Script When A To Do Entry Is Selected](#), you can set up the system to automatically launch a script when a user selects a To Do entry. These types of scripts invariably need to access data that resides on the selected To Do entry. The following points describe the type of information that resides on To Do entries:

- **Sort keys.** These values define the various ways a To Do list's entries may be sorted. For example, when you look at the bill segment error To Do List, you have the option of sorting the entries in error number order, account name order, or in customer class order. There is a sort key value for each of these options.
- **Message parameters.** These values are used when the system finds %n notation within the message text. The %n notation causes field values to be substituted into a message before it's displayed. For example, the message text **The %1 non-cash deposit for %2 expires on %3** will have the values of three fields merged into it before it is displayed to the user (%1 is the type of non-cash deposit, %2 is the name of the customer, and %3 is the expiration date of the non-cash deposit). Each of these three values is stored as a separate message parameter on the To Do entry.
- **Drill keys.** These values are the keys passed to the page if a user drilled down on the entry (and the system wasn't set up to launch a script). For example, a To Do entry that has been set up to display an account on the account maintenance page has a drill key of the respective account ID.
- **To Do ID.** Every To Do entry has a unique identifier referred to as its To Do ID.

You can access this information in the following types of steps:

- **Move Data** steps can move any of the above to any data area. For example, you might want to move a To Do entry's drill key to the page data model so it can be used to navigate to a specific page.
- **Conditional Branch** steps can perform conditional logic based on any of the above. For example, you can perform conditional logic based on a To Do entry's message number (note, message numbers are frequently held in sort keys).
- **Mathematical Operation** steps can use the above in mathematical operations.

A To Do entry's sort key values are accessed by using a **Field Type** of **Current To Do Information** and a **Field Name** of **SORTKEY[index]**. Note, you can find an entry's potential sort keys by displaying the entry's To Do type and navigating to the [Sort Keys](#) tab. If you want to reference the first sort key, use an index value of **1**. If you want to use the second sort key, use an index value of **2** (and so on).

A To Do entry's drill key values are accessed by using a **Field Type** of **Current To Do Information** and a **Field Name** of **DRILLKEY[index]**. Note, you can find an entry's potential drill keys by displaying the entry's To Do type and navigating to the [Drill Keys](#) tab. If you want to use the first drill key, use an index value of **1**. If you want to use the second drill key, use an index value of **2** (and so on).

A To Do entry's message parameters are accessed by using a **Field Type** of **Current To Do Information** and a **Field Value** of **MSGPARAM[index]**. Note, because a To Do type can have an unlimited number of messages and each message can have different parameters, finding an entry's message parameters requires some digging. The easiest way to determine these values is to display the To Do entry on [To Do maintenance](#). On this page, you will find the entry's message category/number adjacent to the description. Once you know these values, display the message category/number on [Message Maintenance](#). You'll find the message typically contains one or more %n notations (one for each message parameter). For example, the message text **The %1 non-cash deposit for %2 expires on %3** has three message parameters. You then need to deduce what each of the message parameters are. You do this by comparing the message on the To Do entry with the base message (it should be fairly intuitive as to what each message parameter is). If we continue using our example, **%1** is the non-cash deposit type, **%2** is the account name, and **%3** is the expiration date. You can access these in your scripts by using appropriate index value in **MSGPARAM[index]**.

A To Do entry's unique ID is accessed by using a **Field Type** of **Current To Do Information** and a **Field Value** of **TD\_ENTRY\_ID**.

In addition, any of the above fields can be [substituted into a text string or prompt](#). Simply prefix the To Do field name with a % as you would fields in temporary storage. For example, assume you want your script to display the following text in the script area: "ABC Supply does not have a bill cycle" (where ABC Supply is the account's name). If the first sort key linked to the To Do entry contains the account's name, you'd enter a text string of **%SORTKEY[1] does not have a bill cycle**.

## How To Reference Fields In Data Areas

Various step types involve referencing field elements residing in the [script's data areas](#). To reference an element in a data area you need to provide its absolute XPath notation starting from the data area name. For example, use "CaseLogAdd/caseID" to reference a top-level "caseID" element in a script data area called "CaseLogAdd".

You don't have to type in long XPath notations. Use the **View Script Schema** hyperlink provided on the [Script - Step](#) tab page to launch the script's data areas schema.

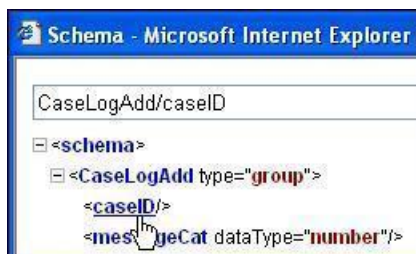


Figure 4: Schema Viewer

Doing this opens the [schema viewer](#) window where you can:

- Click on the field element you want to reference in your script step. The system automatically populates the text box on the top with the element's absolute XPath notation.
- Copy the element's XPath notation from the text box to your script.

You can also use the **View Data Area**, **View Service Script Data Area**, or **View Plug-In Script Data Area** links on [Script - Data Area](#) to the same effect. These open up the schema viewer for a specific data area respectively.

## Script - Data Area

Use this page to define the data areas used to pass information to and from the server or any other data area describing your temporary storage. Open this page using **Admin > System > Script** and then navigate to the **Data Area** tab.

### Description of Page

The grid contains the script's data areas declaration. For steps that invoke an object that is associated with a schema, you must declare the associated schema as a data area for your script. In addition, if you have defined one or more data areas to describe the script's temporary storage, you need to declare them too. The following bullets provide a brief description of each field on a script data area:

- **Schema Type** defines the type of schema describing the data area's element structure.
- The data area's schema is the one associated with the referenced **Object**. Only objects of the specified Schema Type may be selected.
- **Data Area Name** uniquely identifies the data area for referencing purposes. By default, the system assigns a data area with the associated object name.
- Click on the **View Data Area** link to view the data area's schema in the [schema viewer](#) window.

The **View Service Script Data Area** link appears for service scripts only. Use this link to view the script's parameters data area schema in the [schema viewer](#) window.

The **View Plug-In Script Data Area** link appears for plug-in scripts only. Use this link to view the script's parameters data area schema in the [schema viewer](#) window.

---

**FASTPATH:** Refer to [A Script May Declare Data Areas](#) for more information on data areas.

---

## Script - Schema

Use this page to define the data elements passed to and from a service script. Open this page using **Admin > System > Script** and then navigate to the **Schema** tab.

---

**NOTE: Conditional tab page.** This tab page only appears for [service scripts](#).

---

### Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays the script name and description.

The [Schema Designer](#) zone allows you to edit the service script's parameters schema. The purpose of the schema is to describe the input and output parameters used when invoking the script.

---

**NOTE: Script Definition Tips.** A context sensitive "Script Tips" zone is associated with this page. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema as well as other useful information assisting with setting up scripts.

---

The **Schema Usage Tree** zone summarizes all cross-references to this schema. For each type of referencing entity, the [tree](#) displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

## Script - Eligibility

Use this page to define a script's eligibility rules. Open this page using **Admin > System > Script** and then navigate to the **Eligibility** tab.

---

**NOTE: Conditional tab page.** This tab page only appears for [BPA scripts](#).

---

### Description of Page

Use the **Eligibility Option** to indicate whether the script is **Always Eligible**, **Never Eligible** or to **Apply Eligibility Criteria**. The remaining fields on the page are only visible if the option is **Apply Eligibility Criteria**.

---

**CAUTION:** The following information is not intuitive; we strongly recommend that you follow the guidelines under [The Big Picture Of Script Eligibility](#) before attempting to define this information.

---

The **Eligibility Criteria Group** scroll contains one entry for each group of eligibility criteria. The following fields may be defined for each group:

- Use **Sort Sequence** to control the relative order in which the group is executed when the system determines if the script should appear in the [script search](#).
- Use **Description** and **Long Description** to describe the criteria group.
- Use **If Group is True** to define what should happen if the eligibility criteria (defined in the following grid) return a value of **True**.
  - Choose **Eligible** if this script should appear.
  - Choose **Ineligible** if this script should not appear.
  - Choose **Check Next Group** if the next criteria group should be checked.
- Use **If Group is False** to define what should happen if the eligibility criteria (defined in the following grid) return a value of **False**.
  - Choose **Eligible** if this script should appear.
  - Choose **Ineligible** if this script should not appear.
  - Choose **Check Next Group** if the next criteria group should be checked.

The grid that follows contains the script's eligibility criteria. Think of each row as an "if statement" that can result in the related eligibility group being true or false. For example, you might have a row that indicates the script is eligible if the current account in context belongs to the residential customer class. The following bullets provide a brief description of each field on an eligibility criterion. Please refer to [Defining Logical Criteria](#) for several examples of how this information can be used.

- Use **Sort Sequence** to control the order in which the criteria are checked.
- Use **Criteria Field** to define the field to compare:
  - Choose **Algorithm** if you want to compare anything other than a characteristic. Push the adjacent search button to select the algorithm that is responsible for retrieving the comparison value. Click [here](#) to see the algorithm types available for this plug-in spot.
  - Some products may also include an option to choose **Characteristic**. Choosing this option displays adjacent fields to define the object on which the characteristic resides and the characteristic type. The objects whose characteristic values may be available to choose from depend on your product.
- Use **Criteria Comparison** to define the method of comparison:

- Choose **Algorithm** if you want an algorithm to perform the comparison and return a value of True, False or Insufficient Data. Push the adjacent search button to select the algorithm that is responsible for performing the comparison. Click [here](#) to see the algorithm types available for this plug-in spot.
- Choose any other option if you want to compare the **Criteria Field** using a logical operator. The following options are available:
  - Use **>**, **<**, **=**, **>=**, **<=**, **<>** (not equal) to compare the **Criteria Field** using standard logical operators. Enter the comparison value in the adjacent field.
  - Use **IN** to compare the **Criteria Field** to a list of values. Each value is separated by a comma. For example, if a field value must equal **1, 3 or 9**, you would enter a comparison value of **1,3,9**.
  - Use **BETWEEN** to compare the **Criteria Field** to a range of values. For example, if a field value must be between **1 and 9**, you would enter a comparison value of **1,9**. Note, the comparison is inclusive of the low and high values.
- The next three fields control whether the related logical criteria cause the eligibility group to be considered true or false:
  - Use **If True** to control what happens if the related logical criterion returns a value of True. You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.
  - Use **If False** to control what happens if the related logical criterion returns a value of False. You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.
  - Use **If Insufficient Data** to control what happens if the related logical criterion returns a value of "Insufficient Data". You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.

## Merging Scripts

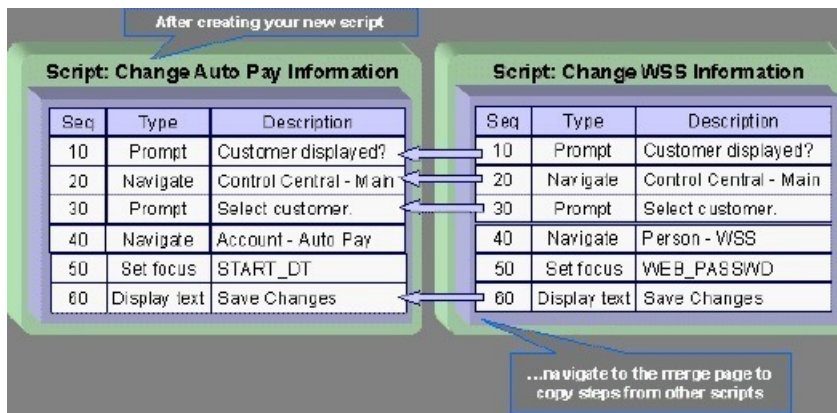
Use the Script Merge page to modify an existing script by copying steps from other scripts. The following points summarize the many diverse functions available on the Script Merge transaction:

- You can use this transaction to renumber steps (assign them new sequence numbers).
- You can use this transaction to move a step to a different position within a script. When a step is moved, all references to the step are changed to reflect the new sequence number.
- You can use this transaction to delete a step.
- You can use this transaction to copy steps from other scripts. For example,
  - You may want to create a script that is similar to an existing script. Rather than copying all the information from the existing script and then removing the inapplicable steps, this page may be used to selectively copy steps from the existing script to the new script.
  - You may have scripts that are very similar, but still unique. You can use this transaction to build large scripts from smaller scripts. In this scenario, you may choose to create special 'mini' scripts, one for each of the various options that may make a script unique. Then, you could use the script merge page to select and merge the mini scripts that are applicable for a main script.

---

**NOTE:** The target script must exist prior to using this page. If you are creating a new script, you must first create the [Script](#) and then navigate to the merge page to copy step information.

---



**NOTE: Duplicate versus Merge.** The **Script** page itself has **duplication** capability. You would duplicate a script if you want to a) create a new script AND b) populate it with *all* the steps from an existing script.

## Script Merge

Open **Admin > System > Script Merge** to open this page.

### Description of Page

For **Original Script**, select the target script for merging steps.

For **Merge From Script**, select the template script from which to copy the steps.

**NOTE:** You may only copy steps from one Merge From script at a time. If you want to copy steps from more than one script, select the first Merge From script, copy the desired steps, save the original script, and then select the next Merge From script.

The left portion of the page displays any existing steps for the **Original Script**. The right portion of the page displays the existing steps for the **Merge From Script**.

You can use the **Copy All** button to copy all the steps from the **Merge From** script to the **Original** script. If you use **Copy All**, the steps are added to the end of the original script.

Each time you save the changes, the system renumbers the steps in the original script using the **Start From Sequence Number** and **Increment By**.

**Merge Type** indicates **Original** for steps that have already been saved in the original script or **Merge** for steps that have been merged, but not yet saved. The **Sequence**, **Step Type** and **Description** for each step are displayed.

The topics that follow describe how to perform common maintenance tasks:

### Resequencing Steps

If you need to resequence the steps:


- Use the up and down arrows in the Original Script grid to reorder the steps.
- Make any desired changes to the **Start From Sequence Number** or **Increment By**.
- Click Save.


The steps are given new sequence numbers according to their order in the grid.




**FASTPATH:** Refer to **Editable Grid** in the system wide standards documentation for more information about adding records to a collection by selecting from a list and repositioning rows within a grid.












## Removing a Step from Script

If you want to remove a record linked to the Original script, click the delete button, , to the left of the record.

For example, to remove the **Reset existing bundle XML** step, click the  icon.

		Merge Type	Sequence	Step Type	Description
	 	Original	10	Edit data	Edit data - Check that the BO is an Export Bundle
	 	Original	20	Edit data	Edit data - Read the Bundle
	 	Original	30	Edit data	Edit data - Reset existing bundle XML
	 	Original	40	Edit data	Edit data - Create new bundle XML

After removal, the grid displays:

		Merge Type	Sequence	Step Type	Description
	 	Original	10	Edit data	Edit data - Check that the BO is an Export Bundle
	 	Original	20	Edit data	Edit data - Read the Bundle
	 	Original	40	Edit data	Edit data - Create new bundle XML

---

**NOTE:** You cannot delete a step that is referenced by other steps unless you also delete the referencing steps, such as **Go to step** or **Prompt** type steps. The system informs you of any missing referenced steps when you attempt to save the original script.

---

## Adding a Step to a Script

You can move any of the steps from the Merge From script to the Original Script by clicking the left arrow adjacent to the desired step. Once a record is moved it disappears from the Merge From information and appears in the Original information with the word **Merge** in the Merge Type column.

For example, to copy the **Navigate to a page** step, click the left arrow.

Merge Type	Sequence	Step Type	Description
	10	Height	Height - 0%
	20	Move data	Move data - Copy 'APP_SVC_ID' to 'APP_SVC_ID'
	30	Move data	Move data - Copy 'USR_GRP_ID' to 'USR_GRP_ID'
	40	Navigate to a page	Navigate to a page - userGroupAppService
	50	Conditional branch	Conditional branch - Compare ACTION with UPDATE
	60	Press a button	Press a button - IM_scrollSec_add
	70	Move data	Move data - Copy 'APP_SVC_ID' to 'UGP\$APP_SVC_ID'
	80	Label	Label - End of Script









The step is moved to the left portion of the page.

Merge Type	Sequence	Step Type	Description
Merge	40	Navigate to a page	Navigate to a page - userGroupAppService
	10	Height	Height - 0%
	20	Move data	Move data - Copy 'APP_SVC_ID' to 'APP_SVC_ID'
	30	Move data	Move data - Copy 'USR_GRP_ID' to 'USR_GRP_ID'
	50	Conditional branch	Conditional branch - Compare ACTION with UPDATE
	60	Press a button	Press a button - IM_scrollSec_add
	70	Move data	Move data - Copy 'APP_SVC_ID' to 'UGP\$APP_SVC_ID'
	80	Label	Label - End of Script

**NOTE:** If you add a step, such as **Go to step** or **Prompt** type steps, that references other steps, you must also add the referenced steps. The step references are updated to use the new sequence numbers when you save the original script. The system informs you of any referenced steps that haven't been added when you attempt to save the original script.



## Removing an Uncommitted Step from a Script

	Merge Type	Sequence	Step Type	Description
	Merge	40	Navigate to a page	Navigate to userGroup
		10	Height	Height - 0%
		20	Move data	Move data - Copy 'APP_SVC_ID' to 'APP_SVC_ID'
		30	Move data	Move data - Copy 'USR_GRP_ID' to 'USR_GRP_ID'
		50	Conditional branch	Conditional branch - Compare ACTION with UPDATE
		60	Press a button	Press a button - IM_scrollSec_add
		70	Move data	Move data - Copy 'APP_SVC_ID' to 'UGP\$APP_SVC_ID'
		80	Label	Label - End of Script

## Maintaining Functions

**NOTE:** Functions were implemented prior to the introduction of business services (BS), service scripts (SS) and business objects (BO). The functionality is still supported, but the recommendation for implementations going forward is to use one of the above configuration tool objects in a script rather than defining a function. The documentation has not been updated throughout this section to highlight where BS, SS or BO could be used to perform the equivalent logic.

**Invoke function** steps may be used to retrieve or update data independent of the page currently being displayed. For example, if you design a script that takes different paths based on the customer's customer class, you could invoke a function to retrieve the customer's customer class. Doing this is much more efficient than the alternative of transferring to the account page and retrieving the customer class from the Main page.

An **Invoke function** step retrieves or updates the relevant data by executing a service (on the server). These types of steps do not refer to the service directly. Rather, they reference a "function" and the function, in turn, references the service.

**NOTE: Functions are abstractions of services.** A function is nothing more than meta-data defining the name of a service and how to send data to it and retrieve data from it. Functions allow you to define a scriptwriter's interface to services. They also allow you to simplify a scriptwriter's set up burden as functions can handle the movement of data into and out of the service's XML document.

The topics in this section describe how to set up a function.

**NOTE: You can retrieve data from all base-package objects.** If you know the name of the base-package "page" service used to inquire upon an object, you can retrieve the value of any of its fields for use in your scripts. To do this, set up a function that sends the unique identifier of the object to the service and retrieves the desired fields from it.

## Function - Main

Use this page to define basic information about a function. Open this page using **Admin > System > Function**.

### Description of Page

Enter a unique **Function** code and **Description** for the function.

Use the **Long Description** to describe, in detail, what the function does.

Define the **Internal Service** that the function invokes.

---

**NOTE:** In this release, only page services can be invoked.

---

Click the **View XML** hyperlink to view the XML document used to pass data to and from the service. Doing this causes the XML document to be displayed in the Application Viewer.

---

**NOTE: XML document may not be viewable.** If you create a new page service and do not regenerate the application viewer, you will not be able to view its XML document.

---

The tree summarizes the following:

- The fields sent to the service. You can use the hyperlink to transfer to the **Send Fields** tab with the corresponding field displayed.
- The fields received from the service. You can use the hyperlink to transfer to the **Receive Fields** tab with the corresponding field displayed.
- Scripts that reference the function. You can use the hyperlink to transfer to the script page.

## Function - Send Fields

Use this page to add or update the fields sent to the service. Open this page using **Admin > Function > System** and then navigate to the **Send Fields** tab.

---

**NOTE: Displaying a specific field.** Rather than scrolling through each field, you can navigate to a field by clicking on the respective node in the tree on the Main tab. Also note, you can use the Alt+right arrow and Alt+left arrow accelerator keys to quickly display the next and previous entry in the scroll.

---

**NOTE: You're defining the service's input fields.** On this tab, you define which fields are populated in the XML document that is sent to the service. Essentially, these are the service's input fields.

---

### Description of Page

Use **Sequence** to define the order of the **Send Fields**.

Enter a unique **Function Field Name** and **Description** for each field sent to the application service. Feel free to enter **Comments** to describe how the field is used by the service.

Use **Field Value Source** to define the source of the field value in the XML document sent to the service:

- If the field's value is the same every time the function is invoked, select **Defined On The Function**. Fields of this type typically are used to support "hard-coded" input values (so that the scriptwriter doesn't have to populate the field every time they invoke the function). Enter the "hard-coded" **Field Value** in the adjacent field.
- If the field's value is supplied by the script, select **Supplied By The Invoker**. For example, if the function retrieves an account's customer class, the script would need to supply the value of the account ID (because a different account ID is passed each time the function is invoked). Turn on **Required** if the invoker must supply the field's value (it's possible to have optional input fields).

Regardless of the Field Value Source, use **XML Population Logic** to define the XPath expression used to populate the field's value in the XML document sent to the service.

---

**NOTE: Usability suggestion.** You populate a field's value in an XML document by specifying the appropriate XPath expression for each field. Rather than referring to an XPath manual, the system can create the XPath expression for you. To do this, click the adjacent **View XML** hyperlink. This will display the XML document used to communicate with the **Service** defined on the Main page. After the XML document is displayed, click the **XPath** hyperlink adjacent to

the desired field to see how the XPath expression looks. You can then cut / paste this XPath expression into the **XML Population Logic Field**.

---

## Function - Receive Fields

Use this page to add or update the fields received from the service. Open this page using **Admin > System > Function** and then navigate to the **Receive Fields** tab.

---

**NOTE: Displaying a specific field.** Rather than scrolling through each field, you can navigate to a field by clicking on the respective node in the tree on the Main tab. Also note, you can use the Alt+right arrow and Alt+left arrow accelerator keys to quickly display the next and previous entry in the scroll.

---

**NOTE: You're defining the application service's output fields.** On this tab, you define which fields are populated in the XML document that is received from the service. Essentially, these are the service's output fields.

---

### Description of Page

Use **Sequence** to define the order of the **Receive Fields**.

Enter a unique **Function Field Name** and **Description** for each field received from the service. Feel free to enter **Comments** to describe the potential values returned from the service.

Turn on **Required** if the invoker must use the field.

Regardless of the Field Value Source, use **XML Population Logic** to define the XPath expression used to retrieve the field's value from the XML document received from the service.

---

**NOTE: Usability suggestion.** You retrieve a field's value in an XML document by specifying the appropriate XPath expression for the field. Rather than referring to an XPath manual, the system can create the XPath expression for you. To do this, click the adjacent **View XML** hyperlink. This will display the XML document used to communicate with the **Service** defined on the Main page. After the XML document is displayed, click the **XPath** hyperlink adjacent to the desired field to see how the XPath expression looks. You can then copy / paste this XPath expression into the **XML Population Logic Field**.

---

**NOTE: Fields in multiple lists.** Note that the XPath expression generated in the application viewer refers to lists using a generic "list" reference. If a field within the list is unique across the service, the generic list reference is sufficient for the XML population logic. However, if the field you are trying to reference is in multiple lists, the XPath must include the list name. Adjust the Application Viewer's generated XPath by adding the list name, which can be found in the overview panel in the Service XML viewer. For example, instead of `/pageBody/list/listBody/field[@name='FIELD_NAME']`, the XPath Population Logic must read `/pageBody/list[@name='LIST_NAME']/listBody/field[@name='FIELD_NAME']`.

---

## Attachments

---

Some implementations may require that attachments be available from the application. These attachments can be stored in the Attachment table and then linked to other objects if applicable.

## Attachment Overview

The following topics provide additional information regarding attachment functionality.

## Attachment Types

The system supports several different attachment content types, for example:

- PDF Document
- Excel Spreadsheet
- Jpeg Image
- Text Document

The attachment data itself may be text or binary. When storing the data in the application however, it is stored as text information only. As a result, the upload of an attachment that is a binary type requires a conversion prior to storing the data. When viewing the attachment, the data is converted again for display.

Each type of attachment is defined using an attachment business object. The business object includes configuration defining the supported file extensions, whether the data is binary or not and the content type that represents the type of data for the attachment.

---

**NOTE:** To view the attachment business objects provided with the base product, navigate using **Admin > Business Object > Search** and search for business objects related to the Maintenance Object for Attachments (**F1-ATCHMT**).

---

## Owned Attachments

Attachments can be either ‘owned’ or ‘common’. An owned attachment is one that is related to a specific record. For example, the specific test results for a given device can be uploaded and linked to that device or to its test records. These types of attachments are typically uploaded and maintained via the object that owns it.

## Common Attachments

Common attachments are ones that are uploaded independent of any transaction in the system. They can be used for general system or company information. Or they can be linked to more than one transaction. For example, instructions for performing a certain type of task can be uploaded as an attachment and linked to a task type where those instructions are relevant. These types of attachments are uploaded and maintained in the central Attachment portal. Objects that may refer to the attachments may link the attachments via characteristics or some other appropriate mechanism.

## Emailing Attachments

The system supports a business service that may be used by system processing to send an email. The business service **F1-EmailService** supports receiving the IDs of one or more attachments as input parameters.

Refer to [Sending Email](#) for more information.

## Configuring Your System for Attachments

In order to link attachments to objects in the system, there may be some configuration or implementation required to support the link. It is possible that one or more objects in your product already support attachments out of the box. Consult the product documentation for the specific object for confirmation. For objects in the system that do not support attachments out of the box, the following sections provide some guidelines for enabling support for attachments. Contact product support for more information.

## Supporting Common Attachments

The attachments themselves are created / uploaded using the attachment portal. Refer to [Maintaining Attachments](#) for more information.

If your implementation has a use case where one or more common attachments may be linked to an object (and the object does not already support this functionality), the object may need to be extended to capture the attachments.

- If the object includes a characteristic collection, this is a recommended way to capture attachments. A characteristic type should be defined for each type of attachment. The characteristic type should be a foreign key type and should reference the Attachment FK reference. The characteristic entity collection should include the object that the common attachment will be linked to.
- Most characteristic collections are sequence based characteristics and would support multiple entries for the same characteristic types, if multiple attachments are applicable.
- If the object to support the attachments is governed by a business object, the implementation must extend the business object to define one or more appropriate elements used to capture the attachments. If only one attachment of a certain type is allowed, a single flattened characteristic may be used. If multiple attachments of a certain type are allowed, the BO schema may define a “flattened list” exposing the sequence and the characteristic type.
- If the object is maintained on a “fixed page” with a generic characteristic collection, no additional configuration is needed to allow users to link attachments to that object.

## Supporting Owned Attachments

When creating an attachment for a specific record, the attachment itself captures the information about the related record, namely its maintenance object code and its primary key. For these types of attachments, no configuration is needed on the related business object to capture the attachments, as was the case with common attachments.

However, it is recommended to configure the user interface of the related object so that the owned attachments can be viewed and maintained from that page. This typically entails the creation of a special zone that retrieves a list of existing attachment records that reference the current record as its foreign key (owner). The zone would include links or buttons to add, upload or view an attachment.

If your product already has support for viewing and maintaining owned attachments on one of the base portals, that may be used as an example to follow. If your product does not have support for owned attachments, contact customer support for more information.

## Defining a New Attachment Type

As mentioned, the product provides support for several content types. If your implementation needs to support attachments for a content type not currently supported, create a new business object copying the configuration of an existing attachment business object.

Configure the following option types for the BO:

- **Binary** indicates whether the attachment data must be converted from binary format. Binary attachments are stored in the database as text, and are then converted back to the original format when retrieved.
- **Content Type** represents the browser's mime type of the attachment.
- **Supported File Extension** specifies the valid file extensions for the content type.

Once the business object is defined, it is ready for use.

## Maintaining Attachments

This section describes the functionality supported for viewing and maintaining attachments.

Navigate using **Admin > General > Attachment**. You are brought to a query portal with options for searching for common attachments.

Once an attachment has been selected, you are brought to the maintenance portal to view and maintain the selected record.

---

**NOTE:** The base search options for the attachments query only support searching for common attachments. Owned attachments may also be viewed on the attachment maintenance portal, but a user may only drill into the attachment maintenance from the maintenance portal of the “owning” entity.

---

The Attachment zone provides basic information about an attachment, including the ability to upload the file and to view an uploaded file.

## Adding Attachments

Common attachments may be added from the attachments portal (or via the standard menu path). In addition, your product may support attachments associated with specific records (“entity owned attachments”) which may also provide the capability to add attachments.

In both cases, when adding an attachment, you are prompted for the file to upload. Once the file is chose, the system determines the appropriate business object to associate with the attachment based on the file extension. Typically one and only one business object is found at which point you are prompted to provide the Attachment Name. (Your specific product may also require additional information at this time). Fill in the details and save.

Please note the following:

- If no business object is found for the uploaded file’s file type, an error is issued. This type of file is not currently supported as an attachment.
- If multiple business objects are found, the user must choose the appropriate one. This should be rare.

## Application Viewer

---

The Application Viewer allows you to explore meta-data driven relationships and other deliverable files online.

**NOTE:** Running Stand-Alone. You can also launch the Application Viewer as a stand-alone application (i.e., you do not need to start it from within the system). Refer to [Application Viewer Stand-Alone Operation](#) for more information about running the Application Viewer as a stand-alone application.

---

To open the application viewer from within your application, navigate to **Admin > Application Viewer**. The application viewer may also be launched from other locations for example when viewing a section of the online help files that contain hypertext for a table name, clicking on that hypertext brings you to the definition of that table in the data dictionary.

## Application Viewer Toolbar

The Toolbar provides the main controls for using the Application Viewer. Each button is described below.

### Data Dictionary Button



The **Data Dictionary** button switches to the Data Dictionary application.

## Physical and Logical Buttons



The **Physical** button changes the display in the List Panel from a logical name view to a physical name view. Note that the Tables are subsequently sorted by the physical name and therefore may not be in the same order as the logical name view. Once clicked, this button toggles to the Logical button.

The **Logical** button changes the display in the List Panel from a physical name view to a logical name view. Note that the Tables are subsequently sorted by the logical name and therefore may not be in the same order as the physical name view. Once clicked, this button toggles to the Physical button.

These buttons are only available in the Data Dictionary.

## Collapse Button



The **Collapse** button closes any expanded components on the list panel so that the child items are no longer displayed. This button is only available in the Data Dictionary viewer.

## Attributes and Schema Button



The **Attributes** button changes the display in the Detail Panel from a related tables view to an attribute view. Once clicked, this button toggles to the Schema button.



The **Schema** button changes the display in the Detail Panel from an attribute view to a related tables view. Once clicked, this button toggles to the Attributes button. Note that only tables have this view available. Columns are always displayed in an attribute view.

These buttons are only available in the Data Dictionary.

## Maintenance Object Button



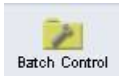
The **Maintenance Object** button switches to the Maintenance Object viewer application.

## Algorithm Button



The **Algorithm** button switches to the Algorithm viewer application.

## Batch Control Button



The **Batch Control** button switches to the Batch Control viewer application.

## To Do Type Button



The **To Do Type** button switches to the To Do Type viewer application.

## Description and Code Buttons



The **Description** button changes the display in the List Panel to Description (Code) from Code (Description). Note that the list is subsequently sorted by the description. Once clicked, this button toggles to the Code button.

The **Code** button changes the display in the List Panel to Code (Description) from Description (Code). Note that the list is subsequently sorted by the Code. Once clicked, this button toggles to the Description button.

These buttons are only available in the Batch Control and To Do Type viewers.

## Service XML Button



The **Service XML** button switches to the Service XML viewer. This button is not available when you are already in the Service XML viewer.

You are prompted to enter the name of the service XML file you want to view. The name of the service XML file should be entered without the extension.

A dialog box with a light blue border. It has a text input field labeled "Service XML Name" with a white background. Below the input field are two yellow buttons: "Load" on the left and "Cancel" on the right.



## Select Service Button



The **Select Service** button loads another service XML file that you specify. This button is only available in the Service XML viewer.

You are prompted to enter the name of the service XML file you want to view. The name of the service XML file should be entered without the extension.

## Java Docs Button



The **Java Docs** button switches to the Java Docs viewer.

## Classic Button



This button is only available in the Java Docs viewer.

The **Classic** button launches the classic Javadocs viewer on a separate window. If you are more comfortable with that look you can use this viewer instead.

## Preferences Button



The **Preferences** button allows you to set optional switches used by the Application Viewer. Refer to [Application Viewer Preferences](#) for more information.

## Help Button



The **Help** button opens the Application Viewer help system. You used this button to access this information.

## About Button

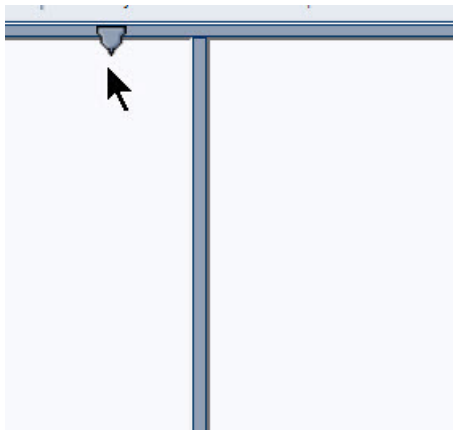


The **About** button opens a window that shows when was each Application Viewer data component recently built. Data for all application viewer components may be regenerated to incorporate up-to-date implementation-specific information. Refer to [Application Viewer Generation](#) for further details.

## Slider Icon



This "slider" icon allows you to resize the list panel and detail panel to your preferred proportions.



## Data Dictionary

The data dictionary is an interactive tool that allows you to browse the database schema and to graphically view relationships between tables in the system.

To open the data dictionary, click the [Data Dictionary button](#). You can also open the data dictionary by clicking the name of a table in other parts of the application viewer or in the online help documentation.

---

**NOTE: Data Is Generated.** A background process generated the data dictionary information. Refer to [Application Viewer Generation](#) for further details.

---

## Using the Data Dictionary List Panel

The list panel displays a list of tables and their columns. The list panel can list the table names by either their logical names or their physical names. Click the appropriate [button](#) on the toolbar to switch between the two views. The list is displayed in alphabetical order, so the order may not be the same in both views. Both views function in a similar manner.

In the list panel, you can navigate using the following options:

- Click the right arrow icon to expand a table to show its columns.
- Click the down arrow icon to collapse the column list for a table. Optionally, collapse all column lists by using the **Collapse** button.
- Click the column name to display information about the column in the detail panel.
- If the detail panel is in [related table](#) view, click the table name to view its related tables. If the detail panel is in [table detail](#) view, click the table name to display its information.

## Primary And Foreign Keys

The columns in the list panel may display key information as well as the column name:

- A yellow key indicates that the column is a primary key for the table.
- A light blue key indicates that the column is a foreign key to another table. If you hover the cursor over the icon, the tool tip indicates the foreign table.
- A dark blue key indicates that the column is a conditional foreign key. A conditional foreign key represents rare relationships between tables where a single field (or set of fields) may reference multiple primary key constraints of other tables within the application as a foreign key.
- A red key indicates that the column is a logical key field. A logical key represents an alternate unique identifier of a record based on a different set of fields than the primary key.

If you hover your cursor over an icon, the tool tip indicates the key type.

## Field Descriptions Shown

The language-specific, logical name of each field is shown adjacent to the physical column name in the data dictionary. You can enter an override label for a [table / field's](#) to be used throughout the system as the field's logical name. Here too it is the override label that is shown.

---

**NOTE: Regenerate.** You should regenerate the data dictionary after overriding labels. Refer to [Application Viewer Generation](#) for further details.

---

## Using the Data Dictionary Detail Panel


The Data Dictionary detail panel displays the details of the selected item. There are three main displays for the Detail Panel:

- Related tables view
- Table detail view
- Column detail view

## Related Tables View

The Related Tables view displays information about the table's parent tables and child tables. Click the [Schema](#) button in the toolbar to switch to related tables view.

In the related tables view, you can navigate using the following options:

- Click the left arrow and right arrow icons to view the related tables for that linked table. The List Panel is automatically positioned to the selected table.
- Click the maintenance object icon (  ) to view the table's maintenance object.
- If you want to position the [List Panel](#) to view the columns for different table click the name of the table for which you want to view the columns.

## Table Detail View

The table detail view displays information about the selected table. Click [Attributes](#) (in the toolbar) to switch to the table detail view.

In the table detail view, you can navigate using the following options:

- If user documentation is available for the table, click the View User Documentation link to read the user documentation that describes the table's maintenance object.
- If the table has an associated Language Table, click the link to view the Language Table details.
- If there is an associated Maintenance Program, click the link to view the source code for the maintenance program (you are transferred to the [Java Docs Viewer](#)).
- If there is an associated Key Table, click the link to view the Key Table details.

## Column Detail View

Click on a column name in the list panel to switch to the column detail view. The Column Detail view displays information about the selected column.

In the column detail view, you can navigate using the following options:

- If user documentation is available for the column, click the View User Documentation link to read about the column's related maintenance object.
- If the column is a foreign key, click the table name to switch to the Table Detail view for that table.
- If the column has a Value List available (normally only present for a subset of flag and switch fields), click the link to view the source code for the copybook (you are transferred to the [Java Docs Viewer](#)).

## Lookup Values

If the selected column is a lookup field its valid values are also listed. Notice that you can enter an override description for [lookup values](#). In this case the override description is shown.

---

**NOTE: Regenerate.** You should regenerate the data dictionary after overriding lookup value descriptions. Refer to [Application Viewer Generation](#) for further details.

---

## Maintenance Object Viewer

The maintenance object viewer is an interactive tool that allows you to view a schematic diagram of a maintenance object. A maintenance object is a group of tables that are maintained as a unit.

To open the Maintenance Object Viewer, click the [Maint. Object](#) button in the application viewer or click a [maintenance object icon](#) in the Data Dictionary.

---

**NOTE: Data Is Generated.** A background process generated the maintenance object information. Refer to [Application Viewer Generation](#) for further details.

---


## Using the Maintenance Object List Panel

The list panel displays a list of maintenance objects. In the list panel, you can click the maintenance object name to display information about the maintenance object in the detail panel.

## Using the Maintenance Object Detail Panel

The Maintenance Object detail panel displays a schematic of the selected maintenance object.

In the detail panel, you can navigate using the following options:

- Click a table name to transfer to the Data Dictionary [table detail view](#) for a table. (Click the [Maint. Object](#) button in the toolbar to return to the maintenance object.)
- Click the service XML icon () to view the XML file of the Service Program used to maintain the displayed object. (Click the [Maintenance Object](#) button in the toolbar to return to the maintenance object.)

## Algorithm Viewer

The algorithm viewer is an interactive tool that allows you to view algorithm types (grouped by their plug-in spot) and their related algorithms.

To open the Algorithm Viewer, click the [Algorithm](#) button in the application viewer. The Algorithm viewer may also be opened from certain locations in the online help documentation.

---



**NOTE: Data Is Generated.** A background process generates algorithm information. Refer to [Application Viewer Generation](#) for further details.

---

## Using the Algorithm Viewer List Panel

The list panel displays a list of algorithm types and their related algorithms, grouped by their plug-in spot.

In the list panel, you can navigate using the following options:

- Click the algorithm plug-in spot description to display information about the plug-in spot in the detail panel.
- Click the right pointer  icon to expand a plug-in spot and view its algorithm types and their related algorithms.
- Click the down pointer  icon to collapse the list of algorithm types for a plug-in spot.
- Click the algorithm type name to display information about the algorithm type in the detail panel.
- Click the algorithm name to display information about the algorithm in the detail panel.

## Using the Algorithm Plug-In Spot Detail Panel

The Algorithm plug-in spot detail panel displays further information about the selected plug-in spot.

## Using the Algorithm Type Detail Panel

The Algorithm Type detail panel displays further information about the selected algorithm type.

In the Algorithm Type detail panel, you can navigate using the following options:

- Click on the program name to view its source in the Java docs viewer.

## Using the Algorithm Detail Panel

The Algorithm detail panel displays further information about the selected algorithm.

## Batch Control Viewer

The batch control viewer is an interactive tool that allows you to view batch controls.

To open the Batch Control Viewer, click the [Batch Control](#) button in the application viewer. The Batch Control viewer may also be opened from certain locations in the online help documentation.

---

**NOTE: Data Is Generated.** A background process generates batch control information. Refer to [Application Viewer Generation](#) for further details.

---

## Using the Batch Control Viewer List Panel

The list panel displays a list of batch controls. The list panel can display the list of batch controls sorted by their code or sorted by their description. Click the appropriate [button](#) on the toolbar to switch between sorting by the code and description.

In the list panel, you can click the batch control to display information about the batch control in the detail panel.

---

**NOTE: Not All Batch Controls Included.** Note that the insertion and key generation programs for conversion (CIPV\*) are not included.

---

## Using the Batch Control Detail Panel

The batch control detail panel displays further information about the selected batch control.

In the batch control detail panel, you can navigate using the following options:

- Click on the program name to view its source in the Java docs viewer.
- If a To Do type references this batch control as its creation or routing process, click on the To Do type to view its detail in the To Do type viewer.

## To Do Type Viewer

The to do type viewer is an interactive tool that allows you to view to do types defined in the system.

To open the To Do Type Viewer, click the [To Do Type](#) button in the application viewer. The To Do Type viewer may also be opened from certain locations in the online help documentation.

---

**NOTE: Data Is Generated.** A background process generates To Do type information. Refer to [Application Viewer Generation](#) for further details.

---

## Using the To Do Type Viewer List Panel

The list panel displays a list of To Do types. The list panel can display the list of To Do types sorted by their code or sorted by their description. Click the appropriate [button](#) on the toolbar to switch between sorting by the code and description.

In the list panel, you can click the To Do type to display information about the To Do type in the detail panel.

## Using the To Do Type Detail Panel

The To Do type detail panel displays further information about the selected To Do type.

In the To Do type detail panel, you can navigate using the following options:

- If the To Do type references a creation process or a routing process, click on the batch process to view its detail in the batch control viewer.
- Click on the table listed in the drill key section to view its detail in the data dictionary.
- Click on the field(s) listed in the drill key section to view its detail in the data dictionary.

## Service XML Viewer

The service XML viewer is an interactive tool that allows you to browse the XML files of service programs that execute on the application server.

You can access the service XML viewer as follows:

- The maintenance object viewer allows you to view the XML file of the maintenance object's service program. This feature is implemented by viewing the maintenance object and then clicking on the [Service XML icon](#).
- When viewing a maintenance object on the [Maintenance Object](#) page, clicking the **View XML** hyperlink causes the service's XML document to be displayed in the Service XML Viewer.
- When viewing a business service on the [Business Service](#) page, clicking the **View XML** hyperlink causes the service's XML document to be displayed in the Service XML Viewer.
- When setting up a [Function](#), you may want to view the XML document used to pass data to and from the service. Clicking the **View XML** hyperlink causes the XML document to be displayed in the Service XML Viewer.

## Using the Service XML Viewer Overview Panel

The overview panel displays a high level nodes and list names structure of the XML document.

In the overview panel, you can click on any node item to position the detail panel to view that item.

## Using the Service XML Viewer Detail Panel

The detail panel displays nodes and attributes of the selected XML file.

Click the **xpath** button to view the XML path that should be used to reference the selected node in the XML document. The box at the top of the overview panel changes to display this information.

---

**NOTE: Fields in multiple lists.** Note that the generated XPath expression refers to lists using a generic “list” reference. For example: `/pageBody/list/listBody/field[@name='FIELD_NAME']`. If a service has a field that appears in more than one list, the above XPath may not be sufficient for referencing that field. In this case, references to the XPath

should be adjusted to include the list name. The list name is visible in the overview panel. To add the list name, use `[@name='LIST_NAME']`. For example: `/pageBody/list[@name='LIST_NAME']/listBody/field[@name='FIELD_NAME']`.

---

## Java Docs Viewer

The Java Docs viewer is an interactive tool that allows you to browse Java documentation files (Javadocs) for Java classes that execute on the application server.

---

**NOTE:** Proprietary Java Classes. A small number of Java classes have been suppressed due to their proprietary nature.

---

**NOTE: Classic view.** If you are more comfortable using the classic Javadocs viewer you may use the [Classic](#) button.

---

To open the Java Docs viewer from within the application viewer, click the [Java Docs button](#). Additionally, the [algorithm viewer](#) and the [batch control viewer](#) allows you to view the Javadocs of a program written in Java.

## Using the Java Docs Viewer List Panel

The list panel displays a tree of Java packages where each package may be expanded to list the Java interfaces classes it includes.

In the list panel, you can navigate using the following options:

- Click the right arrow icon to expand a Java package to view the Java interfaces and classes it includes.
- Click the down arrow icon to collapse the list for a Java package. Optionally, collapse all lists by using the **Collapse** button.
- Click the Java interface or class name to display information about it in the detail panel.

The list details panel designates the interfaces and the classes as follows:

- A green dot indicates Java interfaces.
- A blue key indicates Java classes.

If you hover the cursor over the icon, the tool tip indicates whether it's an interface or a class.

## Using the Java Package Detail Panel

The package detail panel displays a summary of the various Java classes that are included in the selected Java package.

Click the Java class name to display information about the Java class in the detail panel.

## Using the Java Interface / Class Detail Panel

The detail panel displays Java documentation information about the selected Java interface or class.

You can navigate using hyperlinks to other locations in the current detail panel or to view the details of other Java interfaces / classes.



# Application Viewer Preferences

This panel displays the Available Languages and allows you to select the language in which the labels and buttons are displayed. Select your desired language and click OK.

## Application Viewer Stand-Alone Operation

You can run the Application Viewer as a stand-alone application (i.e., you do not need to launch it from the online application environment). To run it as a stand-alone application, you should copy the Application Viewer files (all files in the appViewer directory) and the online help files (all files in the help directory) to the server on which you want to run the Application Viewer.

---

**NOTE:** Online Help. If you do not copy the online help files, online help will not be available for the Application Viewer, nor will you be able to view business descriptions of the tables' maintenance objects.

---

To start the application viewer in stand-alone mode, launch the appViewer.html file (located in the appViewer directory).

## Stand-Alone Configuration Options

You can configure the Application Viewer for stand-alone operation by modifying options in a configuration file. The Application Viewer comes with a default configuration file called config\_default.xml (located in the appViewer\config directory). Create a copy of the default configuration file and rename it to config.xml. Modify the options described in the following table to suit the needs of your installation.

---

**NOTE:** Default Configuration. If you do not create the config.xml file, the Application Viewer launches with its default (internal) configuration.

---

Option	Description
defaultLanguage	The default language used when the application viewer is started. Available values are those marked as language enabled on the language page.
defaultView	The default view then the application viewer is started. Available values include: - Data Dictionary
dataDictionary	Whether the Data Dictionary is available or not: - Y - N
sourceCode	This property is not being used. Simply enter 'N'.
baseHelpLocation	The location of the stand-alone online help in relation to the application viewer. Specify the directory structure relative to the location of the directory in which the Application Viewer files are located. Note that this is the directory in which the language subdirectories for the online help are located. The default location is: ../help
appViewerHelp	The default help topic that is launched when the Help button is clicked in the Application Viewer. Specify a help file and anchor that is under the appropriate language directory under the baseHelpLocation. The default is: Framework/Admin/91AppViewer.html#SPLINKApplication_Viewer

## Example Application Viewer Configuration

The following excerpt shows an example Application Viewer configuration.

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>
<option id="defaultLanguage">PTB</option>
<option id="defaultView">Data Dictionary</option>
<option id="dataDictionary">Y</option>
<option id="sourceCode">N</option>
<option id="baseHelpLocation">../help</option>
<option id="appViewerHelp">Framework/Admin/91AppViewer.html#SPLINKApplication_Viewer</option>
</configuration>
```

## Application Viewer Generation

The Application Viewer is initially delivered with service XML information only.

The other components of the application viewer are generated on site.

- Use the background process **F1-AVALG** to regenerate algorithm information
- Use the background process **F1-AVBT** to regenerate batch control information.
- Use the background process **F1-AVMO** to regenerate maintenance object information
- Use the background process **F1-AVTBL** to regenerate data dictionary information.
- Use the background process **F1-AVTD** to regenerate To Do type information.

These processes have been introduced so that you can more easily incorporate your implementation-specific information into the application viewer.

To keep the information shown in the application viewer current it is important to execute these background processes after you introduce changes to the corresponding system data.

---

**NOTE: Data Generation Is Not Incremental.** Each new execution of these processes first deletes existing data (if any) in the corresponding folder before it generates new data.

---

---

**NOTE: Other Extensions.** Service XML may also be extended to include implementation-specific information. The base package is provided with special scripts that handle this type of extension. Refer to the Software Development Kit User Guide for further information on application viewer extensions.

---

---

**NOTE: War File.** If your application is installed in war file format, each generation of application viewer data rebuilds the corresponding war file. The web application server then needs to be "bounced" in order to make the newly generated data available to the application viewer. Please consult your system administrator for assistance.

---

---

**NOTE: Certain Web Application Servers Are Special.** WebSphere and Oracle Application web application servers require an additional step in order to make the newly generated data available to the application viewer. These web

---

application servers require a rebuild of the application ear file and its redeployment in the web application server. This step is described in the installation document. Please consult your system administrator for further details.

---

## Defining and Designing Reports

---

This section describes how to configure your third party reporting tool and how to define your reports in the system to enable users to submit reports online.

### The Big Picture Of Reports

The topics in this section describe the approach for designing and defining your system reports.

### Integration with BI Publisher and Business Objects Enterprise

Your DBMS, your product, and BI Publisher or Business Objects Enterprise / Crystal Reports can work together to produce reports. You may choose to use a different reporting tool, but this may not be a trivial effort. This section provides high-level information about some of the business requirements that are being solved with the reporting solution.

### Reports Must Be Multi-Language

The system supports a multi-language implementation and the reporting solution for the system must also support a multi-language implementation.

- If a French-speaking user requests a given report, all labels, headings and messages are in French. If the same report is requested by an English-speaking user, all information is in English
- Different fonts may be necessary for a given report (the font for Chinese is rather different than the font for English)
- Dates and numbers must be formatted as per the user's profile in your product
- Currency must be formatted as per the currency definition in your product

In order to provide the above functionality, the third party reporting tool must do the following:

- Access the system's field and message metadata for labels, headings and messages (as different values can be defined for different languages in system's metadata)
- Retrieve the appropriate font, size, and layout based on the requested report and the user's language
- Use the currency code information in the system to format currency-oriented information
- Use the user's display profile to format date and number fields

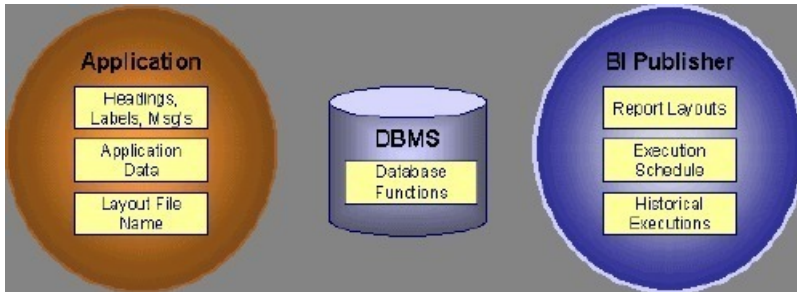
### Requesting Reports from The System

Although reports are rendered in your reporting tool, users must be able to request ad-hoc reports from within the system (assuming users have the appropriate security access).

- The prompts for the input parameters must be shown in the user's language
- Users should be able to use the standard search facilities to find parameter values
- Plug-ins can be optionally used to cross-validate input parameters
- Application security must authorize ad-hoc report requests

## Overview of the Data - BI Publisher

The following diagram provides an overview of where data is stored for your reports for integration with BI Publisher.



**Figure 5: Application and BI Publisher**

The application contains:

- The application data that appears on your reports.
- The language-specific headings, labels and messages on your reports.
- The layout file name to be used for the report.

BI Publisher contains:

- How your reports look.
- Information about scheduled reports and reports that have already run.

The DBMS contains the SQL used to retrieve the data on your reports (residing in database functions).

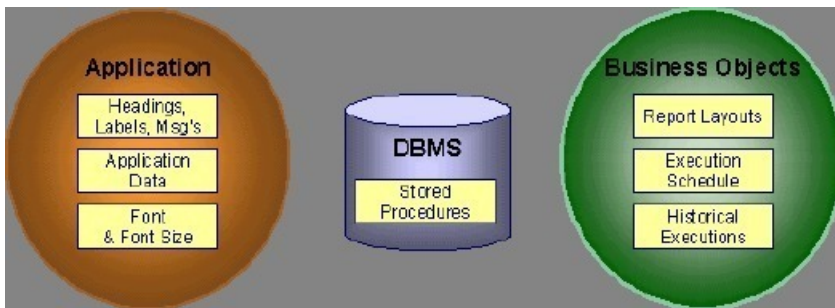
---

**NOTE:** BI Publisher can be configured to retrieve data via a service call. Because every business object can be read via a service call, elements that reside in an XML based column can be displayed on reports produced using BI Publisher. See your product's *Optional Products Installation Guide* for information on this configuration.

---

## Overview of the Data - Business Objects Enterprise

The following diagram provides an overview of where data is stored for your reports for integration with Business Objects Enterprise.



**Figure 6: Application and Business Objects Enterprise**

The application contains:

- The application data that appears on your reports.
- The language-specific headings, labels and messages on your reports.
- For Business Objects Enterprise, the font and size in which each report is rendered.

Business Objects Enterprise contains:

- How your reports look.
- When your reports are produced (the batch scheduler).
- Historical images of reports.

The DBMS contains the SQL used to retrieve the data on your reports (residing in stored procedures).

## How To Request Reports

A user may request an ad hoc report from within your product:

- A [report submission](#) page enables a user to choose the desired report and enter the parameter values for the report
- The user must be granted security access to the report
- The request is passed to the reporting tool real time. Refer to [Configure The System to Invoke BI Publisher](#) or [Configure The System to Invoke Business Objects Enterprise](#) for more information.
- The reporting tool generates the report and displays it in a new browser window

The reporting tools' scheduler creates reports (as per your schedule)

- This function is entirely within the reporting tool. No scheduling functions reside within your product.

A user can request an ad-hoc report from within the reporting tool

- Note, the user's ID must be supplied as a parameter to the report in order for the user's profile to be used to format dates and numbers

## Viewing Reports

As described above, ad-hoc reports requested from within your product are displayed immediately after they are generated in a new browser window

Crystal's report repository can be used to retrieve historical versions of a report. The [Report History](#) page allows users to open the Crystal's report execution history page and request a view of this report.

---

**NOTE:** The [Report History](#) page currently does not display historical reports for BI Publisher.

---

## Configuring The System To Enable Reports

### Configuring BI Publisher Reports

This section contains topics specific about configuring the product to interoperate with BI Publisher.

### Configure the System to Invoke BI Publisher Real-time

The base product provides an [installation algorithm](#) plug-in spot called Reporting Tool. This plug-in spot should contain an algorithm that invokes the third party reporting tool real-time.

For BI Publisher, the system provides an algorithm type called [F1-BIPR-INV](#), which invokes BI Publisher.

These algorithms rely on information defined in the [Reporting Options](#) table: the reporting server, reporting folder and the user name and password for accessing the reporting tool. The values in the reporting options should have been set up

when the system was installed. Contact your system administrator if there are any problems with the values defined on the reporting options.

To use the algorithm types to invoke BI Publisher, perform the following steps:

- Create an [algorithm](#) for the appropriate algorithm type.
- On the [installation options](#), add an entry to the algorithm collection with an algorithm entity of **Reporting Tool** and indicate the algorithm created in the previous step.

## Batch Scheduling in BI Publisher

For many of your reports, you probably want the report to be produced on a regular basis according to a scheduler. The reporting solution relies on the BI Publisher software to provide the batch scheduler functionality. Refer to BI Publisher documentation for details about configuring the batch scheduler.

---

**NOTE:** The [report history](#) page currently does not display historical reports for BI Publisher.

---

## Configuring Business Objects Enterprise Reports

This section contains topics specific about configuring the product to interoperate with Business Objects Enterprise.

### Configure the System to Invoke Business Objects Enterprise Real-time

The base product provides an [installation algorithm](#) plug-in spot called Reporting Tool. This plug-in spot should contain an algorithm that invokes the third party reporting tool real-time.

For Business Objects Enterprise, the system provides an algorithm type called [RPTE-INV](#), which invokes Business Objects Enterprise.

These algorithms rely on information defined in the [Reporting Options](#) table: the reporting server, reporting folder and the user name and password for accessing the reporting tool. The values in the reporting options should have been set up when the system was installed. Contact your system administrator if there are any problems with the values defined on the reporting options.

To use the algorithm types to invoke one of the reporting tools, perform the following steps:

- Create an [algorithm](#) for the appropriate algorithm type.
- On the [installation options](#), add an entry to the algorithm collection with an algorithm entity of **Reporting Tool** and indicate the algorithm created in the previous step.

## Batch Scheduling in Business Objects Enterprise

For many of your reports, you probably want the report to be produced on a regular basis according to a scheduler. The reporting solution relies on the Business Objects Enterprise software to provide the batch scheduler functionality. Refer to Business Objects Enterprise documentation for details about configuring the batch scheduler.

The product provides a [report history](#) page to display report instances that were produced via the batch scheduler and are stored in a repository. The report history page relies on the [reporting tool algorithm](#) to invoke Business Objects Enterprise and display the historic instances for the selected report.

## Defining Reporting Options

The reporting options are provided as a mechanism for defining information needed by your reporting solution. The base product uses the reporting options to define information needed to access the reporting tool from within the system using the algorithm defined on the installation option.

Navigate to this page using **Admin > Reporting > Reporting Options**.

### Description of page

The following information must be defined to interface with BI Publisher real-time. Contact your system administrator to report any problems with the settings defined here.

**Reporting Folder** defines the shared folder where reports are stored.

For Business Objects Enterprise, defines the name of the virtual directory on the server where Java Service pages (JSP) are located. The reporting tool algorithm uses this information to construct the URL to launch the reporting tool. The reporting tool algorithm assumes that a JSP named "logon.jsp" is located there.

**Reporting Server** defines the URL of the web application where the reporting tool is installed. For example, using BI Publisher, the format is: `http://<BI Publisher Server>:<port>`.

**Reporting Tool User ID** is not applicable when integrating with BI Publisher.

For Business Objects Enterprise, defines the user id to use when logging in.

**Reporting Tool Password** is not applicable when integrating with BI Publisher.

For Business Objects Enterprise, defines the password to use when logging in.

---

**NOTE: Customize Options.** The reporting options are customizable using the Lookup table. This field name is **RPT\_OPT\_FLG**. The reporting options provided with the system are needed to invoke the reporting tool. If your implementation requires other information to be defined as reporting options, use the lookup table to define additional values for the reporting option flag.

---

### Where Used

This information is used by the reporting tool algorithm on the [installation option](#) to invoke the reporting tool software.

Implementations may use reporting options to record other information needed for their reporting tool.

## Defining Report Definitions

For each report supplied by your installation, use the report definition page to define various attributes of the report.

### Report Definition - Main

Navigate to this page using **Admin > Reporting > Report Definition**.

---

**CAUTION:** Important! If you introduce new report definitions, you must prefix the report code with **CM**. If you do not do this, there is a slight possibility that a future release of the application could introduce a new system report with the name you allocated.

---

### Description of page

Enter an easily recognizable **Report Code** and **Description** for each report. Use the **External Reference ID** to define the identifier for this report in your external reporting tool.

Define an [application service](#) to enable users to request submission of this report online or to view report history for this report. Once you define an application service for each report, use [application security](#) to define which users may access this report.

---

**NOTE: Access Mode.** The access mode for application services related to reports must be set to **Submit/View Report**.

---

If you have more than one parameter defined for your report and you wish to perform cross-validation for more than one parameter, provide an appropriate **Validation Algorithm**. Click [here](#) to see the algorithm types available for this system event.

Enter a **Long Description** to more fully describe the functionality of this report. This information is displayed to the user when attempting to submit the report online or when viewing history for this report.

For BI Publisher, if you want to use one of the sample reports provided by the system, but with a different layout, indicate the layout to use for the report in the **Customer Specific Font/ Layout** field and BI Publisher uses this information instead. The name for base report layout is <report code>\_Base. For example, a base layout for CI\_VACANT is named CI\_VACANT\_Base.

For Business Objects Enterprise, the **Report Font** and **Report Font Size** are used to control the display of the report information. If you wish to use one of the sample reports provided by the system, but wish to use a different font and font size, indicate your **Customer Specific Font** in the **Customer Specific Font/Layout** field and Business Objects Enterprise uses this information instead.

## Report Definition - Labels

Navigate to this page using **Admin > Reporting > Report Definition** and go to the **Labels** tab.

---

**NOTE: Company name and logo.** Note the company name used as a title in the sample reports is defined as a message on the [installation options](#). For information about installing the company logo, refer to the *Reports Configuration* chapter of the *Installation Information*.

---

### Description of Page

In order to provide multi-language capability for each report, the labels used for the report must support multiple language definitions. For each label used by your report, indicate a unique **Sequence** and the **Field** used to define the **Label**. The label defined here should be the same label that is defined in your report layout defined in the external reporting tool.

When rendering an image of the report, the external reporting tool retrieves the appropriate label based on the language used for the report.

## Report Definition - Parameters

Navigate to this page using **Admin > Reporting > Report Definition** and go to the **Parameters** tab .

### Description of Page

The **Parameters** scroll contains one entry for every parameter defined for the report. To modify a parameter, simply move to a field and change its value. To add another parameter, click + to insert a row and then fill in the information for each field. The following fields display:

**Parameter Code** The identifier of the parameter. This must correspond to the parameter definition in the reporting tool.

**Required** Turn on this switch if the user must supply a value for the parameter when submitting the report.

**Sort Sequence** Indicate the sort sequence for this parameter. This sequence must match the parameter order defined in the reporting tool's report. It is also used when displaying the list of parameters on the [report submission](#) page.

**Characteristic Type** Indicate the characteristic type used to define this parameter.



**Default Value** Use this field to define a default value for this parameter. Default values are displayed to the user when the report is chosen on the [report submission](#) page.

**Description** Define a brief description of the parameter. This description is used when displaying the parameter on the [report submission](#) page.

**Long Description** Define a detailed description of the parameter. This description is used on the [report submission](#) page when the user requests more information for a given parameter.

## Sample Reports Supplied with the Product

Depending on your specific product, there may be sample reports provided that your organization may use as they are or as a starting point for creating a [new report](#). The following sections provide an overview of the sample reports along with instructions on how to use one of the sample reports in your implementation environment.

## How to Use a Sample Report Provided with the System

If you would like to use any of the sample reports, you need to perform some steps to be able to execute them in an implementation environment. This section walks you through the steps needed.

### Steps Performed at Installation Time

The *Installation Guide* provides instructions for setting up and configuring your product and reporting tool to use the sample reports provided with the system. The following steps are described there.

- Setting up the stored procedures used by the sample reports.
- Defining the company title and logo used by the sample reports. Note the company name used as a title in the sample reports is defined as a message on the [installation options](#). For information about installing the company logo, refer to the *Reports Configuration* chapter of the *Installation Information*.
- Defining a user for integration with your product.
- Publishing the sample reports in BI Publisher or Business Objects Enterprise.

Contact your system administrator to verify that the above steps have occurred.

## Subreports Used with Crystal Reports

The sample reports supplied with the system use several common subreports. Subreports are used in Crystal Reports to retrieve common data such as, labels and your company title. They are shared for all reports and may be reused for customer reports. Implementers may also use these subreports when designing new reports.

---

**NOTE: Specific Subreports.** This section only includes common subreports that may be reused by new reports. You may notice that other subreports are supplied with the system. These subreports provide functionality for a specific sample reports and are not meant for reuse.

---

## Display Company Logo and Title

The subreport **CIZCOMP** receives the user id as a parameter and calls the stored procedure **CIZCOMP**. It retrieves the company's title in the user's language from the appropriate [installation message](#) record.

---

**FASTPATH:** Refer to the **Reports Configuration** chapter of the installation guide for more information about defining the location for the company logo.

---

## Format Report Information

The subreport **CIZINST** defines shared variables that are used for formatting fields in the main report. It calls the stored procedure **CIZINST**. This subreport receives the user id and report code as parameters. It retrieves the font and font size from the [report definition](#). It retrieves the format date/time and number format from the user's [display profile](#). Finally, it retrieves the currency from the [installation](#) record and retrieves the currency symbol and position from the currency's record.

---

**NOTE: Multi-currency.** All reports support multiple currencies. Currency information is returned for each row by the appropriate stored procedure. This subreport retrieves the currency code from the Installation Options and should only be used in a report if there is no other currency information available.

---

## Labels

The subreport **CIZLABEL** keeps all labels used in the main report. It calls the stored procedure **CIZLBALL** with the user ID as a parameter. This stored procedure returns all labels defined for all reports. The subreport selects labels specified for the current report and sets shared variables L001...L100 to store the labels. If more than 100 labels are required for a new report, the version of the CIZLABEL subreport used for the new report should be changed to add additional shared variables.

## How To Define A New Report

### Use a Sample Report as a Starting Point

- Make a copy of the report and save it in an appropriate directory. Prefix the new report name with **CM**.
- Review the stored procedure(s) used for this report. Refer to the installation guide for information about where the stored procedures should be defined. If you want to change the data that is being accessed, copy the stored procedure, prefixing the new stored procedure with **CM**. Make the appropriate changes in the new version of the stored procedure. Contact your database administrator to find out the procedure for creating a new stored procedure.

---

**NOTE: Performance considerations.** When designing a stored procedure, you must consider the performance of the report when executed. Consult your database administrator when designing your database access to ensure that all issues are considered.

---

**NOTE: Defining Messages.** The stored procedures provided with the system use messages defined in message category 30. If your new stored procedures require new messages, use message category 90000 or greater, which are reserved for implementations.

---

- Review the parameters used by the report. Make appropriate changes to the parameters required by the report. This affects how you define your report. Refer to [Designing Parameters](#) for more information.
- Determine whether or not you require cross validation for your report parameters. If any cross validation is necessary, you should design an appropriate validation algorithm to be executed when requesting a report in your product. Refer to [Designing Validation Algorithms](#) for more information.

---

**NOTE: Cross Validation for On-line Submission Only.** The cross validation algorithm is only executed for ad-hoc report submissions via your product. If you submit this report through your reporting tool, this algorithm is not executed.

---

- Review the labels used by the report. Labels and other verbiage are implemented in the sample reports using a reference to the field table in the system. This enables the report to be rendered in the appropriate language for the user. For any new report label you require, you must define a new field entry. Refer to [Designing Labels](#) for more information.
- Review the layout of the report and make any desired changes based on your business needs.

When you have finished designing and coding your new report in your reporting tool, you must do the following in order for it to be usable:

- Publish the report in BI Publisher or Business Objects Enterprise. Refer to the documentation for these products for details about publishing a report. Refer to [Publishing Reports in BI Publisher](#) and [Publishing Reports in Business Objects Enterprise](#) for configuration information specific to publishing a report for integration with your product.
- Define the report. Refer to [Designing Your Report Definition](#) for more information.

## Publishing Reports in BI Publisher

Please refer to the documentation for BI Publisher for more information about publishing a report in this system. The remaining topics in this section provide information about settings needed to ensure that the report is accessible using BI Publisher.

### BI Publisher Database Access

When publishing a report in BI Publisher, you are asked for database logon information. The logon user name and password must be the user name and password that has access to the database functions related to this report in your database.

### Verify BI Publisher User Access Rights

To verify the user's access rights to folders in BI Publisher:

- Open the BI Publisher Enterprise Security Center.
- Check that the role for the user has access to the appropriate report folders.

For more information, refer to the "Understanding Users and Roles" section in the Oracle Business Intelligence Publisher User's Guide.

## Publishing Reports in Business Objects Enterprise

Please refer to the documentation for Business Objects Enterprise for more information about publishing a report in this system. The remaining topics in this section provide information about settings needed to ensure that the report is accessible using Business Objects Enterprise.

### Business Objects Enterprise Database Access

When publishing a report in Business Objects Enterprise, you are asked for database logon information. The logon user name and password must be the user name and password that has access to the stored procedures related to this report in your database.

## Verify Parameter Definition

This section describes how to verify parameter definitions in the Crystal Management Console (CMC).

- Once your report has been published, navigate to the CMC. This is the web-based administration component for Business Objects Enterprise and provides access to all administrative functions.
- To verify/change the settings of a report in the CMC go to the Objects management area and select the desired report by clicking its link located in the Object Title column.
- Once you have selected your report, click the **Parameters** tab to change the settings.
- If your report requires parameters to be provided by the user, you must configure the parameter settings in the CMC to ensure that parameter values are passed from the system when submitting the report via the [Report Submission](#) page. If you plan to submit reports from within your product, the **Prompt the user for new value(s) when viewing** check box should be checked.
- Click **OK** on this window and then click **Update**.

---

**NOTE: Submitting Reports Through Business Objects Enterprise.** If you plan to submit reports from Business Objects Enterprise, you must also define an appropriate initial value for each parameter, if applicable.

---

**NOTE: User ID.** The user id is defined as the first parameter in every sample report. This parameter is hidden when the report is submitted from within your product, but it must be defined in the Crystal report.

---

## Verify Business Objects Enterprise User Access Rights

To verify the access rights for a user in CMC:

- Navigate to the **Rights** tab in the Objects management area of the CMC and check that the user has correct security level for the report.
- Integration with your product requires an access level of **View On Demand** for the user.

## Designing Your Report Definition

When adding a new report, you must define it in the system to allow users to request ad-hoc reports from on-line and to take advantage of the multi-language provisions in the system. The following topics illustrate the steps to take to correctly configure your report definition.

### Designing Main Report Definition Values

Refer to field description section of the [report definition](#) main page for information about defining general information about the report.

For the validation algorithm, preliminary steps are required. Refer to [Designing Validation Algorithms](#) for more information.

For the application service, preliminary steps are required. Refer to [Designing Application Services](#) for more information.

### Designing Characteristic Types

The parameter tab on the report definition page uses [characteristic types](#) to define the report parameters. For each report parameter that you plan to use, you must define a characteristic type.

You do not need a unique characteristic type for each report parameter. For example, if Start Date and End Date are parameters your report, only one **Report Date** characteristic type needs to be defined. This characteristic type would be used on both date parameters.

Each characteristic type to be used as a report parameter must indicate a characteristic entity of **Report**.

To illustrate the characteristic type definitions, let's look at the sample report Tax Payables Analysis. It needs the following parameters: From Date, To Date, GL Account Type Characteristic Type and Account Type value.

**NOTE: Account Type Parameters.** The tax payables report must find general ledger entries that have posted to a certain distribution code. In order to find the appropriate distribution code, the report expects each distribution code to be defined with a characteristic indicating its GL account type (for example, **Revenue, Asset**, etc.) The report needs to know the characteristic type used to define this entry.

To support the required parameters, the following characteristic types are needed.

Char Type	Description	Type	Valid Values	Char Entities
CI_DATE	Date Parameter	Ad-hoc	(Uses validation algorithm to validate proper date entry)	Report
CI_CHTYP	Characteristic Type	FK Reference	<b>CHAR_TYP</b>	Report
CI_GLTYP	GL Account Type	Pre-defined	A- Asset, E- Expense, LM- Liability/ miscellaneous, LT- Liability/taxes, R-Revenue	Distribution Code, Report

Highlights for some of the above settings:

- We have defined a characteristic type for defining a characteristic type. This is to allow the user to indicate which Char Type on the Distribution Code is used for the GL account type. This is an FK reference type of characteristic.
- The GL account type characteristic type is referenced on both the Distribution Code entity and the report entity.

## Designing Parameters

Your report definition parameters collection must define a unique parameter entry for each parameter sent to the reporting tool. The sequence of your parameters must match the sequence defined in your reporting tool.

Continuing with the Tax Payables Analysis report as an example, let's look at the parameter definitions.

Parameter Code	Description	Char Type	Default Value
P_FROM_DT	From Date	CI_DATE	N/a
P_TO_DT	To Date	CI_DATE	N/a
P_CHAR_TYPE	Account Type Characteristic	CI_CHTYP	CI_GLTYP
P_TAX_ACCTY_CHAR	Account Type Char Value for Tax Related GL Account	CI_GLTYP	LT-Liability/taxes

Highlights for some of the above settings:

- The from date and to date parameters use the same characteristic type.
- The characteristic type parameter is defined with a default value pointing to the GL account type characteristic type.
- The GL account type parameter defines the liability/taxes account type as its default value.

**NOTE: User Id.** The sample reports provided by the system pass the user id as the first parameter passed to the reporting tool. It does not need to be defined in the parameter collection for the report.

## Designing Validation Algorithms

When designing your report definition, determine if cross validation should occur for your collection of parameters. In the Tax Payables Analysis report, there are two date parameters. Each date parameter uses the characteristic type validation algorithm to ensure that a valid date is entered. However, perhaps additional validation is needed to ensure that the start date is prior to the end date. To do this, a validation algorithm must be designed and defined on the report definition.

The system provides a sample algorithm [RPTV-DT](#) that validates that two separate date parameters do not overlap. This algorithm should be used by the Tax Payables Analysis report.

If you identify additional validation algorithm, create a new [algorithm type](#). Create an [algorithm](#) for that algorithm type with the appropriate parameter values. Plug in the new validation algorithm to the appropriate report definition.

## Designing Application Services

[Application services](#) are required in order to allow a user to submit a report on-line or to view history for a report. Define an application service for each report and define the user groups that should have submit/view access to this report.

Update [report definition](#) to reference this application service.

## Designing Labels

The system supports the rendering of a report in the language of the user. In order to support a report in multiple languages, the verbiage used in the report must be defined in a table that supports multiple languages. Some examples of verbiage in a report include the title, the labels and column headings and text such as "End of Report".

The system uses the [field](#) table to define its labels.

---

**NOTE: Report Definition.** This section assumes that your new report in the reporting tool has followed the standard followed in the sample reports and uses references to field names for all verbiage rather than hard-coding text in a single language.

---

For each label or other type of verbiage used by your report, define a field to store the text used for the verbiage.

- Navigate to the field page using **Admin > System > Field**.
- Enter a unique **Field Name**. This must correspond to the field name used in your report definition in the reporting tool and it must be prefixed with **CM**.
- Define the **Owner** as **Customer Modification**.
- Define the **Data Type** as **Character**.
- **Precision** is a required field, but is not applicable for your report fields. Enter any value here.
- Use the **Description** to define the text that should appear on the report.
- Check the **Work Field** switch. This indicates to the system that the field does not represent a field in the database.

Update the [report definition](#) to define the fields applicable for this report in the **Labels** tab.

If your installation supports multiple languages, you must define the description applicable for each supported language.

## External Messages

---

This section describes mechanisms provided in the product that enable an implementation to configure the system to communicate with an external application.

## Incoming Messages

This section provides information about support for incoming messages.

## Inbound Web Services

Inbound web service functionality is provided to support receiving web service requests from an external system.

### Overview

The following topics provide overview information about inbound web services (IWS).

#### Multiple Operations

An inbound web service supports the configuration of one or more operation per web service. Each operation defines the schema-based object to invoke to perform the desired function. An operation may refer to a Business Service, a Business Object, or a Service Script. If the IWS supports multiple operations, each operation can refer to the same or a completely different schema-based object from other operations within the IWS. In addition, each operation may define a transaction type, which is essentially an action that is supported by the schema-based object.

By default, Inbound Web Services uses the Schema Name to dictate the Request and Response for the service. The API can be overridden with custom formats by specifying Request and Response Schemas with the appropriate Request and Response XSL to transform into the relevant schema formats.

#### Annotations Used for Security

When preparing to deploy inbound web services, the security aspects of the service must be decided. There are three options available:

- Attach a security policy to the IWS via a Web Service Annotation. The product supplies the security policy **Username Token Policy** that may be used. This attaches a policy file (UsernameToken.xml) to the Inbound Web Service at deployment time.
- Attach security policies to the Inbound Web Service via the J2EE Web Application Server. This allows for multiple policies to be attached as support by the J2EE Web Application Server.
- A combination of the internal and external security policies.

#### Inbound Web Service Deployment

An Inbound Web Service must be deployed to the J2EE Web Application Server in order for it to be available to the Web Service Clients to access the system. Refer to [Deploying Web Services](#) for more information.

#### Deploying XAI Inbound Service via IWS

For implementations using XAI inbound services for external messages, the product recommends moving to the inbound web service mechanism, which uses the J2EE Web Application Server to communicate with the product rather than the XAI servlet.

For XAI inbound services that use the **Business Adapter**, it is straight forward to move to IWS because the configuration is similar. In both cases, the service is configured to reference a business object, business service or service script. The associated WSDL for each record is similar. Changing the interface for the incoming message to use IWS instead of XAI inbound services is similar.

However, for XAI inbound services that use the **Core Adapter**, these services reference an underlying “page service” in the product. For these services, the Request and Response schemas for the XAI inbound service were created using the Schema Editor. In order to support calling an underlying “page service” in IWS, first a [business service](#) must be created to reference the page service (if one doesn’t already exist). However, the resulting schema for the business service is different from the Request and Response schemas related to the XAI inbound service. Moving this functionality to IWS using business services requires changes to the format of the incoming messages.

Moving all incoming messages over to use IWS instead of XAI is the product recommendation. However, to aid in implementations that have many integrations in place using the XAI inbound services that use the **Core Adapter** (or any adapter whose message class is **BASEADA**), the product provides the ability to deploy these types of XAI inbound services to the J2EE Web Application Server along with the Inbound Web Services.

To take advantage of this capability, you must define a feature configuration option. Under the **External Messages** feature configuration type, the **Support XAI Services via IWS** is used to indicate if this feature is supported. Setting the value to **true** turns on the feature. If no option is defined for that option type, it is equivalent to setting the value to **false**.

When the system is configured to support XAI services via IWS, the [Inbound Web Service deployment](#) includes XAI inbound services (that are configured with an Adapter that references the **BASEADA** message class). The deployment portal will also include a zone showing the deployment status of these XAI Inbound Services.

## Configuring Inbound Web Service Options

This topic describes the configuration needed for using inbound web services.

### Technical Configuration

In order to use inbound web services, there are tasks a system administrator must perform.

Refer to the Server Administration Guide for technical details of each of these processes.

## Maintaining Web Service Annotation Types

The product provides the annotation type **Policy Annotation** that defines the WS-Policy annotation. If your implementation wishes to define additional annotation types, use the Web Service Annotation Type portal. Open this page using **Admin > External Message > Web Service Annotation Type**.

You are taken to the query portal where you can search for an existing web service annotation type. Once an annotation type is selected, you are brought to the maintenance portal to view and maintain the selected record.

---

**NOTE:** Use of custom policies should only be considered if the policies supplied by the J2EE Web Application Server are not sufficient for your implementation’s needs.

---

The **Web Service Annotation Type** zone provides basic information about the web service annotation type.

Please see the zone's help text for information about this zone's fields.

## Maintaining Web Service Annotations

The product provides the annotation **Username Token Policy**, which references the annotation type **Policy Annotation** and enables WS-security. If your implementation wishes to define additional annotations, use the Web Service Annotation portal. Open this page using **Admin > External Message > Web Service Annotation**.

This is a standard [All-in-One portal](#).



## Maintaining Inbound Web Services

[Inbound Web Services](#) are used to define a specific message that your implementation will receive from an external system and provides configuration needed to process the inbound message.

The product provides several inbound web services out of the box. By default, no annotations are defined for the base inbound web services. You may modify the message options or the annotations for any base IWS record. In addition, you may define additional IWS records for other incoming messages supported by your implementation.

To view an inbound web service, navigate using **Admin > External Message > Inbound Web Service**. You are brought to a query portal with options for searching for inbound web services.

Once an inbound web service has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The Inbound Web Service zone displays the configuration information for the record, including its annotations, if applicable and its operations.

## Deploying Web Services

This topic describes the configuration needed for using inbound web services.

Once an Inbound Web Service is defined it is not automatically available to the Web Service Clients to access the system. The Deployment Status and the Active flag (set to true) indicate whether a Web Service is available or not. The last step is to deploy the Inbound Web Services to the J2EE Web Application Server. This deployment phase has a number of steps that are automatically performed when a deployment is initiated:

- The Web Service files are generated and policies are attached.
- The WSDL is generated with appropriate annotations and enumerations.
- The necessary java stub code to implement the Web Service in the J2EE Web Application Server is generated and compiled.
- The Web Services are built into a valid Web Application Archive (WAR) file.
- Optionally, the newly created Web Services WAR file is deployed to the J2EE Web Application Server. This can also be done manually for clustered deployments, if desired.

There are two methods available for deploying inbound web services:

- Deployment at the command line using the **iwsdeploy[.sh]** command as outlined in the Server Administration Guide. This method is recommended for native installations and production implementations.
- Deployment using the Inbound Web Service Deployment portal. This method is recommended for development and non-production environments.

## Inbound Web Service Deployment Portal

To use the online Inbound Web Service Deployment portal, navigate using **Admin > External Message > Inbound Web Service Deployment**.

The following sections describe the base zones that are provided on the portal.

### Deploy Inbound Web Services

The Deploy Inbound Web Services zone provides information about the last deployment. Use the **Deploy** button to deploy or re-deploy inbound web services. All inbound web services whose Active switch is Yes will be deployed. All whose active switch is No will be undeployed.

---

**NOTE:** When an Inbound Web Service is deployed, the value of its service revision field is captured. Certain changes to configuration will require re-deployment to take effect. When any of the following changes occur, the IWS service revision value is incremented. This will cause the deployment status to show **Needs Deploy**.

- Active switch is changed
  - An Annotation is added or removed
  - An Operation is added or removed.
  - The Operation Name, Schema Type / Schema Name, Request or Response Schema, Request or Response XSL for an Operation is changed.
- 

**NOTE:** In addition, if the implementation supports XAI services deployed through IWS, the appropriate XAI inbound services will also be deployed or undeployed as required.

---

### Deployment Status

The Deployment Status zone displays a list of inbound web services in the product, including the deployment status.

The deployment status is determined by comparing the internal Service Revision field on each IWS against the value captured at the time of deployment.

- **Deployed.** Indicates that the IWS has been deployed and no changes have been detected to the configuration.
- **Needs Deploy.** Indicates that the IWS has never been deployed or has been deployed but in the meantime, changes have been detected to the configuration that require redeployment.
- **Undeployed.** Indicates that the IWS is marked as inactive and the IWS is not found to be deployed at this time.
- **Needs Undeploy.** Indicates that the IWS is marked as inactive but the IWS is found to be deployed at this time.

Use the broadcast button adjacent to any of the inbound web services listed in the zone to view the details of the IWS record. This causes the **Inbound Web Service** zone to appear. It is the same zone that appears on the [Inbound Web Service](#) maintenance portal.

### XAI Inbound Service Deployment Status

The XAI Inbound Service Deployment Status zone is only visible if the feature configuration option **Support XAI Services via IWS** is configured on the **External Messages** feature type or if the system detects that there are XAI inbound services that have been deployed. (The latter condition is checked for the case where an implementation has XAI inbound services deployed and then chooses to discontinue using this functionality. After changing the feature configuration option to false, one more deployment is required to “undeploy” the XAI services.) The zone displays a list of XAI inbound services in the product that are related to page services. Refer to [Deploying XAI Inbound Service via IWS](#) for more information.

The deployment status is determined by comparing the record’s Version field against the value captured at the time of deployment.

- **Deployed.** Indicates that the XAI inbound service has been deployed and no changes have been detected to the configuration.
- **Needs Deploy.** Indicates that the XAI inbound service has not been deployed or has been deployed but in the meantime, changes have been detected to the configuration.
- **Undeployed.** Indicates that the XAI inbound service is marked as inactive or the **Support XAI Services via IWS** is not set to **true** and the XAI inbound service is not found to be deployed at this time.
- **Needs Undeploy.** Indicates that the XAI inbound service is marked as inactive or the **Support XAI Services via IWS** is not set to **true** but the XAI inbound service is found to be deployed at this time.

XAI inbound service does not have the equivalent of a Service Revision field that inbound web service has, which is only incremented when changes are made to the record that impact deployment. For XAI inbound service, the version number on the record is used. This field is incremented when any changes are made, even ones that may not impact deployment. As a

result, some XAI Inbound Services may indicate “Needs Deploy” in cases where a redeployment may not be necessary. The recommendation when this occurs is to simply Deploy again to be safe.

## Guaranteed Delivery

There are alternatives for sending messages to the system besides using inbound web services. An external system may be able to send messages to the system in a generic manner where a new web service does not need to be defined for every new type of message. These types of messages may provide a payload (the message) and the service script or business service to invoke. An example of this type of communication is a message sent from a mobile application using RESTful operations.

The external system may have no mechanism for retrying failed messages. For this situation, the product provides an algorithm that may be used to capture incoming messages that should ‘guarantee delivery’. A servlet processing this type of message may invoke the [installation algorithm](#) - Guaranteed Delivery, passing the details of the message and an indication if a response should be returned. The algorithm is responsible for storing the message information in a table so that it can be subsequently processed.

---

**NOTE:** The framework does not provide a sample algorithm for this plug-in spot. Your specific product may provide an algorithm and additional support for guaranteeing messages. Refer to your product documentation for more information.

---

## Outgoing Messages

"Outgoing messages" is the term used to describe messages that are initiated by our system and sent to an external system. Messages may be sent real time or near real time. The system provides the following mechanisms for communicating messages to external systems.

- **Outbound Messages.** This method allows implementers to use configurable business objects to define the message format and to use scripts to build the message. If sent near real-time the message is posted to the outbound message table waiting for Oracle Service Bus to poll the records, apply the XSL and route the message. If sent real time, the message dispatcher routes the message immediately.
- **Web Service Adapters.** Using a web service adapter, an implementation can consume a WSDL from an external system and create an “adapter” record that references the URL of the external system and creates appropriate request and response data areas to expose the payload information in a format understood by configuration tools. A script may then be written to build the request information and initiate a real-time web service call from within the system.
- **Send Email.** The system supplies a dedicated business service that may be used to send an email real-time from within the application.

All these methods are described in more detail in the following sections.

## Outbound Messages

Outbound messages provide functionality for routing XML messages to an external system real-time or in near real time. In addition the functionality supports batching related messages to then be sent to an external system as a consolidate XML message.

For each outbound message that your implementation must initiate you define a [business object](#) for the outbound message maintenance object. Using the business object's schema definition, you define the fields that make up the XML source field. These are the fields that make up the basis for the XML message (prior to any XSL transformation).

Each outbound message requires the definition of its schema by creating a business object whose schema describes the information that is provided to the external system. An XSL transformation may then be performed when routing the message to an external system.

For each external system that may receive this message, you configure the appropriate message XSL and routing information.

Because the outbound message type is associated with a business object, your implementation can easily create outbound message records from a script using the **Invoke business object** step type. Such a script would

- Determine the appropriate **outbound message type** and **external system** based on business rules
- Access the data needed to populate the message detail
- Populate the fields in the schema and use the **Invoke business object** script step type for the outbound message type's business object to store the outbound message.
- The resulting outbound message ID is returned to the script and the scriptwriter may choose to design a subsequent step to store that ID as an audit on the record that initiated this message.

The following topics provide more information about functionality supported by outbound messages.

---

**NOTE:** For implementations that continue to use MPL and XAI, there is additional functionality related to outbound messages. Refer to [Outgoing Messages](#) for more information.

---

## Polling Outbound Messages Using OSB

If the outbound message that needs to be sent to an external system can be sent as an asynchronous message (in 'near real time'), the process initiating the outbound message should create a record in the outbound message staging table. Oracle Service Bus (OSB), is the recommended tool to use to process outbound messages in near real-time.

Outbound messages that should be processed by OSB should be configured with a processing method defined as **SOA** for the external system / outbound message type. No other information is required for defining outbound message types that are processed by OSB.

For the OSB part of the processing, the product provides a custom transport: OUF Outbound Message that may be used by an implementation to define messages to process and how to process them.

This section provides an overview of steps required to develop OSB integrations for outbound messages created by your product.

Before developing OSB integrations, a developer should be familiar with OSB development such as creating proxy services, business services, and message flow/routing. These terms are defined as follows:

**Proxy Service:** In OSB, a Proxy Service is the entity that processes a given type of message and routes it to a Business Service. A separate proxy service should be defined for each type of outbound message. If a given outbound message type may be routed to different external systems, it is the responsibility of the proxy service to query the external system defined on the outbound message and invoke the appropriate business service (see below). If any transformation is required prior to routing a message to a business service, it is the proxy service's responsibility to perform the transformation.

**Business Service:** In OSB, a Business Service is an entity that receives a message from OSB and routes it to the appropriate destination. This should not be confused with the Business Service object provided in the product in the configuration tools.

---

**FASTPATH:** Refer to the whitepaper OSB Integration for more information.

---

## Batch Message Processing

Your implementation may be required to send messages to the same destination as a single XML file with multiple messages include. The following points describe this logic:

- The individual messages that should be grouped together must have a processing method of **batch** on the external system / outbound message type record. The appropriate batch code that is responsible for grouping the messages must also be provided.
- A separate "consolidated message" outbound message type should be configured for the external system with a processing method of **SOA**.

- When outbound message records are created for the individual messages, the batch code and current batch run number are stamped on the record.
- When the batch process runs it is responsible for building the XML file that is a collection of the individual messages. This batch process should include the following steps:
  - Format appropriate header information for the collection of messages
  - Apply the individual message XSL to each message before including the message
  - Insert a new outbound message for the external system with the "consolidated message" outbound message type.
- The consolidated message is ready to be processed by Oracle Service Bus.

---

**NOTE: No process provided.** The system does not supply any sample batch job that does the above logic.

---

## Real Time Messages

The system supports the ability to make web service calls, i.e. sending real time messages, to an external system. The configuration of real time messages is similar to the configuration of near real-time ones, with the following exceptions:

- The processing method defined for the outbound message type and an external system must be **Real-time**.
- The [message sender](#) defined for the outbound message type and an external system has to be set up with **Real-time** invocation type.

Just like near real-time messages, you can easily create outbound message records from a script. When a real time message is added, the system immediately routes it to the external system. If the external system provided a response message back, the system captures the response on the outbound message. If the outbound message type for the external system is associated with a response XSL it is applied to transform the response. In this case the system captures the raw response as well on the outbound message.

Any error (that can be trapped) causes the outbound message to be in a state of **Error**. It is the responsibility of the calling process to check upon the state of the outbound message and take a programmatic action. When the outbound message state is changed back to **Pending** the message will be retried.

The base package provides a business service called "Outbound Message Dispatcher (**F1-OutmsgDispatcher**)" that further facilitates making web service calls, allowing the calling script to configure the following behavior:

- Whether or not the sent message is captured as an actual outbound message record.
- Whether or not exceptions encountered while sending the message are trapped. Trapping errors allows the calling script to interrogate any errors encountered and take some other programmatic action.

Refer to the description of the business service for a better understanding on how it works.

---

**NOTE: HTTP Sender.** In the current release of the product, only senders that communicate via HTTP are supported.

---

## Configuring the System for Outbound Messages

The following sections describe the setup required when using [outbound messages](#) to communicate with an external system. The configuration walks you through the steps to configure a single external system and all its messages.

### Define the Outbound Message Type

For each outbound message that must be sent to an external system, create a [business object](#) for the outbound message maintenance object. Using the business object's schema definition, your implementation defines the fields that make up the XML source field. These are the fields that are the basis for the XML message. XSL transformations may be applied to this XML source to produce the XML message.

Once you have your business object and schema, define an [outbound message type](#) for each unique outbound message.

## Define the Message Sender

When messages are routed to an external system real-time using the outbound message dispatcher or using the real-time send email business service, there must be a [Message Sender](#), which tells the system how to send the message. The two different mechanisms differ in how the system knows which sender to use:

- For sending emails, the default sender may be defined on the [message option](#) table. Alternatively, the message sender can be provided to the business service as input. Refer to [Sending Email](#) for more information. Senders of this type should be configured with the **RTEMAILSNDR** class.
- For other outbound messages that are routed using the real-time outbound message dispatcher, the sender to use is defined on the [external system](#) / outbound message type collection. Refer to [Real Time Messages](#) for more information. Senders of this type should be configured with the **RTHTTPSNDR** class.

## Define the External System and Configure the Messages

Define an [external system](#) and configure the valid outgoing messages and their method of communication (processing method). Refer to [Outbound Messages](#) for more information.

## Message Sender

The topics in this section describe the maintenance of a message sender

### Message Sender - Main

To define a new sender, open **Admin > External Message > Message Sender**.

#### Description of Page

Enter a unique **Message Sender** and **Description**.

Use **Invocation Type** to define whether the sender is a **Real-time** sender or called by **MPL** to route near real-time messages.

---

**NOTE:** The MPL invocation type remains in the product for upgrade purposes but is not recommended.

---

Indicate the **Message Class** for this sender. The class should be one that is defined for a sender. The real-time sender classes are **EMAILSENDER**- Email sender and **HTTPSNDR**- HTTP sender.

The following sender classes are related to MPL processing and remain in the product for upgrade purposes: **DWNSTGSNDR**- Download Staging sender, **FLATFILESNDR**- Flat file sender, **JMSENDER**- JMS Queue sender, **STGSENDER**- Staging Upload sender, **TPCSNDR**- JMS Topic sender and **UPLDERRHANDLR**- Upload Error Handler.

Indicate whether or not this sender is currently **Active**.

Indicate whether the **MSG Encoding** is **ANSI message encoding** or **UTF-8 message encoding**.

If the Message Class is **JMSENDER** or **TPCSNDR** indicate the appropriate **XAI JMS Connection**

---

**FASTPATH:** Refer to [XAI JMS Connection](#) for more information.

---

If the Message Class is **JMSENDER**, use the **XAI JMS Queue** to define where the response is to be sent.

---

**FASTPATH:** Refer to [XAI JMS Queue](#) for more information.

---

If the Message Class is **TPCSNDR**, use the **XAI JMS Topic** to define where the response is to be sent.

---

**FASTPATH:** Refer to [XAI JMS Topic](#) for more information.

---

If the Message Class for this sender is **STGSENDER** indicate the **XAI JDBC Connection**.

---

**FASTPATH:** Refer to [XAI JDBC Connection](#) for more information.

---

## Message Sender - Context

The sender may require context information to define additional information needed by the system to successfully send outgoing messages. Open **Admin > External Message > Message Sender** and navigate to the **Context** page.

### Description of Page

Define the **Context Type** and **Context Value**, which contain parameters for senders when more information is required. See below for some examples of context for different types of senders.

---

**NOTE:** The values for the Context Type field are customizable using the Lookup table. This field name is **SENDER\_CTXFLG**.

---

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_SENDER](#).

## Email Sender Context

The email sender is used by the business service that [sends email messages real-time](#).

An email sender must point to the Message Class **EMAILSENDER**. In addition, the following context records should be defined for senders of this type.

Context Type	Description
SMTP Host name	The SMTP server host name.
SMTP Username	The user ID used to access the SMTP server.
SMTP Password	The password used to access the SMTP server.
Response Time Out	The amount of time the system should wait for a real time response.

## HTTP Sender

An HTTP sender is one that sends messages to an HTTP server using the HTTP protocol. HTTP senders should reference a Message Class of **HTTPSNDR**.

Various parameters are required to establish a session with the target HTTP server. You specify these parameters by defining a collection of context values for the sender. A set of context types related to HTTP variables is provided with the product. The following section describes the context types and where appropriate, indicates valid values.

Before defining the HTTP sender, you need to find out how the HTTP server on the other side expects to receive the request, and in particular, to answer the following questions:

- What is the address of the HTTP server?
- Is the HTTP server using a POST or GET HTTP method?
- If the server is using POST, how are message contents passed? Does it use an HTTP FORM or does it pass the data in the body of an XML message?



Context Type	Description	Values
HTTP URL1 - URL9	<p>Used to construct the URL of the target HTTP server.</p> <p>Since the URL may be long and complex, you can break it into smaller parts, each defined by a separate context record. The full URL is built by concatenating the values in URL1 through URL9.</p> <p>You may use substitution variables when entering values for URL parts.</p>	
HTTP Method	The HTTP method used to send the message.	<b>POST</b> or <b>GET</b>
HTTP Proxy Host	If connecting to the remote system requires using an HTTP Proxy, then this context field can be used to configure the HTTP Proxy Host. If the Proxy Host is set, the Sender class must use the value specified to connect to the remote system via a proxy.	
HTTP Proxy Port	If connecting to the remote system requires using an HTTP Proxy, then this context field can be used to configure the HTTP Proxy Port. If the Proxy Port is set, the Sender class must use the value specified to connect to the remote system via a proxy. If the HTTP Proxy Host is not set, HTTP Proxy Port is ignored and the connection will be made directly to the remote system.	
HTTP Transport Method	Specifies the type of the message. You can either send the message or send and wait for a response.	<b>Send</b> or <b>sendReceive</b>
HTTP Form Data	<p>Used when the message is in the format of an HTML Form ( <code>Content-Type: application/x-www-form-urlencoded</code>).</p> <p>This context specifies the form parameters (data) that should be passed in the HTTP message. Since a form may have multiple parameters, you should add a context record for each form parameter.</p> <p>The value of a form parameter takes the format of <code>x=y</code> where <code>x</code> is the form parameter name and <code>y</code> is its value.</p> <p>If <code>y</code> contains the string <code>@XMLMSG@</code> (case sensitive) then this string is replaced by the content of the service response XML message.</p> <p>If a context record of this type is defined for a sender, the sender uses the HTML Form message format to send the message even if <code>@XMLMSG@</code> is not specified in one of the context records.</p>	



Context Type	Description	Values
	<p>If a context record of this type is not defined for a sender, then the XML is sent with <code>Content-Type: text/plain</code>. When using POST it is put in the HTTP message body.</p> <p>Always required when using the GET method. If you are using the GET method and do not specify a Form Data context record, no message is transferred to the HTTP server.</p> <p>The MPL server builds formData by concatenating the individual parts.</p> <p>You may use substitution variables when entering values for Form Data.</p>	
HTTP Login User	The HTTP server may require authentication. Add a context record of this type to specify the login user to use.	
HTTP Login Password	The HTTP server may require authentication. Add a context record of this type to specify the login password to use.	
HTTP Header	<p>Sometimes the HTTP server on the other side may require the addition of HTTP headers to the message.</p> <p>For each HTTP header that has to be specified you should add a context record with a value having the following format <code>x:y</code> where <code>x</code> is the header name and <code>y</code> is the value for the header</p>	
HTTP Time Out	Indicates the amount of time to wait for a connection to be established with the remote system.	
Character Encoding	Indicates if the message should be encoded. The sender will add to the HTTP's content type header the string <code>; charset=x</code> where <code>x</code> is the value of this context and when sending the message it will encode the data in that encoding.	<b>UTF-8 or UTF-16</b>
Response Time Out	The amount of time the system should wait for a real time response.	

## Defining Outbound Message Types

Use this page to define basic information about an outbound message type. Open this page using **Admin > External Message > Outbound Message Type**.

**NOTE:** This page is not available if the **External Message** module is [turned off](#).

### Description of Page

Enter a unique **Outbound Message Type** and **Description**. Use the **Detailed Description** to describe the outbound message type in detail.

Indicate the Business Object that defines business rules and the schema for outbound messages of this type.

Indicate the relative **Priority** for processing outbound message records of this type with respect to other types.

This bottom of this page contains a [tree](#) that shows the various objects linked to the outbound message type. You can use this tree to both view high-level information about these objects and to transfer to the respective page in which an object is maintained.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [F1\\_OUTMSG\\_TYPE](#).

## External Systems

Use this page to define an external system and define the configuration for communication between your system and the external system.

### External System - Main

Open this page using **Admin > External Message > External System**.

---

**NOTE:** This page is not available if both the **External Message** and the **Open Market Interchange** modules are [turned off](#).

---

#### Description of Page

Enter a unique **External System** and **Description**.

Use the field **Our Name In Their System** to specify the identity of your organization (i.e., your external system identifier) in the external system.

---

**NOTE:** The workflow process profile and notification download profile are only applicable to products that support workflow and notification. They are not visible in the product if the **Open Market Interchange** module is [turned off](#).

---

If this external system sends inbound communications through notification upload staging, the type of workflow process that is created is controlled by the sender's **W/F (Workflow) Process Profile**.

If you send notifications to this external system, select a **Notification DL (download) Profile** that is used to define the configuration of the outgoing messages.

---

**NOTE:** The remaining fields are not visible if the **External Message** module is [turned off](#).

---

Set **Usage** to **Template External System** for external systems whose outbound message type configuration is inherited by other external systems.

If the outbound message type configuration should be inherited from a template external system, define the **Template External System**. If this field is specified, the outbound message type collection displays the data defined for the template system as display-only.

The **Outbound Message Type**[accordion](#) contains an entry for every type of outbound message defined for this external system. For each type of outbound message identify its **Outbound Message Type**.

Define the **Processing Method** for messages of this type. If the value is **XAI** or **Real-time**, indicate the appropriate **Message Sender**. If the value is **Batch**, indicate the appropriate **Batch Control**.

The **Message XSL** is the schema used to transform information from the format produced by the system to a format understood by the sender, who receives a message of this type. This is not applicable for Processing Method of **SOA**.

Enter the file name of the appropriate **W3C Schema** if you want to validate the message built for outbound messages for this external system / outbound message type prior to being routed to their destination. Refer to [Outbound Message Schema Validation](#) for more information. This is not applicable for Processing Method of **SOA**.

**Response XSL** will have the same search service as is used for the existing Message XSL field. This field will only be displayed when the processing method is **Real-time**. Refer to [Real Time Messages](#) for more information on how it is used.

## External System - Template Use

If you are viewing an external system whose usage is a **Template External System**, use this page to view the other external systems that reference this one. Open this page using **Admin > External Message > External System** and then navigate to the **Template Use** tab.

### Description of Page

The tree shows every external system that references this external system as its template.

## Managing Outbound Messages

Use this page to view information about outbound messages.

### Outbound Message - Main

Open this page using **Menu > External Message > Outbound Message**.

#### Description of Page

**Outbound Message ID** is the system-assigned unique identifier of the outbound message. These values only appear after the outbound message is added to the database.

The **Processing Method** indicates whether this record will be processed by a **Batch** extract process, through the **XAI** tool or **Real-time**. The value defined on the external system / outbound message type collection populates this value.

When records are created with a processing method of **Batch**, the system sets Extract to **Can Be Extracted**. Change the value to **Not to be extracted** if there is some reason that processing should be held for this record.

For records with a processing method of **Batch**, **Batch Control** indicates the process that will extract this record. This value is populated based on the on the external system / outbound message type's value. **Batch Number** indicates in which batch run this record was extracted or will be extracted.

The **Retry Count** is used by the XAI tool to keep track of how many times the tool tried to process this record and could not process the record, resulting in an error.

The **Creation Date** indicates the date that this record was created.

If the processing method is **XAI**, **Status** defines the state of the outbound message record. Refer to [Lifecycle of Outbound Message](#) for more information.

For messages in **error** status, click **Pending** to change the status back to pending for reprocessing.

For messages in **pending**, **error**, or **in progress** status, click **Cancel** to cancel the message and prevent further processing.

### Outbound Message - Message

Use this page to view the XML source used to build an outbound message. Open this page using **Menu > External Message > Outbound Message** and then navigate to the **Message** tab.

## Description of Page

The **XML Source** is displayed.

If a message XSL is defined on the external system / outbound message type record linked to this outbound message, the **Show XML** button is enabled. Click this button to view the XML that is a result of applying the Message XSL to the XML source.

## Outbound Message - Response

Use this page to display the XML response. Open this page using **Menu > External Message > Outbound Message** and then navigate to the **Response** tab.

### Description of Page

The **XML Response** and optionally the **XML Raw Response** is displayed.

XML Response displays the response data from the system called by the real-time message. If a response XSL is defined on the external system / outbound message type record linked to this outbound message, a transform is performed and the XML Raw Response displays the original, unchanged response.

## Message Option

The Message Option page defines various system settings used by the system when processing external messages.

Note that some of the options are only applicable to implementations still using the XAI and MPL servers. The settings here may be overridden by the [AdHoc Parameters section](#) of the XAIParameterInfo.xml or MPLParameterInfo.xml.

To define options for your environment, open **Admin > External Message > Message Option**.

### Description of Page

The following options are supported.

Option	Description	MPL / XAI Option Name
Automatically Attempt Resend to Unavailable Sender (Y/N)	Set to <b>Y</b> if you wish to enable <a href="#">Automatic Resend</a> . Set to <b>N</b> if you wish to log errors when the system fails to send an outgoing message.	shouldAutoResend
Default Email Sender	This is the default Message Sender used for sending e-mails when no explicit Message Sender is specified.	defaultEmailSender
Default Response Character Encoding	Determines the character encoding to be used when a response is sent. For example, you may specify <b>UTF-8</b> or <b>UTF-16</b> . If no value is specified then the default is <b>UTF-8</b> . If no special encoding should be done, then enter the value <b>none</b> .	defaultResponseEncoding
Default User	The default user is used by XAI to access your product when no other user is explicitly specified. Refer to <a href="#">Server Security</a> for more information. Additionally, the Default User is used for MPL transactions where there is no facility to provide a User ID. For example, no facility exists to provide a user id when reading messages from a JMS Queue. In these messaging scenarios, the system will use the Default User for authorization purposes.	defaultUser
Email Attachment File Location	This is the default location of e-mail attachment files. If not specified, the e-mail service provided with the product assumes a full path is provided with each attachment file.	emailAttachmentFileLocation

Email XSL File Location	This is the default location of e-mail XSL files. If not specified, the e-mail service provided with the product assumes a full path is provided to an XSL file as part of an e-mail request.	emailXSLFileLocation
JDBC Connection Pool Max size	The MPL uses a pool of JDBC connections to connect to the database. This option determines the maximum number of JDBC connections that can be opened in the pool. The default value is 100.	JDBConnPoolMaxSize
Maximum Errors for a Sender	This value is required if you have enabled <a href="#">Automatic Resend</a> . It defines how many errors you receive from a sender when attempting to send an outgoing message before you mark the sender <b>unavailable</b> .	maxSendingErrors
Messages JDBC Connection	Specifies the JDBC connection that XAI uses to read the text for its messages.	messagesJDBCConnection
Messages Language	The default language to use for the messages.	language
MPL Administrator Port	The port number to be used for receiving MPL operator commands.	adminPort
MPL HTTP Server Authentication Method	This setting, along with the MPL HTTP Server User and Password are used to secure commands received by your MPL (such as those issued via <a href="#">XAI Command</a> ) through HTTP. Currently only <b>BASIC</b> authentication is supported.	MPLHTTPAuthMethod
MPL HTTP Server Password	This setting, along with the MPL HTTP Server Authentication Method and User are used to secure commands received by your MPL (such as those issued via <a href="#">XAI Command</a> ) through HTTP. The password should be in encrypted form, using the same encryption that is used for the database password. .	MPLHTTPAuthPassword
MPL HTTP Server User	This setting, along with the MPL HTTP Server Authentication Method and Password are used to secure commands received by your MPL (such as those issued via <a href="#">XAI Command</a> ) through HTTP.	MPLHTTPAuthUser
MPL Log File	The MPL Log File setting is used to specify the name of the file where MPL log information is to be written. The log contains technical information about the operation of the MPL.	MPLLogFile
MPL Trace File	The MPL Trace File setting is used to specify the name of the file where MPL trace information is to be written.	MPLTraceFile
MPL Trace Type	The MPL Trace Type is used to enable or disable tracing of the MPL. The possible values are <b>FULL</b> - All trace messages are written to the log file and <b>NOLOG</b> - No information is written to the log file.	MPLTraceType
Outbound Message Schema Location	Enter the full path of the virtual directory where valid W3C schemas are stored if your implementation wants to <a href="#">validate outbound message schemas</a> . For example: http://localhost/cisxai/schemas.	xaiOuboundSchemaLoc
Privileged Users	Comma separated list of users that are allowed to specify an effective User or effective User Id via framework custom SOAP Headers.	superUsers
Records MPL Receiver Will Process At a Time	If your implementation has <a href="#">configured multiple MPL servers</a> , indicate the number of records that each MPL receiver should process.	Not currently used

Schema Directory	The full path of the virtual directory where XML schemas are stored. For example: <code>http://localhost/cisxai/schemas</code> . If this option is not specified, the XAI uses the current directory, from where it is being run, to locate schemas.	<code>schemaDir</code>
Schema Validation Flag	Enter <b>Y</b> to turn on <a href="#">schema validation for outbound messages</a> . Enter <b>N</b> to turn this off.	<code>xaiSchemaValidationCheck</code>
Send SOAP Fault as HTTP 500	Enter <b>Y</b> to ensure that a SOAP error is reported as an HTTP 500 "internal server error".	<code>sendErrorAsHttp500</code>
Sender Retry Seconds	This value is required if you have enabled <a href="#">Automatic Resend</a> . It defines how many seconds to wait after marking a sender <b>unavailable</b> before you mark the sender <b>available</b> again (and retry sending messages to it).	<code>senderWaitTime</code>
System Error JDBC Connection	When a request fails to execute due to a system error, the MPL retries its execution several times. The MPL registers the system error in a table and uses this table for the retries. This setting specifies the JDBC connection required to access this table. Only enter a value in this field if it is different from the database environment used to read the XAI registry.	<code>systemErrorTableJDBCConnection</code>
System Error Max Retry	When a request fails to execute due to a system error, the MPL retries its execution several times until a maximum number of retries is reached. This option specifies the maximum number of retries.	<code>systemErrorMaxRetries</code>
System Error Retry Interval	When a request fails to execute due to a system error, the MPL retries its execution several times. This option specifies the number of seconds the MPL server waits between retries.	<code>systemErrorRetryInterval</code>
Thread Pool Initial Size	The MPL uses a thread of pools to enhance performance. The MPL starts with a minimum number of threads and grows/shrinks the pool based on the MPL system load. This option specifies the initial number of threads in the thread pool. The minimum number of threads is 12.	<code>threadPoolInitialSize</code>
Thread Pool Max Size	This option specifies the maximum number of threads in the thread pool.	<code>threadPoolMaxSize</code>
Thread Pool Non Activity Time	This option specifies how long a thread in the pool may be inactive before it is timed out and released from the pool.	<code>poolNoneActivityTime</code>
To Do Type for Inbound JMS Message Errors	To Do type for inbound JMS message errors. The inbound message processor uses this To Do type when creating To Do entries for inbound JMS messages that cannot be successfully processed. The system provides the To Do type <b>F1-INJMS</b> that may be used here.	<code>todoTypeforInboundJMSMessageErrors</code>
To Do Type for Outbound Message Errors	To Do type for outbound message errors. The outbound message receiver uses this To Do type when creating To Do entries for outbound messages that cannot be successfully processed. The system provides the To Do type <b>F1-OUTMS</b> that may be used here.	<code>outboundErrorTodo</code>
WSDL Service Address Location	Specifies the SOAP address location that XAI uses in creating a WSDL. If no value is present, the XAI's URL is used.	<code>wsdlAddressLocation</code>

XAI Authentication Password	The multi-purpose listener uses this field in combination with the <b>XAI Authentication User</b> when attempting to communicate with the XAI server over HTTP, which is running on a secured servlet and requires authentication.	HTTPBasicAuthPassword
XAI Authentication User	The multi-purpose listener uses this field in combination with the <b>XAI Authentication Password</b> when attempting to communicate with the XAI server over HTTP, which is running on a secured servlet and requires authentication.	HTTPBasicAuthUser
XAI Trace File	The full path name for the file, where the XML messages should be written. For example: c:\inetpub\wwwroot\cisxai\xai.log.	traceFile
XAI Trace Type	Use this option to specify the level of logging. The possible values are <b>FULL</b> - All XML messages are written to the log file and <b>NOLOG</b> - No information is written to the log file.  <b>FASTPATH:</b> Refer to <a href="#">Server Trace</a> for more information about tracing.	traceType
XSL Directory	The full path of the virtual directory where XSL transformation scripts are located. XSL transformation scripts can be defined for each service. By default, this is the same directory as the schemas directory.	XSLDir

### Where Used

Used by the XAI tool to obtain various required settings and locations.

## Web Service Adapters

The base product provides a configuration object called Web Service Adapter that is used to help build configuration objects to allow for functionality in the system to initiate a web service call from within the system. A Web Service Adapter provides the following functionality:

- WSDL (web service description language) import. An implementer can use the WSDL import functionality to read the details of a WSDL into the system
- Internal API generation. The system generates internal data areas that have two main purposes: they provide the API for custom code to define the appropriate input and they provide output data for the web service call using Oracle Utilities Application Framework schema language. In addition, the web service dispatcher uses element mapping defined in the data areas to transform the internal XML into the structure expected by the external system as described in the WSDL.
- Defines the URL needed to perform the web service call at runtime.

## Understanding Web Service Adapters

The following topics describe the system functionality in more detail.

### Importing a WSDL

Configuring a Web Service Adapter starts by identifying the WSDL (the web service description language document used to define the interface) that will be provided by the external system. The following steps describe the base product functionality provided to allow a user to import a WSDL.

- Navigate to the **Web Service Adapter** page in add mode and select the appropriate base business object.

- Enter a meaningful Web Service Name and appropriate descriptions.
- Provide the URL of the given WSDL.
- Click **Import** to retrieve the details of the WSDL. The system then parses the WSDL details and populates the WSDL Service Name, WSDL Source, WSDL Port, URL and a list of Operations (methods) defined in the WSDL.
- Determine which Operations should be **active** based on the business requirements for invoking this web service. **Active** operations are those that the implementation is planning to invoke from the system. These require appropriate request and response data areas generated for them. The following section provides more information about that.
- Specify the appropriate Security Type to configure the type of security to use when invoking this web service.
- Click Save.

At this point, a web service adapter record is created in pending status. The next step is to generate the request and response data areas for the operations configured as active.

## Generating Request and Response Data Areas

Each **active** operation for the web service adapter requires a pair of data areas, request and response, that represent the request and response XML messages for the operation.

The base product provides steps to generate the data areas as follows:

- As described in the Importing a WSDL section above, the operations listed in the WSDL are generated for the web service adapter and the implementer should indicate which operation to activate.
- After saving the **pending** web service adapter, the display lists all the active operations and for each one includes a **Generate** button.
- After clicking **Generate** for an operation, a window appears where the names of the new Request and Response Data Areas may be defined. Click **Save** to generate the data areas.

The generated data areas provide the API for the implementer to use when implementing the web service call in an appropriate algorithm or service in the system. The data areas contain the appropriate mapping from the elements that the implementer works within the code that invokes the web services and the WSDL definitions.

To facilitate generating the request and response data areas, the base product invokes a special business service used to create the appropriate mapping. The business service is defined as a BO option on the Web Service Adapter business object. This allows an implementation to provide a custom business service to further enhance the request and response mapping where appropriate.

---

### NOTE:

**Generated data areas.** It is possible to edit and modify the generated data areas after they are created. An implementer can change element names or remove unneeded elements if desired. Manually changing the generated data areas must be done only when absolutely necessary. This is because the system is not able to validate manual changes and issues with the data areas would only be detected at run time.

---

## Activating Web Service Adapters

The business objects provided by the base package for web service adapters include a simple lifecycle of **Pending** and **Active**. Configure the web service adapter and its data areas while in **Pending** status and activate it when it is ready to be implemented in the appropriate system functionality.

## Invoking Web Services

To make a call to a web service using a web service adapter, the system has provided a Web Service Dispatcher business service (**F1-InvokeWebService**) to submit a web service call. The calling program is responsible for retrieving all the information to correctly populate the request data required by the web service call before invoking the business service.

---

### NOTE:



Refer to the detailed description of the business service for more information.

---

## Limitations

The following points highlight limitations associated with the types of web services that the system supports:

- It is possible for one WSDL document to contain definitions for several web services. The system currently supports only one port or service per WSDL document.
- It is possible for a WSDL to support multiple message patterns. The system currently supports only request / response.

## Setting Up Web Service Adapters

Use the Web Service Adapter portal to define the configuration needed to communicate with an external system using a web service call. Open this page using **Admin > External Message > Web Service Adapter**. You are brought to a query portal with options for searching for web service adapters.

Once a web service adapter has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Web Service Adapter** zone provides basic information about the web service annotation type.

Please see the zone's help text for information about this zone's fields.

---

**FASTPATH:** Refer to [Understanding Web Service Adapters](#) for information about common web service adapter functionality.

---

## Sending Email

The framework provides the ability to initiate an email from within the system. The following topics highlight the functionality available.

- Sending email “real time”. The framework provides a business service **F1-EmailService** that supports sending an email. The schema supports elements for all the information required to create an email. It also supports sending attachments. The SMTP information (host, user name and password) may be provided or may be defined on a message sender, that may be provided as input. Review the business service schema for information about the input elements.

---

**NOTE:** Validating attachments. If a Validate Email Attachment algorithm is plugged into the [installation record](#), it is called to validate the attachments supplied, if applicable.

---

- Staging records to be emailed in “near real time”. This option may be used by options that are creating emails in bulk. This technique uses the outbound message table to capture the details of the email message.

## XML Application Integration

This section describes the XML Application Integration (XAI) utility, which enables you to configure your system to receive information from and to send information to external applications using Extensible Markup Language (XML) documents.

---

**NOTE:** The XAI functionality is legacy functionality and not recommended for new implementations. The topics for the functionality outlined in the previous sections describe the recommended features for supporting sending and receiving external messages. The XAI information remains in the documentation for upgrade purposes.

---

## The Big Picture of XAI

The XML Application Integration (XAI) module provides the tools and infrastructure that businesses require for integrating their applications with your product. The integration your product with other systems across organizational boundaries or business is made possible, regardless of the platforms or operating system used. XAI provides an integration platform that enables the following:

- Integrate with Customer Relationship Management (CRM) systems
- Provide information feeds for web based customer portals
- Fit seamlessly with web based applications
- Facilitate fast implementation of batch interfaces
- Integrate with other XML compliant enterprise applications

XAI exposes system business objects as a set of XML based web services. The service can be invoked via different methods, e.g., Hypertext Transfer Protocol (HTTP) or Java Message Service (JMS). Consequently, any application or tool that can send and receive XML documents can now access the rich set of system business objects. Business-to-Business (B2B) or Business-to-Consumer (B2C) integration with other enterprise applications as well as the setup of web portals is made very simple and straightforward.

## XAI Architecture

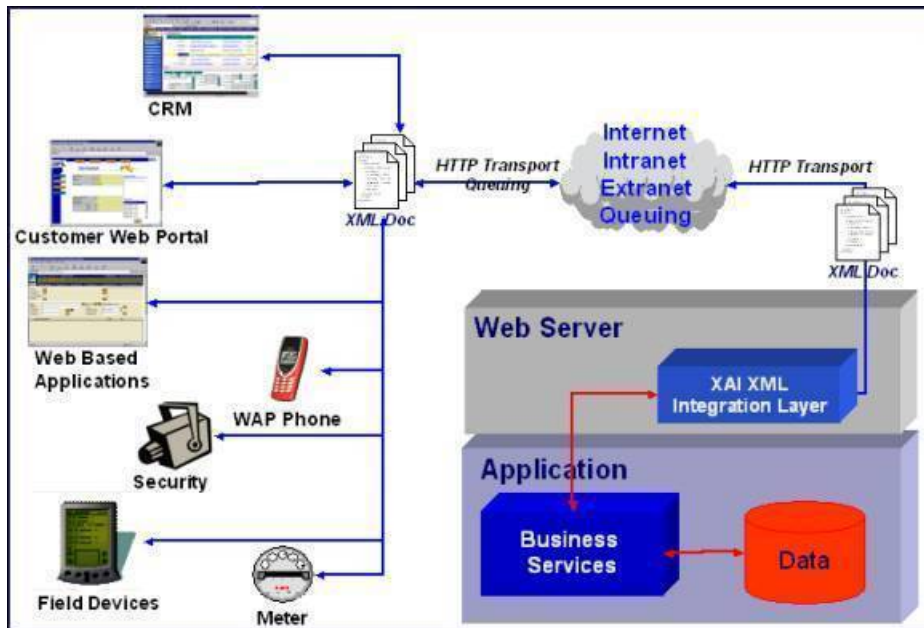
The XAI architecture contains 3 major components:

- The Core Server Component
- The Multi Purpose Listener (MPL)
- The Client Component

## The Core Server Component

The core server component is responsible for receiving XML requests, executing the service and returning the response to the requester.

The following diagram shows the XAI tool operating on a web server and providing integration between the system business objects and various external applications.



The core is built in Java, using a layered, scalable architecture. The basic transport protocol supported by the core is SOAP/HTTP.

---

**FASTPATH:** Refer to [SOAP](#) for more information.

---

The XAI core server provides a Java servlet to process requests through HTTP. You may also use other messaging mechanisms such as message queuing, publish/subscribe or a staging table. The multi-purpose listener processes the messages.

## The Multi Purpose Listener (MPL)

---

**NOTE:** Multi Purpose Listener functionality is legacy functionality that is not recommended going forward. The Oracle Service Bus (OSB) is the recommended tool. This section remains in place for upgrade purposes.

---

The Multi Purpose Listener (MPL) is a multi-threaded Java server that constantly reads XML requests from various external and internal data sources, such as a Java Message Service (JMS) message queue, a JMS topic or system staging tables.

The MPL can be used to process inbound messages (those sent by an external application to invoke a system service), or outgoing messages (those sent by your product to external applications). The MPL uses different receivers to process messages from different data sources.

A receiver is implemented using 3 distinct layers:

- The Receiving Layer
- The Execution Layer
- The Response Layer

## The Receiving Layer

This layer deals with polling various locations to determine if new records, files or incoming requests exist. The various locations include:

- Staging tables, including [XAI staging control](#), [XAI upload staging](#), notification download staging (certain products only), and [outbound message](#).

- An external directory that contains a file, for example a comma delimited file or an XML file.
- A JMS queue/topic.

A separate receiver is defined to read requests from each of these locations. When the MPL server starts, it looks for all defined active receivers in the [XAI Receiver](#) table, and for each receiver it starts a thread that constantly fetches messages from the message source.

Once a request message is read, it is passed to an execution thread that implements the execution layer. Each receiver references an [Executer](#) that is responsible for executing the request.

## Configuring Multiple MPL Servers

A single MPL server may only run one of each of the above staging table receivers for a given JDBC connection. To enhance the performance of the processing of the staging tables, you may define multiple MPL servers where each one runs the active receivers defined in the receiver table.

To ensure that each staging table receiver processes its own set of records in the staging table, the receiver selects a set number of records (specified as [Message option](#) **Number of Records an MPL Receiver Will Process At a Time**) and marks those records with the IP address and port number of the MPL.

## The Execution Layer

The execution layer sends the XML request to the [XAI core server](#) and waits for a response.

---

**NOTE:** Currently the only type of [executer](#) supported is the XAI servlet running either on an XAI server or locally under the MPL. However the architecture allows for executing a request on other execution environments.

---

The executer is invoked and is passed in an [XAI Inbound Service](#) that specifies an XML request schema and an [adapter](#). Adapters tell XAI what to do with a request. The adapters point to a specific Java class that renders a service.

For example you can configure an Adapter to invoke any published application object (by pointing it to the appropriate java class). This adapter accesses system objects through the page service. You can think of an adapter as a plug-in component.

Once the executer processes the request and a response is received, it is transferred to the next layer, the [response](#) layer.

## The Response Layer

The response layer is responsible for "responding" to the execution. The responses are handled by invoking an appropriate [sender](#) defined on the receiver's response information. Each sender defined in the system knows how to process its response. For example:

- The JMS queue sender and the JMS topic sender post responses to the appropriate queue / topic.
- The [staging control sender](#) handles errors received during the execution of the staging control request.
- The [upload staging sender](#) updates the status of the upload staging record based on the success or failure of the staging upload request.
- The download staging sender is a bit unusual because it is helping to build the message being sent (Oracle Customer Care and Billing only).

---

**NOTE:** There are some cases where a response is not applicable. For example, the file scan receiver creates a staging control record to process a file that exists in a directory. There is no "response" applicable for executing this request.

---

## The XAI Client Component

The XAI Client component is the set of online control tables and tools used to manage your XAI environment.

The **Schema Editor** is a tool used to create and maintain XAI schemas.

---

**FASTPATH:** Refer to [XAI Schema](#) for more information.

---

The **Registry** is a term used to refer to all the tables required to "register" a service in the system. It includes the [XAI Inbound Service](#) and a set of control tables defining various options.

The **Trace Viewer** is installed with your XAI client tools and is used to view traces created on the XAI server.

## XML Background Topics

The following section introduces some background information related to XML.

## XAI Schemas

---

**NOTE: Business Adapter.** The **BusinessAdapter** adapter supports communication to configuration tool objects: [business objects](#), [business services](#) and [service scripts](#) through their own schema API. When communicating to these objects, it is not necessary to create XAI schemas for the schemas associated with the objects. As a result, there is no need to use the XAI schema editor when defining [XAI Inbound Services](#) for this adapter.

---

At the core of XAI are XML web services based on XML schemas. XML schemas define the structure of XML documents and are the infrastructure of e-business solutions. They are used to bridge business applications and enable transaction automation for e-commerce applications. Industry standard schemas document common vocabularies and grammars, enhancing collaboration and standardization. Validating XML processors utilize XML schemas to ensure that the right information is made available to the right user or application.

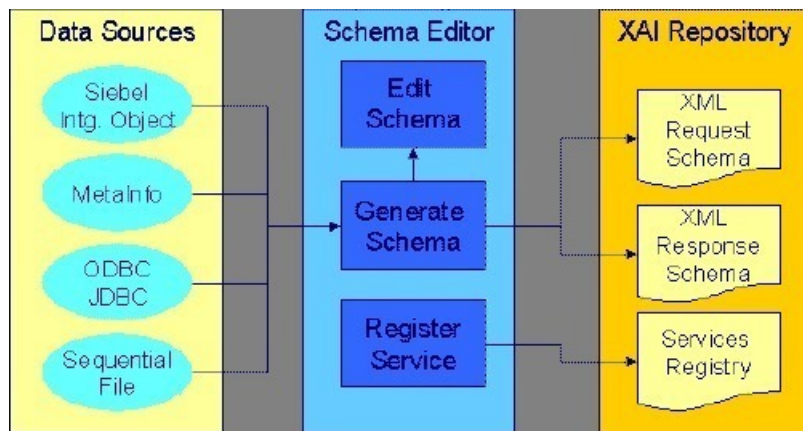
The system exposes its application objects as XML schemas that define the interface to system services. Every service (e.g., CreatePerson or AccountFinancialHistory) is defined using a pair of schemas: the Request Schema and the Response Schema. The request and response schema can be identical.

The Request Schema defines the XML document structure describing the "inputs" for a service.

The Response Schema defines the XML document structure describing the "outputs" of a service.

## The Schema Editor

To facilitate the process of exposing business objects as XML schemas, we use the Schema Editor, a graphical tool to create, import and maintain schemas. The Schema Editor provides automated wizards to import schemas residing in existing data structures and documents. The Schema Editor can import schemas from the following sources: system business objects, ODBC data sources, sequential files.



Before the XAI tool can use a service, it must be registered or published.

---

**FASTPATH:** Refer to [Schema Editor](#) for more information.

---

## XSL Transformations

XSL Transformations (XSLT) is a language used to transform an XML document into another XML document or another document format such as HTML. It is part of the Extensible Stylesheet Language (XSL) family of recommendations for defining XML document transformation and presentation. It is defined by the World Wide Web Consortium (W3C) and is widely accepted as the standard for XML transformations. Several tools are available on the market to generate XSLT scripts to transform an XML document defined by a source schema to an XML document defined by a target schema.

In XAI you can use XSL to:

- Transform an inbound message into the structure required by the XAI request schema for that service.
- Transform the response to an inbound message into a format defined by a schema provided by the requesting application.

---

**FASTPATH:** Refer to [XAI Inbound Service](#) for more information.

---

- Transform an outgoing message before it is sent out.

---

**FASTPATH:** Refer to [XAI Route Type](#) for more information.

---

- Transform data from an external source before it is loaded into the staging upload table.

---

**FASTPATH:** Refer to [XAI Inbound Service Staging](#) for more information.

---

## SOAP

SOAP stands for Simple Object Access Protocol. The SOAP "Envelope" is the wrapping element of the whole request document that may be used by messages going through the XAI tool.

The following diagram shows a simple XML request using the SOAP standard.

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <CorrelationID>1234</CorrelationID>
    <SOAPActionVersion>1.2</SOAPActionVersion>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <CISAccount transactionType='Read'>
      <CISAccountService>
        <CISAccountHeader
          AccountID='1234'
        />
      </CISAccountService>
    </CISAccount>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## XPATH

The XML Path Language (XPath) is an expression language used by XSLT to access or refer to parts of an XML document. It is part of the XSL recommendations maintained by the W3C. XPath provides the syntax for performing basic queries upon your XML document data. It works by utilizing the ability to work with XML documents as hierarchically structured data sets.

In the following XML document, some examples of XPath references are:

- authors/author/name
- authors/\*/name
- authors/author[nationality]/name
- authors/author[nationality='Russian']/name
- authors/author[@period="classical"]

```
<?xml version='1.0'?>
<authors>
  <author period="classical">
    <name>Sophocles</name>
    <nationality>Greek</nationality>
  </author>
  <author>
    <name>Leo Tolstoy</name>
    <nationality>Russian</nationality>
  </author>
  <author period="classical">
    <name>Plato</name>
    <nationality>Greek</nationality>
  </author>
</authors>
```

In the XAI tool, XPath is used to construct outgoing messages.

## Server Security

XAI server security supports the basic HTTP authentication mechanism as well as web service security (WS-Security) to authenticate the user requesting service. When authenticating using WS-Security, the SOAP header contains the authenticating information.

The base package provides two XAI server URLs, one that uses basic HTTP authentication ('/classicxai') and another that supports both methods ('/xaiserver'). Regardless of which authentication method is practiced, it is the latter you should expose as your main XAI server. The main XAI servlet gathers authentication information from the incoming request (HTTP or SOAP header) and further calls the internal ("classic") servlet for processing.

The "classic" XAI server security uses the basic HTTP authentication mechanism to authenticate the user requesting service. It assumes the requester has been authenticated on the Web server running the XAI servlet using the standard HTTP (or HTTPS) basic authentication mechanism. The authenticated user-id is passed to the application server, which is responsible for enforcing application security. This requires the system administrator to enable basic authentication for the Web server running the XAI servlet. To enable HTTP basic authentication, the XAI server '/classicxai' should be defined as a url-pattern in the web resource collection in the web.xml file. When the XAI server is not enabled for basic authentication, it transfers the user-id specified on the **Default UserMessage option** to the application server.

By default, the system would always attempt to further authenticate using SOAP header information. This is true even if the request has already been authenticated via the Web server. Use the **Enforce SOAP AuthenticationMessage Option** to override this behavior so that a request that has been authenticated already by the Web server does not require further authentication by the system.

If SOAP authentication information is not provided, the system attempts to authenticate this time using information on the HTTP header. You can force the system to solely use SOAP authentication using the **Attempt Classic AuthenticationMessage Option**.

Currently the system only supports the standard *Username Token Profile* SOAP authentication method where "Username", "Password" and "Encoding" information is used to authenticate the sender's credentials. The following is an example of a *Username Token Profile* in a SOAP header:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "urn:schemas-xmlsoap-org:envelope">
```



```

<SOAP-ENV:Header xmlns:wssse="http://www.w3.org/2001/XMLSchema-instance">
<wssse:Security>
<wssse:UsernameToken>
<wssse:Username>MYUSERID</wssse:Username>
<wssse:Password Type="PasswordText">MYPASSWORD</wssse:Password>
</wssse:UsernameToken>
</wssse:Security>
<SOAPActionVersion>2.0.0</SOAPActionVersion>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
...
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

By default both user and password are authenticated. You can use the **System Authentication Profile** [Message Option](#) to change this.

---

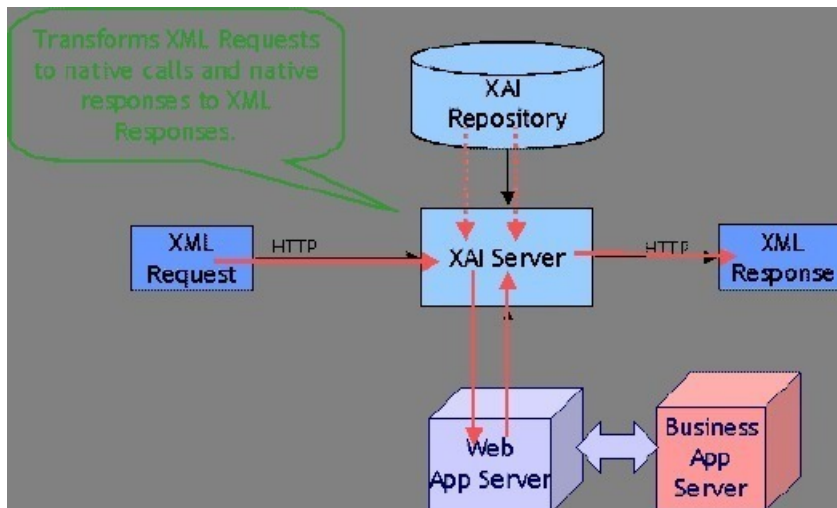
**NOTE: Custom authentication.** You can override the base package user credentials authentication logic using the **System Authentication Class** [Message Option](#).

---

## Inbound Messages

Inbound messages are XML messages sent by an external application to invoke a system service. Inbound messages can be sent one at a time or in batches of several messages in a file. Third party integration points can import web services for inbound service calls. The standard method of doing this is by publishing a WSDL (Web Service Definition Language) definition for each exposed service.

## Synchronous Messages



Synchronous service requests are sent using the HTTP protocol to the XAI servlet, which runs under a web application server such as WebLogic.

- The service request XML document must conform to the request schema that defines the service.
- The XAI servlet on the web server receives the service request XML document and based on the service name in the document identifies the appropriate XAI Inbound Service. Once the service is identified, the XAI servlet accesses the XAI Inbound Service record to find out the properties of the service.
- Based on the service properties, the XAI module loads the request and response schemas from the schemas repository (and caches them in memory).



- Based on the Adapter referenced by the service, it calls the appropriate adapter. The adapter performs most of the work to service the request.
  - The adapter connects to the application server, based on the connection information in the registry control tables.
  - It then parses the request document using the request schema.
  - Once the request document has been validated, the adapter converts the XML request document into a call to the application server.
- The response returned by the application server is then converted into a service response XML document, based on the response schema.
- The XML response document is shipped back to the caller over HTTP.

## Using HTTP for Synchronous Service Execution

Invoking a service is not much different from sending a regular HTTP request. Here the HTTP request contains the XML as a parameter. The XAI server handling requests via the HTTP protocol is implemented using a Java servlet running on the web server.

### Microsoft Visual Basic/C Example

Microsoft provides an easy way to send XML requests over HTTP. To send and receive XML data over HTTP, use the Microsoft XMLHTTP object

```
set xmlhttp = createObject("Microsoft.XMLHTTP")
xmlhttp.Open "POST", http://localhost:6000/xaiserver, false
xmlhttp.Send XMLRequest
XMLResponse = xmlhttp.ResponseText
```

Here **http://localhost:6000/xaiserver** is the URL of the XAI server. **XMLRequest** contains the XML request to be processed by XAI and **XMLResponse** contains the XML response document created by the XAI runtime.

### Java Example

Java provides a very simple way to send a request over HTTP. The following example shows how a request can be sent to XAI from an application running under the same WebLogic server as the one XAI runs. In this example, we use the "dom4j" interface to parse the XML document.

```
import com.splwg.xai.external.*;
import org.dom4j.*;
String xml;
xml = "<XML request>";
XAIHTTPCallForWebLogic httpCall = new XAIHTTPCallForWebLogic();
String httpResponse = httpCall.callXAIServer(xml);
Document document = DocumentHelper.parseText(httpResponse);
```

## Asynchronous Messages

Various types of [receivers](#) running under the MPL server (rather than the XAI server) handle asynchronous inbound messages from several sources.

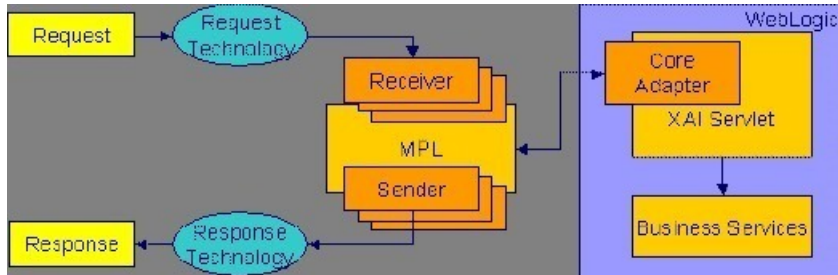
Requests may be received via the following:

- MQSeries, TIBCO or any JMS compatible messaging system
- The XAI Upload Staging table

The response is returned to the JMS queue/topic or to the staging table.

**FASTPATH:** Refer to [Designing XAI Receivers](#) for more information about the different receivers provided by the product for the different data sources.

The following diagram shows the flow for asynchronous inbound messages.

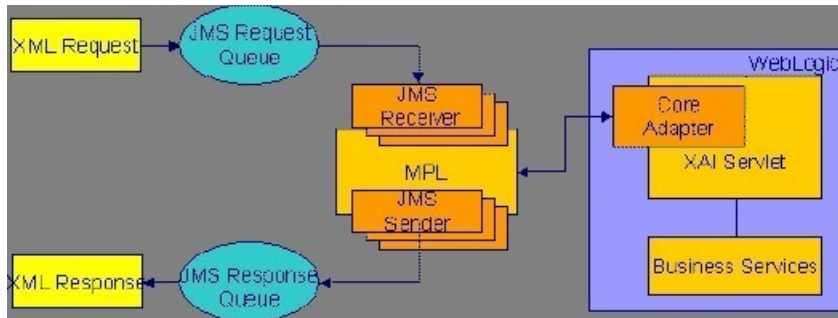


## Using JMS

Java Message Services (JMS) is a standard Java 2 Platform, Enterprise Edition (J2EE) protocol to send/receive asynchronous messages using a queue or topic message protocol.

XML messages may be received and sent via JMS using either a JMS Queue or JMS Topic. In order to access a JMS provider such as MQSeries, TIBCO or WebLogic, the MPL must first connect to the appropriate server using a JMS Connection.

The following diagram depicts a message sent and received through a JMS Queue.



## Using JMS Queues

JMS Queues are used to receive and send messages using the message queue protocol. Products that support this protocol include MQSeries.

The following describes events that take place when a JMS queue is used:

- The requester places the XML request message on a JMS queue. The request contains both the XML message and the name for the *reply queue*.
- The [JMS Queue Receiver](#) waits for messages on that queue. When the message arrives it is selected from the queue and executed using the adapter for the requested service.
- The XML response message is placed on the reply queue specified in the request. It is the requester's responsibility to fetch the response message from the queue.

The MPL uses a [JNDI server](#) to locate the queue resource.

---

**FASTPATH:** Refer to [Designing XAI JMS Queues](#) for more information.

---

## Using JMS Topics

JMS Topics are used to send and receive messages using the publish/subscribe messaging model. Products that support this protocol include TIBCO, MQSeries and WebLogic.

The MPL uses a [JNDI server](#) to locate the topic resource.

- Define a [JMS Topic receiver](#), listening to a predefined JMS Topic.
- The other application builds an XML message based on the schema defined for that service.
- It sends the XML request to the predefined topic and specifies the reply topic name.
- The MPL reads the message from the Topic, executes the service and returns the response to the reply topic specified in the inbound message.

---

**FASTPATH:** Refer to [Designing XAI JMS Topics](#) for more information.

---

## Staging Upload

The system provides a staging table, where an interface can store XML requests to perform a service in the system.

Some external systems interfacing with the system are not able to produce XML request messages. Or you may have external systems that produce XML messages but the messages are sent in a batch rather than real time. The system provides the capability to read an external data source containing multiple records, map the data to an XML request and store the request on the [XAI upload staging](#) table. These records may be in XML requests, sequential input files or database tables.

The XAI upload staging table may be populated in one of the following ways:

- When a collection of messages in a file or database table must be uploaded, a [staging control](#) record should be created for each file/database table. The [Staging Control Receiver](#) processes each file or database table and creates records in the XAI upload staging table for each message.
- The [XML File Receiver](#) creates records directly in the XAI upload staging table. Note, in addition, the XML File Receiver creates a staging control record to group together these records.
- XAI creates records in the XAI upload staging table for [inbound messages in error](#) that are configured to post the error.
- It is possible that when a response to a notification download staging message is received (Oracle Customer Care and Billing only), the response requires some sort of action. If this is the case, the system creates an XAI upload staging record (and an XAI staging control record) for the response.

## Staging Upload Receiver

Once the XML requests are in the staging table, the [Staging Upload Receiver](#) reads the requests from the XAI upload staging table and invokes the XAI server (via the executor) with the appropriate XAI inbound service. Inbound service records typically point to the core [adapter](#) used to invoke system services.

## Staging Upload Sender

The staging upload [sender](#) handles "responses" to the execution of the message in XAI upload staging. If the execution is successful, the sender updates the status of the upload staging record to **complete**. If the execution is unsuccessful, the sender updates the status to **error** and creates a record in the [XAI upload exception](#) table.

---

**NOTE: Configuration required.** The above explanation assumes that you have correctly configured your upload staging receiver to reference the upload staging sender. Refer to [Designing Responses for a Receiver](#) for more information.

---

## Staging Control

The [staging control](#) table is used to indicate to XAI that there is a file or table with a collection of records to be uploaded. The special [Staging Control Receiver](#) periodically reads the staging control table to process new records.

The XAI staging control table may be populated in one of the following ways:

- To process a specific sequential input file, manually create a staging control record and indicate the location and name of the file and the XAI inbound service to use for processing. Use this mechanism to process ad-hoc uploads of files.
- If a file is received periodically, you may define a [File Scan Receiver](#), which periodically checks a given file directory for new files. The file scan receiver creates a new staging control record to process this file.
- The [XML File Receiver](#) processes a file containing a collection of XML messages to be uploaded. The XML file receiver creates a staging control record and creates records directly into the XML upload table. The staging control record is automatically set to a status of **Complete** and is used to group together the XML upload records. Refer to [Processing Staging Upload Records for a Staging Control](#) for more information.
- To upload records from a database, manually create a staging control record and indicate an appropriate XAI inbound service, which contains the information needed by the system to access the appropriate table. Use this mechanism to process ad-hoc uploads of files.

---

**NOTE:** To upload records from a database table, you must create a staging control record. There is no receiver that periodically looks for records in a database table.

---

- Whenever an [XAI upload staging](#) record is created, an XAI staging control record is created as well.

---

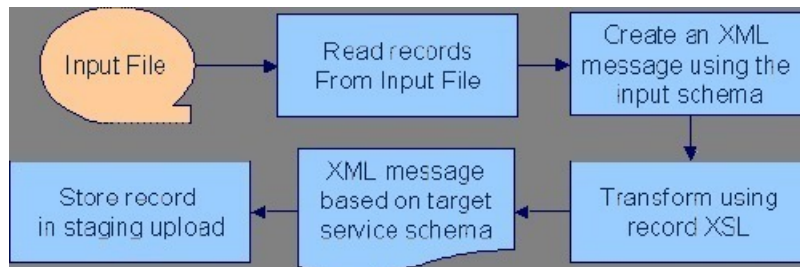
**FASTPATH:** Refer to [Batch scenarios](#) for more information about configuring the system to populate the staging control with requests from external files.

---

## Staging Control Receiver

The Staging Control [Receiver](#) processes staging control records and invokes XAI (via the executor) to execute the request. The executor uses the appropriate adapter to generate records in the [XAI Upload Staging](#) table - one for each record in the file or table.

The diagram below illustrates the information used by the staging control receiver to load data onto the staging table from a sequential input file.



The staging control [adapter](#) does the actual work. It reads the individual records in the input file and applies the XSL transformation script indicated on the XAI inbound service record to the input data to produce an XML request in the XAI upload staging table.

## Processing Staging Upload Records for a Staging Control

In some cases, a process may populate records directly into the XML staging upload table. An example of such a process is the [XML File Receiver](#). In this case, a staging control record is also created and used to group together the staging upload records.

The staging control contains information needed to process a group of staging upload records:

- The user related to these records. This user is for application security purposes. The user indicated here must have the proper rights for the application service and transaction type to be executed by XML upload records processed for this staging control.
- An indication of whether or not the records should be processed [sequentially](#).

## Processing Staging Records in Sequential Order

In some cases, a collection of messages uploaded together in a file must be processed in the order the messages are received. For example, if messages to add a person and add an account for this person are received together, the message to add the person must be processed before the message to add the account.

If messages received in a file must be processed sequentially, turn on the Sequential Execution switch on the [staging control](#) record. When the staging control receiver creates records in the XML upload table, the identifier of each record is built as a concatenation of the staging control record and a sequential number. If your staging control record indicates that the XML upload records should be processed in sequential order, the records are processed in primary ID order.

---

**NOTE: Non-sequential processing.** If your staging control does not indicate that the related XML records should be processed in sequential order, the records are processed by the staging control receiver using a random, multi-threaded mechanism.

---

If you have defined a [receiver](#) to periodically search for files and populate records in the staging control table, you may turn on the Sequential Execution switch on the receiver. This ensures that records processed as a result of this receiver are executed in sequential order.

## Staging Control Parameters

If your staging control accesses the data from a database table, you have the capability of defining the selection criteria. [XAI inbound services](#) that reference the **CISStagingUpload** adapter may contain a collection of fields that are used in an SQL WHERE clause. When adding a new [XAI Staging Control](#) record, you can define the values for the WHERE clause.

For example, imagine that you have a work management system where new premises are defined. Rather than waiting for this system to "push" new data to you, you design the interface to have the system "pull" the new data by looking directly at the "new property" database table.

The Request XML for your XAI service contains SQL statements used to access the data. You could define a Staging Control Parameter of "Add Date". When creating your staging control, you may enter a parameter value of today's date. When this record is processed, it only retrieves new properties from this work management table whose Add Date is today.

The following example shows part of the Request XML schema for an XAI Service that SELECTs premises based on postal code.

```

- <Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-com:datatypes" xmlns:d="urn:schemas-microsoft-com:datatypes" xmlns:b="urn:schemas-microsoft-com: BizTalkServer" b:root_reference="TestDuplicatePremise">
- <ElementType name="TestDuplicatePremise" content="eltOnly">
  <b:property name="adapter">CISStagingUpload</b:property>
  <b:property name="stagingUploadType">DB</b:property>
  <element type="Request" />
  <element type="Response" />
</ElementType>
- <ElementType name="Request" content="eltOnly">
  <AttributeType name="Postal" d:type="string" d:maxLength="12" />
  <b:RecordInfo />
  <attribute type="Postal" />
</ElementType>
- <ElementType name="Response" content="eltOnly">
  <b:RecordInfo />
  <element type="Premises" />
</ElementType>
- <ElementType name="Premises">
  <b:RecordInfo />
  <b:property name="SQL">SELECT PREM_TYPE_CD, KEY_SW, OK_TO_ENTER_SW,
TREND_AREA_CD, ADDRESS1, MAIL_ADDR_SW, POSTAL FROM CISADM.CI_PREM WHERE
POSTAL = @TestDuplicatePremise/Request/Postal</b:property>
  <b:property name="tableName">CISADM.CI_PREM</b:property>
  <AttributeType name="PREM_TYPE_CD" d:type="string" d:maxLength="000">
    <b:property name="dataType">string</b:property>
    <b:property name="uniqueId">PREM_TYPE_CD</b:property>
    <b:FieldInfo />
  </AttributeType>
  <AttributeType name="KEY_SW" d:type="string" d:maxLength="001">
    <b:property name="dataType">string</b:property>
    <b:property name="uniqueId">KEY_SW</b:property>

```

The postal code value to substitute into the WHERE clause is defined on the individual XML Staging Control records.

---

**FASTPATH:** Refer to [XAI Staging Control](#) for more information.

---

## Staging Control Sender

Before the staging control receiver invokes the executer, it changes its status to **complete**, assuming that there will be no problems. If the executer detects an error condition, the staging control sender updates the status of the staging control record to **error** and removes any XAI upload staging records that may have been created.

---

**NOTE: Configuration required.** The above explanation assumes that you have correctly configured your staging control receiver to reference the staging control sender. Refer to [Designing Responses for a Receiver](#) for more information.

---

## Inbound Message Error Handling

For messages that are processed using the [staging upload](#) table, application errors that prevent XAI from successfully processing the message cause the staging record to be marked in error and highlighted via a To Do entry.

For messages that are not processed via staging upload, your implementation should consider what should happen to application errors. If the origin of the message is able to handle an immediate error returned by XAI, then no special configuration is needed. An example of this is an HTTP call to our system where the originator of the message is waiting for a real-time response.

Otherwise, for messages where errors should not be returned to the originator, but should be highlighted in this system for resolution, be sure to mark the **Post Error** switch on the [XAI inbound service](#). When this switch is turned on and an application error is received by XAI when processing the message an [XAI upload staging](#) record is created (along with a staging control record) and marked in **error**.

For example, if a message is received via a JMS queue, application errors that prevent XAI from processing the message should not be returned to the queue because there is no logic to route the error to the sending system.



## Integration Scenarios

### Integration Using an EAI (or Hub)

It is possible for your various systems to be integrated with each other using a hub. The hub is implemented using an Enterprise Application Integration (EAI) tool provided by a third party vendor. Most hubs support HTTP and/or JMS and can work with XML schemas or document type definitions (DTDs).

- XAI services are presented to the EAI tool as schemas or as DTDs, immediately making the system callable from the hub.
- Integration scripts or workflow processes are defined in the EAI tools.
- At run time the hub uses HTTP or JMS to access the system using inbound messages.
- Outgoing messages are used to notify the hub about events occurring in the system. The messages are sent using HTTP or JMS.

### Batch Scenarios

Messages may be sent in batch files, or may be retrieved from a database. In all cases, the system needs to be able to read the file and identify each individual message in order to create an XML request that can be processed by the XAI server. Once each individual message is identified, a request is stored on the XAI Staging Upload table for later execution.

### XML Message File

It is possible for you to receive a file containing a collection of XML messages. The system identifies each separate message within the file and creates an entry for each message on the XAI upload staging table. It also creates a staging control record to group together each newly created XAI upload record. This staging control is created in **Complete** status and is not processed by the staging control receiver.

Since external applications may send messages in a format unknown to XAI, the system needs a mechanism for identifying the messages and mapping them to an XAI service.

#### XAI Groups

First the system associates the entire XML file with an XAI Group. You can think of the XAI Group as a categorization of the collection of messages. For example, you may have a separate XAI Group for each third party who sends you a collection of XML messages.

The system uses an [XPath](#) and XPath value to identify the correct XAI Group for the XML file.

#### Attachments and Rules

After identifying the appropriate group to which an XML file belongs, the system takes each message in the file and applies the appropriate XSL transformation to the message to produce a record on the upload staging table.

To process the messages in a file, the system needs to know how to identify each message in a file containing multiple messages. A file may use the same root element for each message or different root elements for different types of messages. For each XAI group, you must indicate the root element(s) that identifies a message by defining one or more attachments. Each attachment defines a root element, which tells the system when a new message begins.

Once the system has identified each separate message in the file, it must determine the correct XSL transformation script to apply. Once again the system uses an [XPath](#) and XPath value to identify the correct XSL to apply. For each XAI group, you define one XAI rule for every possible type of message you may receive in the file. Each XAI rule defines an XPath, XPath value and XSL transformation script.

Note you may assign a priority to each of your rules. The rules for more common messages may be assigned a higher priority. This enhances performance by ensuring that rules for more common messages are processed before rules for less common messages.

---

**NOTE: Include parent elements.** If the XML message includes parent elements (such as a transmission id or a date/time) that are needed for any of the separate child messages that are posted to the upload staging table, you can configure the appropriate attachment to include **parent** elements.

---

**FASTPATH:** Refer to [Designing an XML File Receiver](#) for more information about defining receivers that process XML files.

---

## Sequential Input File

You may receive messages in a sequential input file, such as a comma-delimited file.

The following steps should be performed when configuring the system to enable data to be uploaded from an input file into the staging upload table:

1. Create an XML Schema that describes the records in the sequential input file.
2. Create the XSLT transformation that maps a record in the input file to an XML service request in your product.
3. Create an [XAI service](#) representing the batch process that loads the input file into the staging table.
4. If desired, create a new file scan receiver, which waits for an input file to appear in a particular directory. (If you do not take this step, then you need to create a [staging control](#) when you want a file to be processed.)

---

**NOTE: Character Encoding.** If the file is encoded with a specific character encoding, you may indicate the encoding as part of the file name to be uploaded. If the file name ends with **?enc=?x**, where x is the file character encoding, the adapter processes the file accordingly. For example, the file name may be specified as **premiseUpload.csv?enc=?UTF-8**. If the encoding is not specified as part of the file name and the file is in UTF-16 or UTF-8 with byte order mark, then the adapter can recognize the encoding.

---

## Database File

It is possible for you to define an interface where inbound messages are retrieved by reading records in a database table.

The following steps should be performed when configuring the system to enable data to be uploaded from a database table into the staging upload table:

- Create an XML Schema that describes the records in the database table.
- Create the XSLT transformation that maps a record in the database table to an XML service request in your product.
- Create an [XAI service](#) representing the process that loads the records from the database table into the staging table.

## WSDL Catalog

Web Service Definition Language (WSDL) is a language for describing how to interface with XML-based services. It acts as a "user's manual" for Web services, defining how service providers and requesters communicate with each other about Web services.

The base package provides the ability to publish a WSDL definition for each service exposed as an [XAI Inbound Service](#). In addition, it is possible to request a catalog of all the XAI Inbound Services and a link to each WSDL. To view the catalog, launch a new browser session and enter the URL **http://\$host:\$port/XAIApp/xaiserver?WSDL** where \$host and \$port are replaced by the appropriate values for the current environment.



## Outgoing Messages

This section describes outgoing message functionality related to MPL, which is no longer recommended.

- Outbound messages. As described in [Outbound Messages](#), outbound messages are supported for sending outgoing messages. This functionality includes support that is only related to MPL.
- Notification download staging (NDS) messages. This method is only supported by *Oracle Utilities Customer Care and Billing*. Using this method near real-time, a record is written to the NDS staging table referencing only key fields. MPL then polls the records, invokes a service to build the message, applies the XSL and routes the message. If sent real-time, no record is posted to the staging table but rather the message dispatcher routes the message immediately. Refer to Oracle Utilities Customer Care and Billing help, Workflow and Notifications, Notification and XAI for more information.

The following sections describe the outbound messages topics specific to MPL in more detail.

## Outbound Message Receiver

The outbound message [receiver](#) processes records in the outbound message table that have a processing method flag equal to **XAI** and a status of **pending** and changes the status to **in progress**. The receiver then retrieves the message XSL and the message sender defined for the external system / outbound message type.

---

**NOTE: Template external system.** If the outbound message's external system references a template external system, the outbound message type configuration for the template external system is used.

---

It applies the message XSL (if supplied). If the option to [validate outbound message schemas](#) is turned on, the schema validation is performed.

Refer to [Outbound Message Error Handling](#) for information about error handling.

If no errors are received, control is turned over to the [outbound message sender](#) for routing.

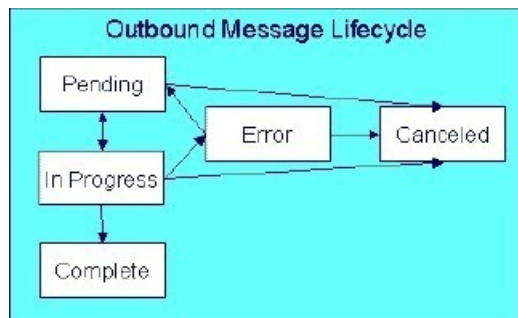
---

**NOTE: Initialization of receiver.** During the initialization of this receiver (for example if there is a problem with MPL and it is restarted) any records that are found to be **in progress** are changed to **pending** so that those messages are sent properly.

---

## Lifecycle of Outbound Message

The outbound message receiver processes outbound message records based on their status. The following diagram describes the lifecycle of an **XAI** type outbound message.



- Records are created in **pending** status.
- The outbound message receiver processes pending records and changes the status to **in progress**.

- If the message is sent successfully the system changes the status to **complete**.
- If there was a problem sending the message the system changes the status to **error**.
- When the user resolves the error they can change the status back to **pending**.
- A user can change the status of a **pending** or **error** record to **canceled**.
- For the rare cases where there is a problem with MPL and a message is left in the status **in progress**, users may manually change the status to **canceled**. In addition the outbound message receiver includes a step at startup to find **in progress** messages and change them to **pending**.

## Outbound Message Sender

The outbound message sender is responsible for routing the message to the [Message sender](#) determined by the receiver. If the routing is successful the outbound message status is marked **complete**. If the routing is unsuccessful, the status is marked in **error**.

Refer to [Outbound Message Error Handling](#) for information about error handling.

---

**NOTE: Automatic Resend.** If you have configured the system for [automatic resend](#) and the system detects that the error is due to the sender being unavailable, the message remains in **pending** status.

---

**NOTE: Configuration required.** The above explanation assumes that you have correctly configured your outbound message receiver to reference the outbound message sender. Refer to [Designing Responses for a Receiver](#) for more information.

---

## Outbound Message Error Handling

If the outbound message receiver or the outbound message sender detects an error while attempting to process the outbound message, it marks the message in **error**, captures the error message and its parameter values and creates a To Do entry using the To Do type specified in the Message option **To Do Type for Outbound Message Errors**.

A separate background process [F1-DTDOM](#) is responsible for completing To Do entries for outbound messages no longer in **Error**.

## Outbound Message Schema Validation

The outbound messages that are generated by the system should be well formed and valid so that they do not cause any issues in the external system. To ensure this validity you may configure the system to validate messages before they are routed to their destination.

- Define a directory where the valid W3C schemas are located using the Message option **Outbound Message Schema Location**
- Each [external system](#) message must indicate the appropriate W3C schema to validate against

You may turn on or off this validation checking using an Message option **Schema Validation Flag**.

## Automatic Resend

If a system error is received by the MPL when attempting to route the message to a sender, (using the outbound message method or the NDS message method), the system marks the appropriate table in **error**. This is true even if the reason for the error is that the connection to the sender is unavailable. When the connection is restored, a user must change the status of the appropriate record to **pending** (for outbound messages) or **retry** (for NDS messages) in order for the message to be resent.

Alternatively, you can configure your system to attempt to automatically resend the message. This section describes the logic available for auto resend. To enable automatic resend, you must set the flag **Automatically Attempt Resending to Unavailable Senders** on [Message option](#) appropriately.

If an error is received by the MPL when it attempts to invoke a sender and the auto resend option is on, the system does not mark the record in **error**. It continues to attempt sending messages to the sender until the number of errors has reached a predefined maximum error number (defined as an [Message option](#)). When the maximum is reached, the sender is marked as **unavailable** and an MPL log entry is created. The MPL ignores messages for **unavailable** senders.

The system tries to resend messages to this sender the moment the sender is reset to be **available**. The following points describe how a sender becomes **available**:

- MPL attempts to retry sending messages to unavailable senders every X minutes, where X is defined on [Message option](#).
- On MPL startup all senders are marked as **available**
- A user may navigate to the [XAI Command](#) page and issue a command **MPL Refresh Sender** to refresh the cached information for a particular sender

## Designing Your XAI Environment

This section guides you through the steps required to design the tables that control your XAI processing.

### Installation

The XAI server is installed with default configuration. This section describes how you may customize the XAI server configuration.

Startup parameters are defined in two parameters files

- The XAIParameterInfo.xml file is used by the XAI HTTP server. This file is found within the XAI directory in the path (...\\splapp\\xai).
- The MPLParameterInfo.xml file is used by the MPL server. This file is found within the XAI directory in the path (...\\splapp\\mpl).

Both files store the parameters as XML files with the following elements (sections):

- Source
- ParameterVariables
- AdHocParameters

### The XAI Source Section

The XAI tool accesses XAI [registry](#) information through the standard system programs. The `<Source>` section in the XAIParameterInfo.xml file tells XAI the user ID for accessing the registry information. It contains the following attributes:

Attribute Name	Description
Source Type	This should be set to <b>CorDaptix</b> , which tells XAI to access the registry through the standard access to system programs.
CorDaptixUser	The user ID to use when accessing the registry data.

## The MPL Source Section

The <Source> section in the MPLParameterInfo.xml file defines the database connection information used to connect to the database storing the XAI table information. It contains the following attributes:

Attribute Name	Description
Source Type	Defines the source of the data, for example <b>ORACLE</b> or <b>DB2</b> .
jdbcURL	The URL used to connect to the product database. For example: <b>jdbc:oracle:thin:@//server-name:1234/DBNAME</b>
databaseUser	The Oracle User Id used to connect to the database. For example: sysuser
databaseUserPassword	The Oracle password used to connect to the database. For example: sysuserpassword

## The Parameter Variables Section

When defining values for fields in certain control tables in the registry, you may reference substitution variables that point to the <ParameterVariables> section of the installation files. Substitution variables provide for dynamic substitution of values based on parameters provided at server startup.

To specify a substitution parameter in a string value you enter the name of the substitution parameter enclosed with @.

For example if you have a field in the XAI control tables that should contain the URL for the XAI HTTP servlet, you could enter the value in the following way: **http://@HOST@:@PORT@/xaiserver**.

In the parameters section, define the appropriate values for these parameters, for example:

```
<ParameterVariables>
<ParameterVariable name="HOST" value="localhost" />
<ParameterVariable name="PORT" value="8001" />
</ParameterVariables>
```

At run time, the system builds the URL as `http://localhost:8001/xaiserver`.

Every substitution parameter is defined using an <ParameterVariable> element with the following attributes:

Attribute Name	Description
Name	The name of the substitution parameter
Value	The value to replace an occurrence of the substitution parameter in an XAI control table field

**NOTE:** Substitution parameters can only be used for string fields, and not for fields that are foreign keys to other objects.

## The AdHoc Parameters Section

The <AdHocParameters> section is used to provide registry definitions that override the existing ones. Unlike the <ParameterVariables> section, a whole registry object definition can be specified in this section. When the XAI server starts, it first reads the registry definitions from the database and then it reads the <AdhocParameters> section. If it finds an object definition in this section, it uses it to replace the one read from the database.

Attribute Name	Description
Object Name	The object name may be one of the following objects: Option Receiver

	Sender
Object Attributes	The attributes of the object. Each object type has its own set of attributes:
'Option' object attributes	
name	The option flag. Must be defined in the OPTION_FLG table
value	The value for that option
'Receiver' object attributes	
name	The receiver ID
Class	The JMS provider. May be 'MQ'
TargetClient	The client type writing/reading to the JMS queue/topic. May be 'JMS' or 'MQ'. Only relevant for interfacing with MQSeries
JMSProvider	The JMS provider. May be 'MQ'
TargetClient	The client type writing/reading to the JMS queue/topic. May be 'JMS' or 'MQ'. Only relevant for interfacing with MQSeries
Executer	The XAI Executer ID for this receiver
'Sender' object attributes	
Class	The JMS provider. May be 'MQ'
JMSProvider	The JMS provider. May be 'MQ'
TargetClient	The client type writing/reading to the JMS queue/topic. May be 'JMS' or 'MQ'. Only relevant for interfacing with MQSeries

## Designing XAI Inbound Services

When designing your XAI environment, you should first identify the services that you would like to perform. Determining your services facilitates your design for the other registry options.

To design your inbound services,

- Determine each service that needs to be performed
- Determine the correct [adapter](#) that is needed by your service.
- Determine the required layout of the request and response messages and specify the request [schema](#) and response [schema](#).
- If a transformation of the data is required, you need to design the appropriate request XSL and response [XSL transformation](#) scripts.
- If the service references the staging upload adapter, determine the staging file type and design the record XSL transformation script. In addition, determine if you want to enter an input file name and interface name. Finally, determine whether or not you need to indicate a special [JDBC connection](#).
- As new releases of the system are installed, it may be necessary to modify your service for the new release. If this is the case, you need to design separate versions of the inbound service.

---

**FASTPATH:** Refer to [XAI Inbound Services](#) for more information.

---

## Designing XML Schemas

You need XML schemas for the services you designed in [Designing XAI Inbound Services](#).

For each message, identify what service you need to invoke and what action you need to perform. If you have multiple actions that you may need to perform for the same service, you may choose to create a single generic XML schema or you may choose to create multiple schemas, which are more specific. For generic messages, the transaction type, indicating the action to perform would be passed in on the XML request document to indicate what must be done. For more specific

messages, you may be able to indicate the transaction type directly on the schema and it would not need to be overwritten at run time.

You need to create a response schema for each request schema. It is possible for you to use the same schema for both functions.

---

**FASTPATH:** Refer to [Schema Editor](#) for more information.

---

## Designing XSL Transformations

You need an XSL transformation script for each service you designed in [Designing XAI Inbound Services](#), where you determined a transformation is necessary. In addition, you need XSLT scripts for your outgoing messages. Each sender, which receives a message, probably requires a transformation of the message into a local format. Refer to [Outgoing Messages](#) for more information.

For each message requiring transformation, determine the format used by the external system. In most cases, it is not the same format recognized by the system. For each case, you must create an XSL transformation, which maps the message format from the external format to one expected by your product or from your product format to one expected by the external system.

When identifying the required XSL transformations, remember to take into consideration the data that is processed by the staging control table. This service reads data stored in a file or database table and uses the Record XSL to map the individual records to an individual service request.

---

**FASTPATH:** Refer to [XAI Inbound Service](#) for more information.

---

## Designing Your Registry Options

The XAI registry is a set of control tables that is used to store service definitions as well as various system information required by the XAI and MPL servers. The following sections describe each table in the registry.

### Designing XAI JNDI Servers

The XAI tool, including receivers and senders, uses a Java Name Directory Interface (JNDI) server to locate resources on the network. JDBC connections, JMS connections, JMS queues and JMS topics should be defined on the JNDI server. In addition, adapters that need to access your product information reference the JNDI server to determine where your product is running.

Design your JNDI server values as follows:

- Define a JNDI server that indicates where the system is running.
- If you are using JMS resources, such as JMS Queue or JMS Topic, and these are located in a server other than the one where the system is running, define the server where these resources are located.
- If there are any other resources that may be needed by XAI, define a JNDI server to indicate where these are located.

The product is shipped with a JNDI server running under the same Weblogic server used for the system.

When defining the URL, you may use substitution parameters such as those shown in the example below. XAI uses the parameter variables section of your start up parameters to build the appropriate URL. Refer to [Installation](#) for more information

---

JNDI Server	Description	Provider URL
WLJNDI	Weblogic JNDI Server	t3://@WLHOST@:@WLPORT@

---

## Designing XAI JDBC Connections

If you need to access a database table to process your messages, XAI needs to know the location of the database and how to access it. If the tables are located in the same database used for the system (defined in your [Installation](#)), then you do not need to enter any extra JDBC Connections. If you need to access data that lives in another database, design the additional JDBC Connections and determine the type of connection and connection information.

---

**FASTPATH:** Refer to [XAI JDBC Connections](#) for more information about defining XAI JDBC Connections.

---

## Designing XAI JMS Connections

If you are using JMS to send and receive messages, then you must define a JMS Connection to indicate the JNDI server to use to locate these resources. For each JMS connection defined, the [MPL](#) server creates a pool of connections that are later shared by multiple threads.

---

**FASTPATH:** Refer to [XAI JMS Connections](#) for more information about defining XAI JMS Connections.

---

## Designing XAI JMS Queues

If your business uses JMS Queues to send and receive messages, then you need to add an entry on the [XAI JMS Queue](#) page, defining the JNDI server and Queue Name.

## Designing XAI JMS Topics

If your business uses JMS Topics to send and receive messages, then you need to add an entry on the XAI JMS Topic page, defining the JNDI server and Topic Name.

## Designing XAI Formats

The Formats section of the registry is used to define data formats. Data formats can be used in schema definitions to specify data transformations. To determine what data formats you need to define for your XAI environment, you must review the expected format of data that you will be exchanging and determine whether or not data transformation is required.

The following sections describe the four different types of formats and some guidelines in their use.

### Date Formats

Date formats may be specified using any valid Java format supported by the `java.text.SimpleDateFormat` class.

To specify the time format use a time pattern string. For patterns, all ASCII letters are reserved. The following usage is defined:

Symbol	Meaning	Presentation	Example
G	era designator	Text	AD
y	year	Number	1996
M	month in year	Text & Number	July & 07
d	day in month	Number	10
h	hour in am/pm (1–12)	Number	12
H	hour in day (0–23)	Number	0
m	minute in hour	Number	30

s	second in minute	Number	55
S	millisecond	Number	978
E	day in week	Text	Tuesday
D	day in year	Number	189
F	day of week in month	Number	2 (2 <sup>nd</sup> Wed in July)
w	week in year	Number	27
W	week in month	Number	2
a	am/pm marker	Text	PM
k	hour in day (1~24)	Number	24
K	hour in am/pm (0~11)	Number	0
z	time zone	Text	Pacific Standard Time
'	escape for text	Delimiter	
"	single quote	Literal	'

## Currency Formats

Currency formats are used to specify formatting for elements representing currencies. They may include the following:

Symbol	Meaning
#	number place holder
,	thousands separator
.	decimal point
\$	currency sign

For example to define the currency format for US dollar, indicate: \$#,#.00

## Phone Formats

Phone formats can be used to specify formats for telephone numbers. The supported format specification is limited to the following format characters:

Symbol	Meaning
0	number place holder
\0	0

Any other character appearing in the formatting expression is a placeholder for that character. To specify the '0' character, use '\0'.

**Phone Format Example:** (000) 000-0000

## Text Formats

Text formats are used to specify formats for character string attributes. The following expressions are supported:

Symbol	Meaning
\cUpperCase	Translate the string to upper case.
\cLowerCase	Translate the string to lower case.
\cProperCase	Translate the string to proper case. The first character of every word is translated to uppercase. The remaining characters are translated to lowercase.

**FASTPATH:** Refer to [XAI Format](#) to define your XAI Formats.



## Designing XAI Adapters

The product provides a set of adapters to process your XML requests. The adapters point to a specific Java class that renders a service. If you find that you need to use a protocol, which is not supported by the adapters provided, you will need to add a new Message Class (which points to a Java class) and a new XAI Adapter. It is recommended that your implementers contact customer support. The following adapter classes are provided.

- **BASEADA:** This is the core adapter class that provides access to any published system service. This adapter accesses system objects through the page server. Services with this adapter need to indicate the object (application service), which should be invoked.
- **BUSINESSADA:** This is the core adapter class that provides access to schema-based objects. This adapter accesses [business objects](#), [business services](#) and [service scripts](#) through their schema API. Services with this adapter need to indicate the schema of the object, which should be invoked. When communicating to these objects, it is not necessary to create XAI schemas for the schemas associated with the objects. XAI is able to directly communicate with these objects using their existing schema definitions. As a result, there is no need to use the XAI schema editor when defining XAI Inbound Services for this new adapter.
- **STGUPADA:** This staging upload adapter class is used when an extra step is required prior to using a service with the core adapter. For example, perhaps you need to read a file, which is not in XML format, and convert it to an XML format recognized by the system. Services with this type of adapter do not need to indicate an application service but must indicate information about the file to be converted. Refer to [XAI Staging Control](#) for more information.
- **XAICMNDADA:** This is an internal adapter. It is used to send commands to the XAI Server.
- **LDAPIMPRTADA:** This adapter is no longer supported .
- **SIEBELADA:** This adapter is no longer supported.

## Designing XAI Executors

The executor is responsible for executing messages received through a message receiver. The product provides an executor, which uses the XAI server; however the architecture allows for implementing additional execution classes. If you require a different executor and therefore a different execution class, it is recommended that your implementers contact customer support.

---

**FASTPATH:** Refer to [XAI Executor](#) for more information.

---

## Designing Message Senders

This section only describes message sender functionality that is only related to the XAI/MPL processing. Refer to [Define the Message Sender](#) for details about other types of senders that are supported independent of XAI/MPL.

Message senders are responsible for define outgoing message destinations and for " [responding](#)" to the XAI executor.

- For NDS messages, the sender to use is defined on the [XAI route type](#) for the notification download profile.
- For outbound messages, the sender to use is defined on the [external system](#) / outbound message type collection.
- For responding to the XAI executor, the sender to use is defined on the [receiver](#).

For each sender, you must reference an appropriate Message Class. The information in this section describes the sender classes that are provided with the system.

You must create senders to "respond" to the various staging table receivers in the system.

- Create a sender to be used for "responses" to messages processed by the staging control receiver. You should create one sender, which points to the Message Class **UPLDERRHNDLR**. Refer to [Staging Control Sender](#) for more information about this sender.

- Create a sender to be used for "responses" to messages processed by the upload staging receiver. You should create one sender, which points to the Message Class **STGSENDER**. Refer to [Staging Upload Sender](#) for more information about this sender.
- Create a sender to be used for messages processed by the download staging receiver. You should create one sender, which points to the Message Class **DWNSTGSNDR**. (DWNSTGSNDR is not supported in all products.)
- Create a sender to be used for messages processed by the outbound message receiver. You should create one sender, which points to the Message Class **OUTMSGSNDR**. Refer to [Outbound Message Sender](#) for more information about this sender.

Next, design the senders for "responses" to other receivers, for example the JMS queue receiver or JMS topic receiver. The system provides message classes to use for these senders. Use the class **JMSENDER** for a JMS queue sender and **TPCSNDR** for a JMS topic sender.

Finally, review all your [outgoing messages](#) and determine the mechanism for communicating with the target system for each message.

- For all senders that are used for [real time messages](#), define a context entry with a context type of **Response Time Out** to define the amount of time the system should wait for a real time response.
- An HTTP sender is one that sends messages to an HTTP server using the HTTP protocol. For an HTTP sender, reference a message class of **HTTPSNDR**. In addition, the context described in [Message Sender — Context](#) for the **RHTTTPSNDR** apply to this sender as well.
- An email sender allows for XML messages to be sent as email messages through an SMTP server. It can be used in notification download processes to send a response as an email message. The email sender supports standard email functionality such as "CCs" and attachments. The content of the email message is controlled by the XSL script defined in the [XAI route type](#) of the NDS message. The XSL script has access to all context records of the NDS message as well as the input XAI message that was created by processing the NDS. Reference a message class of **EMAILSENDER**. In addition, the context described in [Message Sender — Context](#) for the **RHTTTPSNDR** apply to this sender as well.
- If you want XML messages to be written to a flat file, use a flat file sender. For example, it can be used in the notification download process to write a response message to a flat file. Flat file senders should reference an Message Class of **FLATFILESNDR**. In addition, the following context records should be defined for senders of this type.

Context Type	Description	Values
Flat file output directory	Directory in the file system where to write the file	
Flat file filename pattern	The name of the output file. The file name may be a literal constant, or generated dynamically.  To create a dynamic filename use <file name>\$\$ID, where \$\$ID is replaced at run time by the ID of the NDS message that triggered the response message. If no file name is defined for the sender, the XAI server generates a file name with the following format 'XAI\$\$ID.xai'.	
Append data to file	This parameter controls whether the content of the response message is appended to an existing file, or a new file is created (possibly replacing an existing one).	<b>YES or NO</b>
Character Encoding	Indicates if the message should be sent with character encoding. The sender will write the content of the file with encoding specified in the context value. If no value is specified, the sender uses the default Java system encoding, which is determined according to the operating system locale.	<b>UTF-8 or UTF-16</b>

The topics below describe configuration required for senders that route a message via an HTTP sender, a flat file sender or an email sender.

## Designing XAI Groups

XAI groups are used by the system to process an XML file containing multiple messages to be uploaded into the system. One or more groups may be defined for an [XML file receiver](#).

---

**FASTPATH:** Refer to [XML Message File](#) for more information about how groups are used to process an XML file.

---

When setting up your XAI environment, identify the interfaces that require uploading an XML file containing multiple XML messages into the system through XAI.

First you need to categorize the XML files that you may receive. Define an XAI Group for each logical categorization. For example, you may want to define a separate XAI Group for each third party who may send you a collection of XML messages. Or, if all third party service providers send direct access messages in a standard format, you may want to define a single XAI Group for direct access messages.

For each group, you need to identify the root elements that indicate when a new message is starting. This collection of unique root elements for a group is called the attachments.

For each group, you must identify every possible message that may be sent. For every message, define an XAI Rule. The rule indicates the XSL transformation script to be executed along with the XPath and XPath value that the system uses to identify each message.

---

**FASTPATH:** Refer to [XAI Group](#) to define groups, their attachments and their rules.

---

## Designing XAI Receivers

Receivers define small pieces of code that wait for requests to be received through various sources. Each receiver references an Message Class where the small piece of code is defined. The following receiver classes are provided:

- **STGRCVR** The receiver that references this class polls the XAI upload staging table for new inbound requests.
- **STGCTLR**: The receiver that references this class polls the XAI staging control table for new upload processes.
- **DWNSTGRCVR**: The receiver that references this class polls the notification download staging table (NDS) for new messages. (Not available in all products).
- **OUTMSGRCVR**: The receiver that references this class polls the outbound message table for new messages.
- **JMSRCVR**: Receivers that reference this class receive requests through a message queue that supports the JMS Queue interface, such as IBM MQSeries.
- **TPCRCVR**: Receivers that reference this class receive requests through a publish/subscribe model, such as TIBCO, or any system supporting the JMS Topic interface.
- **FILESCANRCVR**: Receivers that reference this class poll a given directory for files with a given file name pattern.
- **XMLFILERCVR**: Receivers that reference this class poll a given directory for XML files with a given file name pattern.

Multiple receivers may be defined for these receiver classes. For example, the XML file receiver defines the scan directory. If you have multiple directories that contain files to be uploaded, define a receiver for each directory.

All types of receivers reference an Message Class and XAI Executer. If you require a new Message Class or Executer because you use a protocol that is not currently supported, it is recommended that your implementers contact customer support.

## Designing Responses for a Receiver

Once a request has been sent for execution to the XAI server (via the executer), [the response layer](#) processes the response. For some receivers, a response may not be applicable. For example, a file scan receiver reads flat files in a given directory and posts records to the XAI staging control table. Responses are not applicable for this type of receiver.

The response may be conditional on the outcome of the request and may be sent to more than one destination ( [sender](#)). To design your receiver responses, determine the conditions under which a response should be sent for each request processed by each receiver:

- Never send a response
- Send a response if the request was successful
- Send a response if the request was unsuccessful due to a system error
- Send a response if the request was unsuccessful due to an application error

Once you determine when to send a response, you must determine where to send the response. Responses for different conditions may be sent to different Message Senders or to the same Message Sender.

## Designing Receivers that Poll Staging Tables

The following receivers are needed to poll the various system staging tables.

- Create a [staging upload receiver](#) that references the Message Class **STGRCVR**. For responses, **All Events** should reference the [staging upload sender](#).
- Create a [staging control receiver](#) that references the Message Class **STGCTLR**. For responses, **All Events** should reference the [staging control sender](#).
- If your implementation uses the NDS message method of communicating outgoing messages, create a staging download receiver that references the Message Class **DWNSTGRCVR**. For responses, **All Events** should reference the download staging sender. NDS messaging is not supported in all products.
- If your implementation uses the [outbound message](#) method of communicating outgoing messages, create an [outbound message receiver](#) that references the Message Class **OUTMSGRCVR**. For responses, **All Events** should reference the [outbound message sender](#).

For all the above receivers, if you need to access multiple environments, simply create receivers for each JDBC connection. Note that you should not add more than one of each of the above receivers pointing to the same JDBC connection. To improve performance for a single JDBC connection you may [configure multiple MPL servers](#).

---

**NOTE: Sample Data for Initial Install.** When first installing the system, records for each of the above receivers are provided. Your implementation may use these records or remove them and create your own.

---

## Designing a JMS Queue Receiver

If you need to receive messages through a JMS compatible queue, you need to define a JMS Queue receiver. When designing a JMS Queue receiver you first need to design a [JMS Connection](#) and a [JMS Queue](#).

If you would like to post [responses](#) back to the JMS queue, you may create an [Message sender](#) to send the response to the JMS queue.

## Designing a JMS Topic Receiver

If you need to receive messages through a JMS Topic using the publish/subscribe model, you need to define a JMS Topic receiver, which receives messages published under a specific topic. When designing a JMS Topic receiver you first need to design a [JMS Connection](#) and a [JMS Topic](#).

If you would like to post [responses](#) through a JMS Topic using the publish/subscribe model, you may create an [Message sender](#) to send the response to the JMS topic.

## Designing a File Scan Receiver

The file scan receiver constantly looks in a given directory for files with a given pattern. When it finds a matching file, it creates a record in the [staging control](#) table to upload the contents of the file into the [upload staging](#) table.

When setting up your XAI environment, identify the interfaces that require uploading a file from a directory into the system through XAI. For each unique file, define a file scan receiver. For each receiver record, indicate the Scan Directory, where new files will be placed, the Scan File, which is the naming pattern to look for and the XAI Inbound Service to use for mapping the data into a system service.

In addition, if you want to specify a character encoding, the following Context record should be defined.

---

Context Type	Description	Values
Character Encoding	Indicates that the message is character encoded. When the receiver creates a staging control entry for a file, it will add ?enc=?x to the name of the file in the table where x is the value of this parameter. Refer to <a href="#">Sequential Input File</a> for more information.	<b>UTF-8 or UTF-16</b>

---

## Designing an XML File Receiver

The XML file receiver constantly looks in a given directory for XML files with a given pattern. When it finds a matching file, it goes through steps to identify each separate message in the file, determine the appropriate XSL transformation and create a record in the staging upload table.

---

**FASTPATH:** Refer to [XML Message File](#) for more information.

---

When setting up your XAI environment, identify the interfaces, which require uploading an XML file containing multiple XML messages into the system through XAI. For each unique file, define an XML file receiver. For each receiver record, indicate the Scan Directory, where new files will be placed, the Scan File, which is the naming pattern to look for and the collection of XAI groups. XAI groups are used by the system to identify each separate message in the file and to determine the appropriate XSL transformation for each message.

---

**FASTPATH:** Refer to [Designing XAI Groups](#) for more information.

---

## Configuring the System for NDS Messages

Refer to the documentation for your product to find out if NDS messaging is supported.

## Schema Editor

The Schema Editor is a Graphical User Interface (GUI) tool to create XML schemas. The tool provides wizards to generate schemas from various sources.

## Opening the Schema Editor

After launching the schema editor, you are asked to connect to a database. On the Connect dialog:

- Select an ODBC data source pointing to the desired database.
- Enter the data source, user ID, password and database owner required to log on to that database.
- Click **Connect**.

After connecting, the schema editor appears.

Use the File/Open dialogue to select a schema from the schema directory. Refer to [The Options Menu](#) for information about setting the default schema directory.

---

**NOTE:** When opening a schema, the schema editor validates the schema and any errors are displayed. Refer to [Validating a schema](#) for more information.

---

## Schema Editor Window

The schema editor allows you to modify individual elements and attributes of a given schema.

### Description of Page

Refer to [System Wide Functions for Schema Editor](#) for information about the various menu options available for the schema editor.

**Service Name** Enter the name of the service to be created in the service name text box. This is the name of the first element under the Body element in the XML document.

---

**CAUTION:** Important! When adding new schemas, carefully consider the naming convention for the Service Name. Refer to [System Data Naming Convention](#) for more information.

---

**Adapter** The adapter used to process services using this schema.

**Internal Service Name** If the schema is for an adapter that should invoke a system service, this is the internal name of the service.

**Transaction Type** Select the transaction type performed by the service. The available values are **Read, Add, Change, Update, Delete, List** and **Search**.

---

**NOTE:** The difference between Change and Update is that for Change, all field values must be passed in with the request. Field values that are not passed in to the request are set to null. For Update, you need only pass the primary key field values and the values of the fields to be updated. All other fields retain their existing values.

---

### Left Panel

The left panel of the schema editor displays a tree view of the hierarchical elements in the schema. The (+) expands a node, the (-) collapses a node.

### Right Panel

The following attributes appear on the right panel of the Schema Editor. Some fields cannot be modified in the schema editor. The field description indicates when the field is protected.

**Tag Name** The XML element tag name. This field is protected, but you may modify this attribute to give the element a self-explanatory name by right-clicking on the element name in the left tree-view.

**MetaInfo Name** Maps the element to a fully qualified field name in the service, for example PER\_ID. This field is protected.

**Internal Type** This property is populated automatically when you generate the schema from your product. The values further define elements and attributes. The values are **page**, **pageBody**, **list**, **listHeader**, **listBody**, **searchHeader**, **codeTableDesc**, **Private**. The values of **codeTableDesc** and **Private** are used to define special types of attributes.

**Private attribute** A field that does not exist on the server side, but one that you still want to have in the schema.

**Description** A description of this field.

**Content** The element type. This field is only available for elements. Possible values are **eltOnly**- element may contain only other elements and no text, **TextOnly**- element may only contain text.

**Search Type** Services, which perform a Search, may allow searching based on different criteria. The values are taken from the system meta information when the schema is generated. The possible values are **Main**, **Alternate1**, **Alternate2**, **Alternate3**, **Alternate4**, **Alternate5** and **Alternate6**.

---

**NOTE:** You would not typically modify this value because it corresponds to a value in the meta information. However, the value is modifiable to accommodate the rare case in which a service may change in a new release. In this scenario, you may prefer to update the schema manually rather than regenerate a new schema for the new version.

---

**Is Part of Primary Key** Used to indicate to the XAI server whether or not this field makes up part of the primary key of the record. The values are taken from the metadata information when the schema is generated. Value may be **true** or **false**.

---

**NOTE:** Typically you would not modify this value because it corresponds to a value in the meta data information. However, the value is modifiable to accommodate the rare case where a service may change in a new release. In this scenario, you may prefer to update the schema manually rather than regenerate a new schema for the new version.

---

**Min Occurs** This field is available for elements only and is used for repeating elements. It defines the minimum number of occurrences for an element. Value may be 0 or 1.

**Schema Max Occurs** This field is available for elements only and is used for repeating elements. It defines the maximum number of occurrences for an element. Value may be 0, 1 or \*.

**Limit Number of occurrences** This field is available for elements only and is used for repeating elements. If the **Schema Max Occurs** field has been set to '\*', define the number of max occurrences here.

**XML Data Type** The data type for the attribute. Possible values are **number**, **string**, **decimal**, **date**, **dateTime**, and **boolean**.

**Server Data Type** Indicates the data type of this attribute on the server. This field is protected.

**Service Format** The format expected by the service. At runtime, XAI converts the Tag format to the Service Format before executing the request. Formats are defined in [XAI Format](#).

**Tag Format** The format used to format an element/attribute in the schemas. Formats are defined in [XAI Format](#).

**Min Length** Use this property to define the minimum length of the attribute, if applicable.

**Max Length** Use this property to define the maximum length of the attribute, if applicable.

**Precision** This is used for decimal attributes to define the maximum number of digits.

**Scale** This field is used for decimal attributes to define the number of digits at the right of the decimal point.

**Required** A value of **Y** indicates that the element must appear in XML document. A value of **N** indicates that the element is optional.

**Default value** Default value to be used for Request schema, when the element is not supplied as part of the XML request document.

**Fixed Value** Fixed value to be used for Request schema. This value is used regardless of the value supplied in the request document.

**Code Table Field** This property is used for attributes that are descriptions of a code table, where the description is not automatically returned by the system service. Use this property to indicate the code whose description should be retrieved by the XAI server.



**Code Table Program** This property is used for attributes that are descriptions of a code table, where the description is not automatically returned by the system service. Use this field to indicate the program that XAI should call to access the description for the **Code Table Field**.

### Creating a Schema

Usually you do not create schemas from scratch; rather you use Schema Creation Wizards to import existing data structure definitions from a variety of data sources:

- System services
- Comma Delimited Files (CSV)
- Database Extract
- Any XML document

Once a schema is created based on the existing data structure, it is displayed in a TreeView on the left panel. Once the imported schema has been edited, it serves as the basis for creating the request and response schemas. When imported, the schema exposes all fields defined in the service. You may want to remove some attributes/elements from the request or response schema.

---

**NOTE:** Although the main purpose of the editing process in the creation of the request and response schemas is the elimination of elements, which makes the schema shorter and more understandable, it is not required for processing purposes. Therefore, if you don't mind that you have not used elements in your schemas, you could stay with one schema, which serves as both the request and response schema.

---

1. Save the Schema as a Request schema with an appropriate name, for example `PersonInfoRequestSchema.xml`
2. To create the Response schema, which is identical to the request schema, use the Save As Response menu option. This renames the top element of the schema to `ServiceNameResponse`, for example `PersonInfoResponse` and save the schema under a different name i.e. `PersonInfoResponseSchema.xml`. Note that if the request and response schemas are identical then one schema may be used for both and there is no need to create separate schemas.
3. Read in the Request Schema (File/Open) and modify its structure. Depending on the service type, you'll have to modify the contents of the Request Schema. This is usually required when the service is an "Info" service, which requires very few input elements. In such cases you'll delete most of the elements on the schema and only leave the necessary elements required to invoke the service. For example: in the `PersonInfo` request, you only need the `PersonId` and the `Company` elements in the request schema.
4. Read in the Response Schema (File/Open) and Modify its structure. Depending on the service type, you'll want to modify the contents of the Response Schema. This is usually required when the service is an "Add" or "Delete" service, which returns very few input elements. In such cases you'll delete most of the elements on the response schema and only leaves the necessary elements required by the requester of the service.

### Adding an Element/Attribute

Usually, you won't have to add element or attributes to a schema. However if the schema already exists and you want to add an element/attribute, you can follow this procedure. Be aware that any element/attribute added here must also exist on the xml metainfo.

- Select the element's node on the TreeView.
- Right click on it and select the 'Add Element' or 'Add Attribute' option in the pop-up menu
- Enter the element/attribute name in the prompt dialog box and click OK.

### Removing Elements/Attributes

When generating a schema using one of the wizards, the generated schema may contain information that you do not want to publish as part of the service, or is not required for a particular service. You can remove elements/attributes from the schema, and though these elements/attributes may still exist on the service they are not seen by the XAI service using this schema. To remove an element or attribute:



- Select the node to be removed.
- Right click and select "Delete" from the popup menu.

### Renaming an Element

To rename an element:


- Select the element's node on the TreeView.
- Right click it and select the Rename option in the pop-up menu
- Enter the new name in the prompt dialog box and click OK.

---

**NOTE:** The information in this table is cached by the XAI server and by the MPL server. Changes to values in this table do not affect the runtime copy of these servers. Refer to [XAI Command](#) for information about refreshing a schema.

---

## Validating a Schema

Although a schema is validated against the metainfo XML file when it is read into the editor or before it is saved, you can perform the validation at any time while the schema is being edited. To validate a schema, click on the toolbar "Validate" button . If the schema fails to validate the schema errors dialog is displayed.

## Schema Validation Errors

When the editor fails to validate a schema against the xml metainfo file, it pops up a dialog that lists the errors found in the schema definition. These errors may be of two types:

- An element or attribute in the schema could not be found in the xml metainfo file. You can click **Remove** to remove the element from the schema.
- The data type of an attribute does not match the one defined in the metainfo file. You can click **Correct**, and the editor fixes the data type so it does match.

The **Correct All** button can be used to correct all fields that have data types that do not match the one in the XML metainfo file.

---

**NOTE:** Correcting errors does not save the schema definition into a file. You have to save it manually.

---

If you **Exit** without correcting the errors, the schema displays with the mismatch information highlighted in red.

## Registering a Service

Before a service can be used it must be defined in the [XAI Inbound Service](#) table in the XAI registry. A service can be registered in the XAI registry directly from the schema editor. Go to the menu item 'Register Service' in the 'Schemas' menu. The Register service UI page appears. Fill in the required fields.

---

**NOTE:** The registry entry for the service can be later modified using the [XAI Inbound Service](#) page.

---

## Testing a Schema

The schema editor provides a testing option.

---

**NOTE: Testing in your product.** You may also test your schemas and your services using either [XAI Submission](#) or [XAI Dynamic Submission](#).

---

## System Wide Functions for Schema Editor

Because the schema editor is in an application outside of the standard products, this section introduces some general functions related to the application.

### Application Standards

- The schema editor is a Multiple Document Interface (MDI) windows application. It may contain multiple active windows. You may jump from one window to the other using the Window menu option.
- The Editor window is always active. Closing the schema editor window quits the application.
- In the Editor window, a splitter bar is available to resize the schema tree view horizontally.
- Actions may be performed using menu items or by clicking on toolbar buttons.
- All messages are displayed on a status bar at the bottom of the screen. Some may also be displayed using message boxes.

### The File Menu

Use the File menu to open existing schemas and to save a schema to a file.

Connect

Connects to the database.

Open - Loading an existing schema into the editor

You can read an existing schema into the editor.

1. Click on the Open toolbar button or select the File/Open menu option.
2. A file selection dialog is shown. Select the schema file name.
3. The editor first validates the schema against the xml metainfo file. If it fails to validate it shows the [Schema Validation Errors](#) dialog.

Save

Save the current schema to a file. Using the current file name.

To save a Schema, click on the Save toolbar button, or select the File/Save menu option. When you save a schema, the editor first attempts to validate the schema. If it fails to validate it against the XML Metainfo file, you are prompted to save it with inconsistency errors or to return to the editor.

Save As

Save the current schema to a file. Use a different file name.

Save As Response

Save a copy of the current schema to a file as a response schema. Use a different file name.

### The View Menu

Use the view menu to perform actions on the Tree View nodes or to view errors. The following menu options are available:

Expand All

Expand all nodes in the Tree View

Collapse All

Collapse all nodes in the Tree View

Expand Branch

Expand the selected node and all the node's children

Search (Ctrl+F)

Find a node with a node name containing a given string

Search Again (F3)

Find the next node with containing the search string

View Schema Errors

Display the Schema Validation Errors dialog.

Web Browser

Display the current schema definition on a web browser page

## The Schemas Menu

Use the Schemas menu to create, test, validate and register schemas.

- To create schemas from various sources use the **Create menu**.
- To [validate a schema](#) select the **Validate** option (Ctrl+V).
- To create a sample instance and test the schema, select the **Test** option (Ctrl+T).
- To create an entry in the service registry for a service represented by the current schema, select the **Register Service** option (Ctrl+R). Refer to [Registering a Service](#) for more information.

## The Export Menu

The options on this menu are no longer supported.

## The Options Menu

### Always Save As W3C

Turn on this option to save schemas in W3C format by default instead of XDR format.

### Always Save As DTD

Turn on this option to save schemas in DTD format by default instead of XDR format.

### Preferences

The following options may be set on the preferences dialog (Select Options and then Preferences):

- The Default Date Time Format - used for schema date fields
- The default Schema Directory - used to read/save schemas
- The default Metainfo Directory - used to create/validate schemas
- The XAI Servlet URL - used when executing requests from the test schema dialog.
- The Default Account Id - used when generating sample instances of a schema
- Language - used to access language specific data

## The Tools Menu

Schemas tools can be invoked from the **Schema Editor Tools** menu.

### Converting Schemas to a W3C compatible schema

Schemas generated in the Microsoft BizTalk-compatible format (XDR format) may be saved in a format compatible with the *October 2000, W3C XML* schema standard. To save a schema in a W3C format:

- Select **Convert to W3C** from the **Tools** menu. The **Convert to W3C** dialog box appears.
- Select the schema(s) to be converted. Multiple schemas may be converted in a single step.
- The name of the W3C schema is the same name as the schema but with an ".xsd" file extension, and is saved to the same directory as the original schema.
- Click **Convert**.

### Converting Schemas to a DTD

Schemas generated in the Microsoft BizTalk-compatible format (XDR format) may be saved as a DTD.

- Select the **Convert to DTD** option from the **Tools** menu. The **convert DTD** dialog box appears.
- Select the schema(s) to be converted. Note that multiple schemas may be converted in a single step.
- The name of the W3C schema is the same name as the schema but with a ".dtd" file extension, and is saved to the same directory as the original schema.
- Click **Convert**.

### Validating multiple schemas

The schema validation tool can be used to validate the correctness of an XAI schema when compared to the metainfo xml definition used to generate the schema. For each validated schema, the validation tool scans the list of elements/attributes and compares them with those defined in the XML metainfo file. Select **Validate Schemas** in the **Tools** menu.

- The **Validate Schema** dialog box appears.
- The left list box is used to select the directory where the schemas to be validated are stored. By default this is the **Schema Directory** as defined in the preferences.
- On the right, the list of schemas is displayed on a grid.
- Multiple schemas may be validated in a single step.
- To select a schema click on the left button (the first column in the row).
- To select multiple schemas, hold the **Ctrl** key and select the required schemas.
- Click **Validate Schemas**.
- The schema grid is updated. When an attribute defined in the schema is missing from the metainfo file or when the properties of the element do not match defined in the metainfo, a button is displayed at the right of the schema name. Clicking on the **edit** button loads the schema into the editor and displays the **Schema Validation Errors** dialog for that schema. You can correct the schema and save it.

## Setting Up Your XAI Environment

This section describes the control tables available to administer your XAI environment.

## Message Class

The Message Classes are references to actual Java classes. The Message Class defines the Java class used to implement Receivers, Senders, Adapters and Executors. This information is provided with the system and does not need to be modified for a specific installation.

To view an Message Class, open **Admin > XAI > Message Class**.

### Description of Page

The **Message Class** and **Description** are unique identifiers of the Message Class. The **Class Definition** indicates the Java class, implementing the adapter, receiver, sender or executor.

**Owner** indicates if this Message Class is owned by the base package or by your implementation ( **Customer Modification**). The system sets the owner to **Customer Modification** when you add an Message Class. This information is display-only.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_CLASS](#).

## XAI Envelope Handler

To view your envelope handlers, open **Admin > XAI > XAI Envelope Handler**. This information is provided with the system and does not need to be modified for a specific installation.

### Description of Page

Enter a unique **XAI Envelope Handler ID** and **Description**.

Indicate whether the **Envelope Type** is **Custom SOAP Envelope**, **Default** (no SOAP environment) or **SOAP Envelope**. Note that the values of **Siebel Integration Message** and **Siebel VBC** are not supported.

When the envelope type is **SOAP Envelope**, indicate the **Envelope URI**.

**Owner** indicates if this XAI envelope handler is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI envelope handler. This information is display-only.

## Setting Up Your Registry

The following section describes the control tables that are logically considered part of the XAI Registry.

### XAI JNDI Server

To define a new JNDI Server, open **Admin > XAI > XAI JNDI Server**.

#### Description of Page

Enter a unique **XAI JNDI Server** and **Description**.

Indicate the Provider URL, which is the URL of the [JNDI server](#).

Indicate the **Initial Context Factory**, which is a Java class name used by the [JNDI server](#) provider to create JNDI context objects.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_JNDI\\_SVR](#).

## XAI JDBC Connection

To view an XAI JDBC Connection, open **Admin > XAI > XAI JDBC Connection**.

### Description of Page

Enter a unique **XAI JDBC Connection** and **Description**.

Use the **Connection Type** to indicate how the JDBC connects to a database. The following connection types are valid:

- **Oracle Defined Connection** indicates the connection is to an Oracle database through a JNDI entry.
- **DB2 Defined Connection** indicates the connection is to a DB2 database through a JNDI entry.
- **JNDI Defined Connection** indicates the connection is using the MQ series classes implementing JMS.
- **Determined by parameter file** indicates that the connection information should be determined by looking at the parameters defined at [Installation](#).

For connection types of **Oracle** or **DB2**, use the **JDBC URL** to indicate URL of the database connection to be initialized at XAI/MPL startup time. Indicate the **Database User** and **Database Password** required for accessing the database. The JDBC connection URL can either be a Type 2 or a Type 4. For example:

- Type 2: jdbc:oracle:oci8:@CD200ODV
- Type 4: jdbc:oracle:thin:@myhost:1521/ CD200ODV

For a connection type of **Determined by parameter file**, indicate the parameter substitutions, which should be accessed from the parameter file for the JDBC URL, database user and database password, for example, @JDBCURL@, @DBUSER@ and @DBENCPASS@.

When the connection type is **JNDI**, indicate the **XAI JNDI Server** and the **JNDI Data Source** name as defined in the JNDI.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_JDBC\\_CON](#).

## XAI JMS Connection

To define a JMS Connection, open **Admin > XAI > XAI JMS Connection**.

### Description of Page

Enter a unique **XAI JMS Connection** and **Description**.

Indicate the **XAI JNDI Server** to be used. Refer to [XAI JNDI Server](#) for more information.

Use the **JNDI Connection Factory** to indicate the lookup keyword in the JNDI server used to locate the JMS connection.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_JMS\\_CON](#).

## XAI JMS Queue

To define your JMS Queue values, open **Admin > XAI > XAI JMS Queue**.

### Description of Page

Enter a unique **XAI JMS Queue** and **Description**.

Enter the **Queue Name** as defined in the JNDI server. This is the JNDI lookup name identifying the queue.

Use the **Target Client Flag** to indicate whether or not the target client is **JMS** or **MQ**.

Select the **XAI JNDI Server** where the queue is defined. Refer to [XAI JNDI Server](#) for more information.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_JMS\\_Q](#).

## XAI JMS Topic

To define your JMS Topic values, open **Admin > XAI > XAI JMS Topic**.

#### Description of Page

Enter a unique **XAI JMS Topic** and **Description**.

Select the **XAI JNDI Server** where the topic is defined. Refer to [XAI JNDI Server](#) for more information.

Enter the **Topic Name** as defined in the JNDI server. This is the JNDI lookup name identifying the topic.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_JMS\\_TPC](#).

## XAI Format

Open **Admin > XAI > XAI Format** to define the various formats.

#### Description of Page

For each new format, specify a unique **XAI Format** name and **Description**.

Indicate whether the Format Type is a **Currency formatting string**, a **Date/Time formatting string**, a **Phone formatting string** or a **Text formatting string**.

Finally, indicate the **Format Expression**, which defines the formatting pattern to be applied.

**Owner** indicates if this format is owned by the base package or by your implementation ( **Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI format. This information is display-only.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_FORMAT](#).

## XAI Adapter

To define a new adapter, open **Admin > XAI > XAI Adapter**.

#### Description of Page

Indicate a unique **Adapter Name** and **Description**.

Indicate the **Message Class**, which is the name of the Java class, implementing the adapter. The class should be one that is defined for an adapter. The adapter classes provided with the product are **BASEADA**- Core Adapter, **BUSINESSADA**- Business Requests Adapter and **XAICMNDADA**- XAI Command Adapter. Note that the values **LDAPIMPRTADA**, **SIEBELADA** and **STGUPADA** are no longer supported.

---

**FASTPATH:** Refer to [Message Class](#) for more information.

---

The following fields are not applicable for the **BusinessAdapter** adapter.

Use the **XAI JNDI Server** to indicate the name of the WebLogic JNDI server running your product. Refer to [XAI JNDI Server](#) for more information.

Indicate the **Default User** to be passed to your product server when this adapter is executed.

---

**NOTE:** If the XML request is sent over an HTTP connection, which has been authenticated, the authenticated User Id is passed to your product.

---

The **Default Date** format and the **Default DTTM** (date / time) **Format** specify date and date/time [formats](#) to use when a schema does not explicitly indicate formats.

**Owner** indicates if this XAI adapter is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI adapter. This information is display-only.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_ADAPTER](#).

## XAI Executer

To define a new Executer, open **Admin > XAI > XAI Executer**.

#### Description of Page

Enter a unique **Executer ID** and **Description**.

Indicate the **Message Class** for this executer. The class should be one that is defined for an executer. The executer class provided with the product is **XAIURLEXEC**- XAI Executer.

Indicate the appropriate **Executer URL**.

**Owner** indicates if this XAI executer is owned by the base package or by your implementation ( **Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI executer. This information is display-only.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_EXECUTER](#).

## XAI Group

XAI groups are used to process XML files, which contain a collection of [XML messages](#) to be uploaded in batch.

### XAI Group - Main

To define your XAI groups, open **Admin > XAI > XAI Group**.

#### Description of Page

Enter a unique **Group** and **Description** for the XAI Group.

Indicate the **Parser** used for this group. Possible values are **Dom Parser** and **StAX Parser**.

---

**NOTE:** **Dom Parser** reads the full XML document into memory and therefore is not ideal for larger XML documents.

---

Indicate the **XPath** and **XPath Value**, which an XML file receiver uses to identify which group a given XML file belongs to.

- For **StAX Parsers** the XPath is limited to the root element.
- For **Dom Parsers**, the XPath supports defining elements at a lower level than the root element.

### XAI Group - Attachments

Open **Admin > XAI > XAI Group** and navigate to the **Attachments** tab to define attachments for your group.

#### Description of Page



For each entry in the attachments collection, indicate the **Sequence** and the **Root Element**. Use **Include Elements** to indicate if **Parent** elements should be included along with the current element when applying the XAI rules.

---

**FASTPATH:** Refer to [XML Message File](#) for more information about how this is used.

---

## XAI Group - Rules

Open **Admin > XAI > XAI Group** and navigate to the **Rules** tab to define rules for your group.

### Description of Page

For each entry in the rules collection, indicate the **Sequence**, the **Priority**, the **XPath** name and **XPath Value** and the **XSL File Name**.

---

**NOTE: Include Parent.** If your attachment indicates that **Parent** elements should be included, be sure that the parent element is included in the XPath defined here.

---

**FASTPATH:** Refer to [XML Message File](#) for more information about how this is used.

---

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_RGRP](#).

## XAI Receivers

### XAI Receiver - Main

To define your XAI receivers, open **Admin > XAI > XAI Receiver**.

#### Description of Page

Enter a unique **Receiver ID** and **Description** for the XAI Receiver.

Indicate the **Message Class** for this receiver. The class should be one that is defined for a receiver. The receiver classes are **DWNSTGRCVR**- Download Staging receiver, **FILESCANRCVR**- Upload Files from a directory, **JMSRCVR**- JMS Queue receiver, **OUTMSGRCVR**- Outbound Message receiver, **STGCTLRRCVR**- Staging Control receiver, **STGRCVR**- Staging Upload Receiver and **TPCRCVR**- JMS Topic receiver, **XMLFILERCVR**- XML File receiver.

---

**FASTPATH:** For more information, refer to [Designing XAI Receivers](#) about different types of receivers.

---

Indicate whether or not this receiver is currently **Active**.

Identify the **Executer ID**. Select the XAILOCAL executer if the Message Class for this receiver is STGCTLRRCVR. Select the BYPASSXAI executer if the Message Class for this receiver is OUTMSGRCVR. For all other receivers select the XAIURL executer. For more information, refer to [XAI Executer](#).

Indicate whether the **MSG Encoding** is **ANSI message encoding** or **UTF-8 message encoding**.

The **Read Interval** indicates the number of seconds between read cycles.

**Start At Time** and **Duration** are not currently in use.

If the Message Class for this receiver is **FILESCANRCVR**, **STGRCVR**, **STGCTLRRCVR** or **XMLFILERCVR**, indicate the **XAI JDBC Connection**.

---

**FASTPATH:** Refer to [XAI JDBC Connection](#) for more information.

---

Turn on **Sequential Execution** if the received requests should be processed in [sequential order](#) (instead of multithreaded). If this value is turned on then XAI staging control records created by this receiver are marked for sequential execution.

#### JMS Information

The following information is only available if the Message Class is **JMSRCVR** or **TPCRCVR**.

Indicate the appropriate **XAI JMS Connection**

---

**FASTPATH:** Refer to [XAI JMS Connection](#) for more information.

---

Indicate the appropriate **XAI JMS Queue**.

---

**FASTPATH:** Refer to [XAI JMS Queue](#) for more information.

---

Indicate the appropriate and **XAI JMS Topic**.

---

**FASTPATH:** Refer to [XAI JMS Topic](#) for more information.

---

#### File Information

The following information is only available if the Message Class is **FILESCANRCVR** or **XMLFILERCVR**.

Use the **Scan Directory** to indicate where to look for new files.

In **Scan File**, indicate the file pattern. All files with names matching the pattern are uploaded into the staging upload table. For each file found, a record in the staging control table is created.

---

**CAUTION:** WARNING. MPL expects all files conforming to the Scan File pattern to be complete. If a file is in the process of being copied into the scan directory and its name conforms to the naming pattern, MPL still attempts to process it and may issue an error for the incomplete file. It is suggested that files first be copied into the scan directory with a different name that does not conform to the naming pattern, for example filename.xml.inprocess. Once the file copy/transfer is complete, rename the file to one that conforms to the naming pattern, for example, filename.xml.

---

The following information is only available if the Message Class is **FILESCANRCVR**.

Use the **XAI In Service Name** to indicate how the records in the file are mapped and how they are transformed to match a system service request structure.

## XAI Receiver - Context

Open **Admin > XAI > XAI Receiver** and navigate to the **Context** tab to define context for your receiver.

### Description of Page

The Context collection enables you to define a collection of **Context Types** and **Context Values** defining. Use this collection when you need to store an attribute of a receiver that is not catered for in the current table.

---

**NOTE:** The values for the Context Type field are customizable using the Lookup table. This field name is RCVR\_CTXT\_FLG.

---

## XAI Receiver - Response

Open **Admin > XAI > XAI Receiver** and navigate to the **Response** tab to define where to send responses to requests made by this receiver. Refer to [Designing Responses for a Receiver](#) for more information.

### Description of Page

The response collection enables you to define the destination (**Message Sender**) where responses to a request may be sent under various circumstances (**Event**). The events currently defined with the product are **All events**, **Message executed OK**, **Application Error**, **System Error**.

---

**NOTE:** The values for this field are customizable using the Lookup table. This field name is ON\_EVENT\_FLG.

---

## XAI Receiver - Groups

Open **Admin > XAI > XAI Receiver** and navigate to the **Groups** tab to the valid XAI groups for an XML file receiver.

### Description of Page

This collection is only available if the Message Class is **XMLFILERCVR**.

For each entry in the Group collection, indicate the **Priority** and the **Group**. Refer to [XAI Groups](#) for more information about defining groups.

### Where Used

Receivers are used by the XAI server and by the MPL server to process messages sent to the system from various sources.

## XAI Inbound Services

The XAI Inbound Services section in the registry is the main section of the registry. It is used to define the service characteristics. Basically, a service is defined by an Adapter responsible for executing the service, a pair of XML schemas and connection attributes. The Adapter defines the interface with the target application server, while the schemas define the structure of the request XML document expected by the service and the structure of the response XML document generated by the service.

## XAI Inbound Service - Main

To update an inbound service, open **Admin > XAI > XAI Inbound Service**.

---

**CAUTION:** Important! When adding new inbound services, carefully consider the naming convention of the XAI In Service Name. Refer to [System Data Naming Convention](#) for more information.

---

### Description of Page

Define a unique **XAI In Service Name**. This information is used in the system to identify the service. The service name is also the first XML element after the <Body> element in the XML request/response document. The system generates a unique **XAI Service ID**, which serves as the primary key.

**Owner** indicates if this XAI inbound service is owned by the base package or by your implementation ( **Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI inbound service. This information is display-only.

Indicate the **Adapter**, which defines the interface with the target application server.

---

**FASTPATH:** Refer to [XAI Adapter](#) for more information.

---

If adapter for this service should invoke a system service, then indicate the appropriate **Service Name**.

---

**FASTPATH:** Refer to [Service Program](#) for more information about defining services.

---

If adapter is the base package **Business Adapter** then **Service Name** does not appear. Instead, use **Schema Type** to indicate the type of object this service invokes and **Schema Name** to reference the object to invoke. Using this adapter, you may set up service to invoke [business objects](#), [business services](#) and [service scripts](#).

---

**FASTPATH:** Refer to [Designing XAI Adapters](#) for more information about the **Business Adapter**.

---

Use the **Description** and **Long Description** to describe the service.

Check the **Active** switch if this service is enabled and available for execution. If this switch is unchecked, then the service is defined in the registry, but not yet available for public use.

Check the **Post Error** switch to support [inbound message error handling](#) for messages that are not processed via the staging upload table.

Check the **Trace** switch if you would like the trace to be on for this particular service. If the general trace option is not activated, you can force a trace for a particular service.

---

**FASTPATH:** Refer to [Server Trace](#) for more information about trace functionality.

---

When the **Debug** switch is checked, debug information is generated on the XAI console when this service is executed. The debug information can be useful to resolve problems.

### Schema Definitions

---

**NOTE:** Request Schema and Response Schema are not applicable to services invoking schema-based objects. They do not appear when the **Business Adapter** is used.

---

The next two properties define the request and response XML schemas. The schemas were created using the [Schema Editor](#) and are SOAP compatible. The schema XML files are expected to be stored in the Schemas Directory on the Web server running the XAI server.

The **Request Schema** is the XML schema defining the service request. The request sent to the server must adhere to the schema definition.

The **Response Schema** is the XML schema defining the service response. The response generated by the XAI server corresponds to the response schema definition.

The same service may perform several actions on a business object. Use the **Transaction Type** to define the default action performed by a service. The transaction type can be provided when invoking a service, by dynamically specifying a transaction type attribute on the Service element of the XML request. This field may take the following values: **Read, Add, Change, Update, Delete, List** and **Search**.

---

**NOTE:** The difference between **Change** and **Update** is that for **Change**, all field values must be passed in with the request. Field values that are not passed in to the request are set to null. For **Update**, you need only pass the primary key field values and the values of the fields to be updated. All other fields retain their existing values.

---

Services, which perform a Search, may allow searching based on different criteria. When the Transaction Type value is **Search**, use the **Search Type** to define the default search criteria. The possible values are **Main, Alternate1, Alternate2, Alternate3, Alternate4, Alternate5** and **Alternate6**.

---

**NOTE:** This is a default definition only and it may be overridden at run time when the service is invoked. To override the search type at run time, you should specify the searchType attribute on the Service element of the XML request.

---

### XSL Transformation Definitions

Sometimes, the XML request document does not conform to the request schema, or the response document expected by the service requestor is not the one generated by the adapter. In such cases the request and/or the response documents must be transformed. The XAI server supports transformation through XSL transformation scripts. Transformation scripts may be applied to the request before it is passed to the adapter or applied to the response document before it is sent to the service requestor.

The **Request XSL** is the name of the XSL transformation to be applied to the request document before processing it. The transformation is usually required when the incoming document does not correspond to the XAI service request schema therefore it has to be transformed before it can be processed by the adapter.

The **Response XSL** is the name of the XSL transformation to be applied to the response document when the requester of the service expects the response to have a different XML document structure than the one defined by the response schema for the service.

Click the **WSDL URL** hyperlink to launch a separate window that contains the WSDL definition for the inbound service. Note that the server name and port number for the URL are built using a setting in the common properties file using the XAI HTTP Caller URL setting.

---

**NOTE:** Refer to [WSDL Catalog](#) for information on how to obtain the WSDL catalog for all XAI Inbound Services.

---

## XAI Inbound Service - Staging

The staging tab is used to define parameters for services that use the Staging Upload adapter.

---

**FASTPATH:** Refer to [XAI Upload Staging](#) for more information.

---

Open **Admin > XAI > XAI Inbound Service** and navigate to the **Staging** page to define attributes for your upload staging adapters.

### Description of Page

Indicate the **Staging File Type** to be processed by the staging upload service. Possible values are **Comma Delimited file**, **Database Extract** and **Sequential file**.

The format of the records in the input file are not in an XML format and do not correspond to an XAI service schema. As a result, the input record must be transformed into an XML message that conforms to an XAI service request schema. Enter the **Record XSL**, which indicates the XSL transformation script used to transform the input record into the appropriate XML message.

For **sequential files** and **Comma delimited files**, indicate the **Input File Name** to be processed.

---

**NOTE:** This parameter can be overridden in the **Staging Control** table when a request to execute such a service is made.

---

When the service takes its input from a **Database extract**, indicate the **JDBC Connection** used to connect to the database that contains the input data.

---

**NOTE:** If this value is not populated XAI uses the default JDBC connection, which is the current product database.

---

---

**FASTPATH:** Refer to [XAI JDBC Connection](#) for more information about defining these values.

---

Use the **Interface Name** to provide a description of the interface being implemented through this service.

## XAI Inbound Service - Parameters

This tab enables you to define parameters that are used as selection criteria by the DB Extract staging upload service.

Open **Admin > XAI > XAI Inbound Service** and navigate to the **Parameters** page.

### Description of Page

The **Parameters** that were defined under the Request element in the schema are displayed here. They are used to drive the extraction process. This tab only displays the list of parameters. The values for these parameters can later be entered when the control record to invoke this service is created.

---

**FASTPATH:** Refer to [Staging Control Parameters](#) for more information.

---

**Owner** indicates if this XAI inbound service is owned by the base package or by your implementation ( **Customer Modification**). The system sets the owner to **Customer Modification** when you add an XAI inbound service. This information is display-only.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [CI\\_XAI\\_IN\\_SVC](#).

---

**NOTE:** The information in this table is cached by the XAI server and by the MPL server. Changes to values in this table do not affect the runtime copy of these servers. Refer to [XAI Command](#) for information about refreshing a service.

---

## XAI Route Type

Refer to the documentation for your product to find out if XAI Route Type is supported.

## Running Your XAI Environment

---

**NOTE:** The XAI functionality is legacy functionality and not recommended for new implementations. This information remains in the documentation for upgrade purposes. Refer to [Incoming Messages](#) and [Outgoing Messages](#) for information about the recommended features for supporting sending and receiving external messages.

---

The [XML Application Integration](#) (XAI) module provides the tools and infrastructure that businesses require for integrating their third-party systems with the application.

This section describes the pages used to manage incoming and outgoing information via the XAI tool.

## XAI Staging Control

Refer to [Staging Control](#) for more information about the purpose and functionality of this page.

Navigate to this page using **Menu > External Message > XAI Staging Control**.

### Description of Page

The **XAI Staging Control ID** is a system generated unique identifier of this record.

The **XAI Staging Control Status** indicates the current state of this record.

**Pending** This status indicates that this record needs to be processed.

**In Progress** This status indicates that the background process, which uploads the staging records, is currently processing this record.

**Error** This status indicates that the background process found a problem with this record. When the data related to this record has been fixed, change the status back to **Pending** so that it will be processed again.

**Complete** This status indicates that this record has been processed.

The **Number of Uploaded Records** indicates how many XAI staging records related to this staging control record were uploaded.

The **Run Date Time** indicates the date and time this record was processed by the upload process.

The **User**, who created this staging control record, is captured.

If the data to be uploaded is found in a file, indicate the **Upload File Name**, which should include the directory path and file name.

---

**NOTE: Character Encoding.** Refer to [Sequential Input File](#) for information about specifying character encoding for a file.

---

**NOTE: Complete File.** MPL expects this file to be complete. If the file is in the process of being copied into the directory, MPL still attempts to process it and may issue an error for the incomplete file. The staging control record should only be created once the complete file is ready for upload.

---

Enter a description of the **XAI Interface Name**. This is simply information and is not used by the system.

Turn on **Sequential Execution** if you want the XAI upload staging records related to the staging control record to be processed in [sequential order](#) (instead of multithreaded).

The **XAI Service ID** tells the system how to create the XML related to the XAI upload staging record to be created. In Oracle Utilities Customer Care and Billing, for example, XAI Services with an adapter type of **CISStagingUpload** may be indicated here.

---

**FASTPATH:** For more information about XAI services, refer to [XAI Inbound Service](#).

---

Use the **Comments** to provide free format information related to this staging control record.

The collection of staging control parameters is used to define selection criteria when the staging control is accessing data from a database table. The **Staging Control Parameter** defines the field used in the WHERE clause and **XAI Staging Control Parm Value** defines the value to be used in the WHERE clause.

Refer to [Staging Control Parameters](#) for more information.

The **Message** collection displays any completion messages for XML requests related to the staging control record.

## XAI Upload Staging

The XAI upload page allows you to view the details or to fix errors for a request in the Staging Upload table. This page displays the text of the XML request document linked to the upload.

### XAI Upload Staging - Main

To view or modify an XAI upload document, navigate to **Menu > External Message > XAI Upload Staging** and navigate to the main tab.

#### Description of Page

The **XAI Upload Staging ID** is a system assigned identifier of the XAI upload staging record.

The **Application Service** identifies the internal system service called by the XAI tool to load/update the system data.

---

**NOTE:** The upload process does not use this. Rather, the application service is part of the XML Request. This field is used to help in searching for XAI Upload records.

---

The **XAI Staging Control ID** related to this XAI upload staging record is displayed.

The status of the upload is indicated by the **XAI Upload Staging Status**. Values are **Pending**, **In Progress**, **Complete**, and **Error**. If the record is in error, an error is written to the [XAI Upload Exception](#) table.

**Create Date/Time** indicates when the record was posted.

**Completion Date/Time** indicates when the work was completed.

If this XAI upload staging record is a response to an **XAI Download Staging** record, information about the related download staging record is displayed.

If there is an error with this record, you will see the error **Message** associated with this record. The message area is suppressed if there are no problems with the record.

Click the adjacent button to view the long explanation. The long explanation provides information about the cause of the error (and how to fix it).

The upload staging data appears in the **XML Request** text box.

## XAI Upload Staging - Response

To view the response to a completed XAI upload staging document, navigate to **Menu > External Message > XAI Upload Staging** and go to the **Response** page.

### Description of Page

The **XAI Upload Staging ID** is a system assigned identifier of the XAI upload staging record.

The system's response to the upload XML appears in the **XML Response** text box.

## Uploading XAI Staging Records

### Staging Control Layout

In order to load XML requests from flat sequential files, comma separated value (CSV) files and from database tables, you need to create a staging control record. The name of this table is **CI\_XML\_STG\_CTL**. The following table describes each column on this table.

Column Name	Length	Req'd	Data Type	Comments
XML_STG_CTL_ID	10	Y	N	This is the unique identifier of the record. This value does NOT have to be a random number, but it does need to be unique.
RUN_DTTM	26	N	Date/Time	Date and time this record was executed.
XAI_IN_SVC_ID	10	Y	A/N	This is the XAI service associated with this record. This service should have an adapter of <b>CISStagingUpload</b> .
XML_STG_FILE_NAME	250	N	A/N	This is the location and name of the file containing the data to be uploaded. This is used for comma delimited files and flat sequential files.
XML_INTFC_NAME	30	N	A/N	The name of the interface used to populate this table. This is simply information and is not used by the system.
XML_STG_STATUS_FLG	2	Y	A/N	Must be set to <b>P (Pending)</b> .
NT_XID_CD	30	N	A/N	This field is not in use.
NT_UP_XTYPE_CD	30	N	A/N	This field is not in use.
USER_ID	8	Y	A/N	The ID of the user who created this record.



COMMENTS	254	N	A/N	Use the free format comments, if necessary.
NBR_RECORD_UPLD	10	N	A/N	Leave this blank. This will be populated by the process which creates upload staging records for this control record.

## Staging Control Parameters Layout

If your staging control record should read a database table, the XAI service may include selection criteria in its **WHERE** clause. If that is the case, then your staging control record should populate the selection criteria. The name of this table is **CI\_XML\_STGCTL\_P**. The following table describes each column on this table.

Column Name	Length	Req'd	Data Type	Comments
XML_STG_CTL_ID	10	Y	N	The unique identifier of the staging control record.
XML_STG_CTL_PARM	18	Y	A/N	Indicate the element in the WHERE clause whose value will limit the selection of records.
XML_STG_CTL_PVAL	250	Y	A/N	Use this to indicate the value used to limit selection of records.

## Upload Staging Layout

You create an XAI upload staging record directly for each XML request you wish to make. The name of this table is **CI\_XML\_STGUP**. The following table describes each column on this table.

Column Name	Length	Req'd	Data Type	Comments
XML_STGUP_ID	15	Y	N	This is the unique identifier of the record. This value does NOT have to be a random number, but it does need to be unique.
APP_SVC_ID	20	Y	A/N	This is the foreign key to the application service table, and identifies the application service that is being requested.
COMPLETE_DTTM	26	N	Date/Time	Completion date and time.
CRE_DTTM	26	Y	Date/Time	Creation date and time.
RETRY_COUNT	3	Y	N	Number of retries to process the request.
XML_REQUEST	30,000	Y	A/N	The XML request document.
XML_RESPONSE	30,000	Y	A/N	The XML response document.
XML_STG_CTL_ID	10	Y	A/N	The id of the staging control record linked to this record.
XML_STG_STATUS_FLG	2	Y	A/N	Must be set to <b>P (Pending)</b> .

## XMLUP-PR - Purge XAI Upload Objects

**Completed** XAI upload staging objects should be periodically purged from the system by executing the **XMLUP-PR** background process. This background process allows you to purge all **Completed** XAI upload staging objects older than a given number of days.

We want to stress that there is no system constraint as to the number of **Completed** XAI upload staging objects that may exist. You can retain these objects for as long as you desire. However we recommend that you periodically purge **Completed** XAI upload staging objects as they exist only to satisfy auditing and reporting needs.

## XAI Dynamic Upload

This page displays the XAI upload record. However, rather than displaying the XML Request as an XML document, it dynamically builds the screen with prompts and input fields to allow for changes without having to understand the XML. This screen is useful for fixing errors. Once the data is fixed, the status of the record can be changed back to **Pending** to once again attempt to upload this record.

---

**NOTE: The Dynamic Nature of this Transaction.** This transaction is designed to display the schema of the represented XAI upload staging. The system reads the XML schema associated with the upload XAI service and builds the tab(s) with the associated prompts and fields. For example, if the upload record is relevant to the Premise Maintenance service, Premise Characteristics and Premise Geo Types tabs are dynamically built.

---

## XAI Dynamic Upload - Main

Navigate to this page using **Menu > External Response > XAI Dynamic Upload**.

### Description of Page

The **XAI Upload Staging ID** is a system assigned identifier of the XAI staging record.

The **Application Service** identifies the internal system service called by the XAI tool to load/update the system data.

---

**NOTE:** This is not used by the upload program. Rather, the application service is part of the XML Request. This field is used to help in searching for XAI upload staging records.

---

The **XAI Staging Control ID** related to this XAI upload staging record is displayed.

The status of the upload is indicated by the **XAI Upload Staging Status**. Values are **Pending**, **In Progress**, **Complete**, and **Error**.

**Create Date/Time** indicates when the record was posted.

**Completion Date/Time** indicates when the work was completed.

If there is an error with this record, you will see the error **Message** associated with this record. The message area is suppressed if there are no problems with the record.

Click the magnifying button to view the long explanation. The long explanation provides information about the cause of the error (and how to fix it).

The **Transaction Type** indicates the action that should be performed for the application service when this document is uploaded. The valid values are **Add**, **Change**, **Delete**, **List**, **Read**, **Search** and **Update**.

Set the **Trace** option to **yes** to request level tracing to be executed inside the XAI tool. This will result in information written to a file, which may be useful in debugging. Refer to the XAI tools documentation for more information.

The bottom portion of the screen will contain field prompts and input fields for the data associated with the XML request linked to this upload record. The system dynamically builds this portion of the page by reading the XML request associated

with the upload record. A user can change data in the displayed fields. This will result in a change to the XML request associated with the upload record.

---

**NOTE:** This page will also generate other tabs dynamically for any collections that exist for the service being displayed. These will vary based on the service. The response tab is the only other tab, which is always present.

---

## XAI Dynamic Upload - Response

Navigate to this page using **Menu > External Message > XAI Dynamic Upload** and go to the response tab.

### Description of Page

The **XAI Upload Staging ID** is a system assigned identifier of the XAI staging record.

The system's response to the upload XML appears in the **XML Response** text box.

## XAI Upload Exception

A record is written to the XAI upload exception table for every XAI upload staging record that is in error.

To view the messages associated with the exception records, schedule the **TD-XAIUP** background process. This process generates a To Do entry for every record in the XAI upload exception table.

After correcting the cause of the error, drill into the [XAI Upload Staging](#) page and change the status from **Error** to **Pending** and the system will attempt to process the record again.

## XAI Download Staging

### XAI Download Staging - Main

To view individual XAI download staging records associated with a notification download staging record, open **Menu > External Message > XAI Download Staging**.

#### Description of Page

**XAI Download Staging** displays information about the record.

The **Download Staging ID** and **XAI Route Type** are the primary identifiers for this record.

The **XAI Download Status** is displayed. The values are **Pending**, **Complete**, **Canceled** or **Error**. When a record is in **Error**, it is displayed on the [XAI Download Exception](#) table.

Values from the NDS record associated with this XAI download staging record are displayed including **Service Provider**, **Notification Download Type**, **NT Download Status Flag**, **Retry Count** and the **Context** collection.

### XAI Download Staging - Request

To display the XAI Request built by the download staging receiver, open **Menu > External Message > XAI Download Staging**.

#### Description of Page

The **XML Request** built by the download staging receiver is displayed.

## XAI Download Staging - Response

To display the Response to this request, open **Menu > External Message > XAI Download Staging**.

### Description of Page

The **XML Response** to this message is displayed.

## XAI Download Exception

A record is written to the XAI download exception table for every XAI download staging record that is in error.

To view the messages associated with the exception records, schedule the **TD-XAIDN** background process. This process generates a To Do entry for every record in the XAI download exception table.

## Maintaining Your XAI Environment

This section describes various tools provided to enable your XAI administrators to more easily maintain your XAI environment.

## XAI Submission

This page exists for testing purposes. It allows you to create an XML request document and submit it to the system, to ensure that the XML has been built correctly.

### XAI Submission - Main

To submit an XML document for testing, navigate to **Admin > XAI > XAI Submission** and navigate to the main tab.

#### Description of Page

This page is used to test XML schemas, which are defined for the XAI tool. Enter an appropriate XML document in the **XML Request** field. Typically, you define the XML schema using the schema editor in the XAI application. Then you would copy and paste the document here, then modify the schema to enter actual data for testing purposes.

When you have entered the document, choose **Save** to submit this document to the system. Note that this request information is not saved anywhere. It simply calls the system with the appropriate service name and executes the XML request.

Navigate to the **Response** tab to view the response.

### XAI Submission - Response

To view the response to a XML document for testing, navigate to the response tab.

#### Description of Page

After choosing **Save** on the main tab to submit a test for an XML request, the response to your request is displayed in the **XML Response** text box.

## XAI Dynamic Submission

This page exists for testing purposes. It is similar to the XML Submission page, but it dynamically builds your XML document based on a selected XAI service and data that you enter.

### XAI Dynamic Submission - Main

To create and submit an XML document for testing, navigate to **Admin > XAI > XAI Dynamic Submission** and navigate to the main tab.

#### Description of Page

Select the **XAI In Service Name**, which identifies the XAI Inbound Service called by the XAI tool to load/update the system data. After selecting, click **Load UI**. This button causes the system to read the XML schema associated with the service and build the screen with the associated prompts and fields, which enables a user to enter data.

---

**NOTE: Testing schema.** This page tests submitting the schema referenced on an XAI inbound service. If your service references an XSL transformation script, that information is ignored. This page tests post-XSL transformation.

---

Select the **Transaction Type** associated with this XML request. The valid values are **Add, Change, Delete, List, Read, Search** and **Update**. This information is built into the XML request document.

Set the **Trace** option to **yes** to request level tracing to be executed inside the XAI tool. This results in information written to a file, which may be useful in debugging.

The bottom portion of the screen contains field prompts and input fields for the data associated with the XML request linked to this service. The system dynamically builds this portion of the page by reading the XML Request associated with the service. You can enter data in the displayed fields.

---

**NOTE:** This page also generates other tabs dynamically for any collections that exist for the service being displayed. These tabs vary based on the service. The response tab is the only other tab that is always present.

---

Enter values in the appropriate fields. You need to have some knowledge of what information is needed based on the service and transaction type. For example, if your service is accessing the account record and you want to **read** the record, you only need to provide the account id. However, if your transaction type is **add**, you need to fill in all the fields necessary for adding an account. If you have not entered values correctly, you receive an error in the Response.

When finished entering values in the fields, click **Show XML** to see the XML request built based on the schema and the values of the fields entered.

If everything looks OK, click **Submit** to execute the XML request. Note that this request information is not saved anywhere. It simply calls the system with the appropriate service name and executes the XML request.

You are brought to the Response tab, where you may view the response.

### XAI Dynamic Submission - Response

To view the response to the XML document submitted for testing, navigate to the response tab.

#### Description of Page

After choosing Submit on the main tab to submit a test for the XML request built dynamically, the response to your request is displayed in the **XML Response** text box.

## Additional XAI Tools

This section introduces some additional tools to help you maintain your XAI environment.

### XAI Service Export

The service export page allows you to export the definition of an XAI Inbound Service to a file. This function may be helpful if you need to copy the definition of this service to a separate environment. To export a service, open **Admin > XAI > XAI Service Export**.

#### Description of Page

Upon entry into this page, you are provided with the current list of **XAI In** bound **Services** and their **Description** s. Use the **XAI In Service Name** search field to find the XAI service that you would like to export.

Use the **Export?** column to indicate which XAI service(s) you would like to export. Once you have selected your services, choose **Save**.

---

**NOTE:** If multiple services are selected, they are exported together into the same output file.

---

You are presented with the standard File Download dialogue where you can open or save the file.

### XAI Service Import

The service import page allows you to import the definition of an XAI Inbound Service from a file into the XAI service table. This function may be helpful if you need to copy in the definition of this service from a separate environment. To import a service, open **Admin > XAI > XAI Service Import**.

#### Description of Page

Upon initial entry into this page, you are provided with an input field, where you can enter the file name to import. Click **Browse** to search for the desired file in a directory.

Once the file is identified, click **Read File**, to read in the contents of the file.

---

**NOTE:** The format of the file must include tags indicating the column names for XAI Inbound Service table along with the values of the columns. For an example of how the format should be, simply go to the [XAI Service Export](#) page, export a Service and view the format of the resulting file.

---

Once the file has been read in, the list of XAI services found defined within the file is displayed in the Import grid, identified by their **XAI In Service Name** and **Description**. In the **Import?** column, indicate which services to import.

If a service with this service name already exists in the table, you must check the **Overwrite Existing** switch in order to indicate that the imported file information should replace the current service. An [XAI Inbound Service](#) that is provided as part of the system (i.e., with an owner of **Base Product**) may not be overwritten.

Click **Save** to proceed with the import. If any problems are found, information is displayed in the **Message Text** column.

### XAI Command

Use the XAI Command page to send commands to the XAI and MPL server. To execute a command, open **Admin > XAI > XAI Command**.

## Description of Page

The following operator commands may be sent to the XAI server. For each of these commands, you may check **Also Sent to MPL URL**, in which case, the command is also sent to the MPL server. You need to indicate the URL of the MPL server.

**Display Registry** Use this command to display the current registry information that the XAI instance is running with.

**Refresh Code and Description** This is related to an attribute in the schema editor where you may indicate that the description of a control table code should be returned along with the code itself. This information is kept in cache memory in the server. As a result, changes made to descriptions have no effect on the runtime server. This command clears the cache of control table codes and descriptions accessed by the server.

**Refresh Registry** The registry contents are kept in cache memory in the server. As a result, making changes to the registry control tables has no effect on the runtime server. Use this command to refresh the contents of the registry cache with the new values in the registry control tables. The command reloads all registry control table data into the server.

**Refresh Schema** Schema definitions are stored in cache memory on the XAI server. As a result, modifying a schema definition has no effect on the runtime server. To refresh schema definitions, use the Refresh Schemas command.

**Refresh Service** Service definitions are stored in cache memory on the XAI server. As a result, modifying an XAI inbound service definition has no effect on the runtime server. To refresh service definitions, use the Refresh Service command. You are prompted to indicate which service to refresh.

**Refresh XSL** XSL Transformation script definitions are stored in cache memory on the XAI server. As a result, modifying an XSL transformation definition has no effect on the runtime server. To refresh XSL transformation definitions, use the Refresh XSL command.

**Trace On** Use this command to start the XAI server trace.

**Trace Off** Use this command to stop the XAI server trace.

**XAI Trace Clear** Use this command to clear the contents of the trace file.

**XAI Trace Swap** Use this command to rename the current trace file by appending the date and time to the end. A new trace file is then created with the name as defined in the [Message option](#) page.

The following operator commands can be sent to the MPL server. You must set the URL of the MPL server first.

**MPL Refresh Executer** Executer definitions are stored in cache memory. As a result, adding or modifying executer definitions has no effect on the runtime server. Use this command to refresh executer definitions. You are prompted to indicate the executer to refresh.

**MPL Refresh Receiver** Receiver definitions are stored in cache memory. As a result, adding or modifying receiver definitions has no effect on the runtime server. Use this command to refresh receiver definitions. You are prompted to indicate the receiver to refresh.

**MPL Refresh Sender** Sender definitions are stored in cache memory. As a result, adding or modifying sender definitions has no effect on the runtime server. Use this command to refresh sender definitions. You are prompted to indicate the sender to refresh.

**MPL Start Receiver** Use this command to start a particular receiver. You are prompted to indicate the receiver to start.

**MPL Stop** Use this command to stop all MPL activity. It stops all receivers and waits for all executers and senders to complete.

**MPL Stop Receiver** Use this command to stop a particular receiver. You are prompted to indicate the receiver to stop.

**MPL Trace On** Use this command to start the MPL server trace.

**MPL Trace Off** Use this command to stop the MPL server trace.

When you have chosen the appropriate command and indicated any extra information, click **Send Command** to send the command to the server(s).

If you have sent a command to the XAI Server, then the bottom portion of the screen displays the response in the **XAI Response**. If you have sent a command to the MPL Server, then the bottom portion of the screen displays the response in the **MPL Response**. If you have sent a command to both servers, the bottom portion of the screen displays both responses.

## MPL Exception

The MPL Exception table is used by the MPL to keep information about requests that resulted in a system error. These are errors that occurred inside the MPL. For example, if the MPL fails to send a request to XAI (maybe WebLogic is down), this is a system error, which would be logged in the MPL exception table.

There are errors that are defined recoverable. This means that the MPL will retry the action that failed, according to the parameters it received.

## Server Trace

The XAI server traces every request and response. The requests/responses are written to a trace file on the server. The trace file may be viewed using the [Trace Viewer](#).

## Starting the Trace

The log starts automatically based on definitions in the [Message Options](#) in the traceType and traceFile options. To manually start the trace:

- Navigate to **Admin > XAI > XAI Command**.
- Select the **Start Trace** command from the command dropdown
- Click **Send Command**

## Stopping the Trace

- Navigate to **Admin > XAI > XAI Command**.
- Select the **Stop Trace** command from the command dropdown
- Click **Send Command**

## Trace Viewer

Use the Trace Viewer utility to view the log file. The Trace Viewer is installed when you install the XAI client tools. It can be found in the XAI program group under Start/Programs.

## Main Page

When the Trace Viewer starts, select a trace file to view. A trace file may be opened in one of two ways:

- To open a trace file directly from its location on the web application server, use the **File, Open HTTP** menu item and provide the appropriate URL.
- To open a trace file on the local/network file system use the **File, Open** menu item

### Description of Page

Once a trace file is opened, it displays a list of all the requests on the left side including the **Service Name**, the **Start Time** and the **End Time**.

To display the XML contained in the request and response entries for a displayed request, select a request entry.



## Filtering Options

Since the trace file may contain a very large number of messages, the trace viewer limits the number of messages that can be displayed. It does that by displaying messages traced within the last x number of **Minutes**, **Hours** or **Days**.

Use the **Max Messages** to limit the number of messages displayed.


---

**NOTE: Default Note.** By default, the Trace viewer displays the first **200** messages in the trace file.

---

To view only errors in the trace, check the **Show only Errors** option.

---

**NOTE: Refresh Display.** After changing any of the above filtering options, click the refresh button  in order to redisplay the request entries based on the new options.

---

The **First Message Found** field indicates the date and time of the earliest entry in the trace file.

## Viewing as Text

To view the trace file as text rather than viewing each entry in its XML format, use the **View, As Text** menu option. The contents of the trace file are displayed in text format in a separate window.

## Statistics Page

Use the **View, Statistics** menu item to view the statistic page, which displays performance statistics about the XAI services that were executed in the XAI trace file.

For each type of XAI Service and transaction type, it displays the following information based on the requests traced in the XAI trace file:

- The **Service Name** with the transaction type in parentheses
- The **Number of calls** for this service in the listed trace records
- The **Average duration Time** (in seconds)
- The **Max** imum duration **Time** (in seconds)
- The **Min** imum duration **Time** (in seconds)

---

**NOTE: Requests Included in Statistics.** Only requests falling in the time selection criteria and listed on the main log viewer are processed for calculating the statistics.

---

To display a Duration Chart for a particular service, check the Service. A chart such as the one below is displayed.

# Integrations

---

This chapter provide high level information about product integrations supported for all products that use Oracle Utilities Application Framework.

## LDAP Integration

Organizations commonly use a Lightweight Directory Access Protocol (LDAP) security repository as a source of security credentials. The system provides support for importing users and groups from an external LDAP repository into the product to populate Users and User Groups in the system. Once imported, all user and group functions are available. You can resynchronize your LDAP users and groups at any time.

---

**NOTE: Import only.** The system currently supports importing LDAP information. Exporting users and groups from the system to LDAP is not provided.

---

**NOTE:** Additional configuration. When importing new users and / or groups, additional configuration is needed in the base product. For example, after importing a new user group and its users, the user group configuration should be updated to define the valid application services for the user group. After importing a new user, additional configuration may be needed on the user such as valid To Do Roles, valid Home Page, etc.

---

**FASTPATH:** Refer to [Defining Security and User Options](#) for more information about the application security and what it controls.

---

This section the functionality provided in the framework application that supports LDAP. Refer to the *LDAP Integration* white paper for more information about typical steps related to the full integration.

## LDAP Integration Overview

This topic provides a high level overview of the integration process.

At a high level, the base product provides a process to import user group and / or user definitions from and LDAP repository. This is a one way integration.

- When importing a user, if it is not already found in the system, it will be added; otherwise its attributes will be updated according to the imported information.
- When importing a user group, if it is not already found in the system, it will be added; otherwise its attributes will be updated according to the imported information.
- When importing a user, its user group links will be updated as per the information in the import file. In addition, if there are any user groups linked to the user that are not found system, they will be added (however, the other users linked to that group in the LDAP repository will not be added as part of this step).
- When importing a user group, its user links will be updated as per the information in the import file. In addition, if there are any users linked to the user group that are not found system, they will be added (however, the other user groups linked to that user in the LDAP repository will not be added as part of this step).
- The import will not cause any deletions of the User or User Group to occur.

### A Batch Process Initiates the Import

A batch process is used to initiate the import of information from the LDAP repository. **F1-LDAP** may be submitted ad hoc or may be set up in a scheduler to periodically re-sync the information from the LDAP repository into the application.

The batch process uses parameters to define how to connect to the LDAP repository. In addition, parameters are used to indicate which user or group is being imported.

### Adjusting Data to Import

The system provides several mechanisms for adjusting data that is being added to the system:

- There is an **LDAP Import Preprocess** algorithm plug-in spot on the [installation](#) record. Algorithms plugged in here are called by the batch process prior to the add or update of any records. It may be used to make adjustments to the data before doing updates in the application.
- Specifically for creating or updating Users, the **F1-IDMUser** business object is used to add and create the user. The standard BO Preprocessing algorithm plug-in spot may be used to adjust data prior to creation.
- The LDAP mapping file supports some attributes to perform simple modifications to data.

- The **transform** attribute supports values to truncate values or to convert data to upper case.
- The **autoGenerate** attribute is specific to the User ID field. Setting this to true will trigger code that will automatically populate the User ID based on the user's name. Refer to [LDAP Mapping](#) for more information.

## Performing Additional Processing After Import

The system provides a plug-in spot on the [installation](#) record called **LDAP Import**. Algorithms plugged into this spot are called after users or user groups have been added or updated. It may be used to perform any extra processing that may need to be executed.

In addition, for any additional processing related to the creation or update of a User, the standard [Business Object plug-ins](#) may be used for the **F1-IDMUser** business object which the LDAP batch process uses to create or update users.

## Configuring LDAP Integration

To interface the LDAP based security repository with the authorization component of the Oracle Utilities Application Framework product the following must be performed:

- The location and port number of the LDAP based security repository must be defined to in the JNDI Server.
- • The LDAP based security repository must be mapped to the Oracle Utilities Application Framework security model. This mapping is expressed as an XML file containing the LDAP query, rules and defaults used in the transformation.
- • The mapping file must be configured on the **F1-LDAP** batch job.

### Define the JNDI Server

The first step in the configuration process is to define the location of the LDAP based security repository server so that the interface can connect to the physical attributes of the interface. This is done by creating a [JNDI Server](#).

---

**NOTE:** The LDAP server is strictly not a JDNI source but is treated as a JNDI source for the integration.

---

Enter a reasonable JNDI Server name and description.

Populate the **Provider URL** using the format **ldap://<hostname>:<portnumber>** where **<hostname>** is the host of the LDAP server and **<portnumber>** is the port used for the interface.

For the **Initial Context Factory**, the interface uses the standard **com.sun.jndi.ldap.LdapCtxFactory** provided with java for the LDAP interface. If your vendor supplies a custom context factory it may be used. Refer to the documentation provided with your LDAP based security repository for further information.

### Define Mapping

The critical component of the interface is a file that describes the mapping between the LDAP based security repository and the system's security model. This file contains the mapping, rules and queries used by the LDAP batch program to provide the interface. The LDAP batch job includes the reference to the mapping file as a parameter. Refer to [LDAP Mapping](#) for more information on defining the mapping file.

### Configure LDAP Batch Process

At this point, many parameters for the **F1-LDAPbatch control** can be updated with system wide configuration.

- JDNI Server, User and Password may all be configured appropriately. Note that it is recommended that the **Security** setting for the Password be set to **Encrypt**.
- The LDAP Configuration File should be populated with the name and location of the LDAP Mapping file.

---

**NOTE: Group and User Parameters.** The assumption is that the Group or User input parameters are specific to a given import request and as such would not be populated as part of a configuration step.

---

**NOTE: L2 Cache.** The LDAP Import batch process requires the L2 Cache to be disabled since it needs to perform some updates in the outside of the worker threads. Any environment using LDAP Import must set **spl.runtime.batch.L2CacheMode=OFF** in the **threadpoolworker.properties** file. It is recommended to run the LDAP import in its own dedicated threadpoolworker.

---

## LDAP Mapping

An LDAP repository consists of multiple entries. Each entry represents an object in the directory that is identified by a Distinguished Name (DN) and may contain one or more attributes. In a typical LDAP repository there is usually an entry for users and an entry for groups. The connection between users and groups may be implemented in two different ways:

- The users belonging to a group are defined in a special multiple-value attribute on the Group entry.
- The groups to which a user belongs are defined in a special multiple-value attribute on the User entry.

The mapping between LDAP security objects and base security objects is stored in an XML document that can be processed by the LDAP import batch job. As part of setting up your system for LDAP import, you need to define this mapping. The base package provides a sample mapping file called **ldapdef.xml** that can be used as a starting point and changed per your business requirements and your particular LDAP repository.

Once you have defined the mapping XML document, this is configured as a parameter in the **F1-LDAP** batch job.

The XML structure:

- The **LDAPEntry** element maps the LDAP entries to system objects (User or Group). The mapping file must contain one and only one LDAPEntry element for User and one for Group.
- The **LDAPCDXAttrMapping** element within the LDAPEntry element maps attributes in the LDAP entry to attributes in the system object.
- The **LDAPEntryLinks** element describes objects linked to the LDAP entry. When mapping the user entity you need to describe how the groups the user belongs to are retrieved. When mapping the group entity you need to describe how the users contained in the group are retrieved.

The following table describes the attributes to define for each element.

Element	Attribute	Description
LDAPEntry	name	The name of the LDAP entry: - <b>Group</b> - <b>User</b>
	baseDN	The base distinguished name in LDAP for this entry.
	cdxEntity	The name of the base product entity to which the LDAP entry is mapped: - <b>Group</b> - <b>User</b>
	searchFilter	An LDAP search filter that is used to locate LDAP entries. A %searchParm% string in that filter is replaced by the value from the user or group parameter from the <b>F1-LDAP</b> batch job submission.
	Scope	Sets the scope of the search. Valid values are: - onelevel (the value normally used) - subtree
LDAPCDXAttrMapping	ldapAttr	The name of the LDAP attribute to be mapped. Note that this may be referenced more than once to allow one LDAP element to map to multiple base product elements. For example, if an email address should be used both for the Login ID and the Email Address.
	cdxName	The name of the base product attribute to be mapped. For User, this is the element within the F1-IDMUser business object.

Element	Attribute	Description
		For Group, this is either the 'group' or the 'description'.
	default	The default value that will be assigned to the element referenced in the cdxName attribute when the following occurs: <ul style="list-style-type: none"> <li>- The LDAP attribute contains a null or empty value</li> <li>- The LDAP attribute does not exist or is not specified.</li> </ul> Default values are applied only when creating a new entity and are not applied to updated entities.
	autoGenerate	Set this to <b>true</b> in order to turn on auto generation of the user ID. If this is true, the system will define user id as <first initial of first name>+<last name> all uppercase, to a maximum of 8 digits. If an existing user is found for the generated ID, a number will replace the eight digit (or be appended to the end). The system will increment the number until a unique ID is found.
	transform	Use this attribute to indicate if a transformation of the data should occur. Valid values: <b>uppercase truncate</b> . Note that this attribute should not be used in conjunction with the autoGenerate attribute.
LDAPEntryLink	linkedToLDAPEntry	The name of the linked entity ( <b>User</b> or <b>Group</b> ). Use User when describing the Group entity. Use Group when describing the User entity.
	linkingLDAPAttr	The multiple-value attribute name on the LDAP entity that contains the linked entity.
	linkingSearchFilter	The search filter to be applied to retrieve the list of linked objects, for example: (&objectClass=group)(memberOf=%attr%)  The search filter may contain the string % attr % that acts as a substitution string and is replaced at run time by the value of the attribute named "attr" of the imported entity. If the LDAP entry you are describing is a Group and the string is %name%, it is replaced by the value of the "name" attribute of the group you are importing. If the LDAP entry you are describing is a User and the string is %dn%, it is replaced by the "dn" attribute of the User you are importing.
	linkingSearchScope	Sets the scope of the search. Valid values are: <ul style="list-style-type: none"> <li>- <b>onelevel</b> (the value normally used)</li> <li>- <b>subtree</b></li> </ul>

## Sample Mapping

The following XML describes a sample mapping. The example makes the following assumptions:

- The base product attribute **displayProfileCode** is defaulted to "NORTHAM" when adding a new user.
- The LDAP Group entry contains the list of users belonging to the group in the **departmentNumber** attribute.
- The groups to which a user belongs are retrieved by applying a search filter.

```
<LDAPEntries>
  <LDAPEntry name=" User" baseDN="ou=people,dc=example,dc=com" cdxEntity=" user" searchFilter=" (&
(objectClass=inetOrgPerson)(uid=%searchParm%)) ">
    <LDAPCDXAttrMappings>
      <LDAPCDXAttrMapping ldapAttr="uid" cdxName=" user" />
      <LDAPCDXAttrMapping ldapAttr="cn" cdxName="externalUserId" />
      <LDAPCDXAttrMapping cdxName="language" default=" ENG" />
      <LDAPCDXAttrMapping ldapAttr="givenName" cdxName="firstName" />
      <LDAPCDXAttrMapping ldapAttr="sn" cdxName=" lastName" />
      <LDAPCDXAttrMapping cdxName="displayProfileCode" default="NORTHAM" />
      <LDAPCDXAttrMapping cdxName="toDoEntriesAge1" default="30" />
      <LDAPCDXAttrMapping cdxName="toDoEntriesAge2" default="90" />
      <LDAPCDXAttrMapping cdxName="userEnable" default="ENBL" />
    </LDAPCDXAttrMappings>
    <LDAPEntryLinks>
      <LDAPEntryLink linkedToLDAPEntry="Group" linkingLDAPAttr="departmentNumber" />
    </LDAPEntryLinks>
  </LDAPEntry>
  <LDAPEntry name="Group" baseDN="ou=people,dc=example,dc=com" cdxEntity=" Group" searchFilter=" (&
(objectClass=organizationalUnit)(ou=%searchParm%)) ">
    <LDAPCDXAttrMappings>
      <LDAPCDXAttrMapping ldapAttr="name" cdxName="Group" />
      <LDAPCDXAttrMapping ldapAttr="description" cdxName=" Description" default="Unknown" />
    </LDAPCDXAttrMappings>
    <LDAPEntryLinks>
      <LDAPEntryLink linkedToLDAPEntry="User" linkingSearchFilter=" (&
(objectClass=inetOrgPerson)(departmentNumber=%distinguishedName
%))" linkingSearchScope="onelevel" />
    </LDAPEntryLinks>
  </LDAPEntry>
</LDAPEntries>
```

```
</LDAPEntryLinks>
</LDAPEntry>
</LDAPEntries>
```

## Oracle Identity Manager Integration

The *Oracle Identity Manager* product allows a site to centralize their user definitions and password rules to manage and deploy across the enterprise set of products. When an employee joins an organization, changes their name or departs an organization their security presence across an enterprise must be appropriately managed. Oracle Identity Manager allows for users to be provision and managed in a central location.

An integration is provided to allow the ability to create, maintain and remove users in the identity management product and sync those changes to the users defined in the application. The following sections provide additional details about the integration with respect to configuration steps required in an Oracle Utilities Application Framework based product. For more information about the configuration required in the identity management product, refer to the *Identity Management Suite Integration* white paper.

### Template User Functionality

The user object in this product captures configuration used to control access but also preferences. The identity management product allows for extending the configuration to capture user configuration that is specific to this product. However, it does not support providing searches or dropdowns to select valid values. For example, to define the user's Home Page requires the reference to a navigation option. To set up your business process such that the home page is configured when defining the user in the identity management product dictates that the security user types in the correct navigation option reference.

On the other hand, to define a minimal amount of user information in the identity management product may result in a two step process for defining users: first define them in the identity management product with the basic authentication details and setting system defaults for some important fields, then after submitting the new user to be added to this product, navigate to the [user](#) page in this product and fill in all the configuration that is specific to this product.

The product provides support for defining a template user that can facilitate the definition of users and reduce some of the challenges listed above. The concept is as follows:

- Define a template user for each broad category of users in the system. For example, Oracle Utilities Mobile Workforce Management may define the following template users: Dispatcher, Mobile Worker, System Administrator and Contractor. Each user would define the typical configuration for users of that type including the home page, the user groups, the To Do roles, the portal preferences, etc.
- When extending the configuration in the identity manager product, simply map the information that is unique to a user and in addition, define a field for the template user. For example, you may choose to only capture the Name (first and last), Email address and User IDs for the user along with its Template User (which is mapped to a user characteristic). Additional fields may be included for capture in the identity management product when defining new users as per an implementation's business needs. For example, if the organization covers multiple time zones, perhaps it is easier to define the user's time zone when defining the user in the identity management product.
- When the new user is uploaded to the system, the interface uses the user BO **F1-IDMUser** to create the user. The BO includes a preprocessing algorithm that looks for the existence of a template user (sent as a characteristic of type **F1-TMUSR**). All the information from the template user will be copied onto the new user record except for the information passed in from the identity manager. The template user is captured on the newly created user via a characteristic for information / audit purposes.
- Once the new user is created, its configuration can now be adjusted, if applicable. Note that the template user is only a tool used when adding a user. Updates to the template user will not "ripple" to all the other users that were created based on this template.

## Configuring Template Users

Before configuring template users, all the administrative control tables that are part of User configuration must be defined, including time zones, display profiles, To Do roles, data access roles and user groups.

The next step is to define the user configuration for your system users. During this exercise, you will find that you have broad categories of users. But you will also see that within a given category of user there may be variations in the user privileges and preferences. For example, perhaps there are supervisors within the Mobile Worker role that have more security privileges than a typical Mobile Worker. In addition, there may be variations based on the attributes of the users themselves. For example, maybe your organization exists in multiple time zones and some of your workers are in one time zone and some are in the other.

At this point your security users that are designing their user provisioning procedures must decide the following:

- What information about a new user will be captured in the identity system (besides the expected information like Name, Email and the User IDs)? For example, for the case of multiple time zones, maybe the best solution is to capture the time zone when defining the user.

What information is defined on the template user and how many template users should be created to reduce the need for manual steps or additional data captured in the identity management system? In the case of multiple time zones, proliferating the template users to have one set for one time zone and another set for the other time zone may not make sense since this is one field that is different. However it may be reasonable to create additional templates in the case of variations in the levels of privileges for workers of a different category. So rather than template users for Dispatcher, Mobile Worker, System Administrator and Contractor, your organization may have template users for Dispatcher, Mobile Worker, Mobile Worker (Supervisor), System Administrator, Contractor (Short Term) and Contractor (Long Term).

What information is must be configured on the User record in the application after the user is added? If only a small number of users have a variation from other users, it may be that the easiest way to deal with those variations is to simply update those user records manually. Using the above examples,

- If your organization covers 2 time zones but only a small group of people work in one of the time zones whereas the bulk of the users are in the other time zone, the simplest procedure may be to define the template users for the main time zone and use that for the creation of all users. Then for the small group of users in the separate time zone, navigate to the User page to adjust the time zone after the record is added.
- If only a small number of Mobile Workers are supervisors with separate privileges, rather than defining a special template user for those type of workers, the simpler procedure may be to use the Mobile Worker template and then navigate to the User page to add the additional privileges to the supervisor users after the record is added.

To create a template user, navigate using **Admin > Security > User** .

- Define a User ID that will become the template user reference in the identity management system.
- Be sure to choose a **User Type** of **Template User**.
- Define all the information that should be copied onto a new user that references this user as a template user. Note that **Bookmarks** are not included in the data that is copied from a template user.

---

**NOTE:** There is configuration needed in Oracle Identity Management to capture the template user and any other information that the implementation has chosen to define in the identity management product when provisioning a new user. Refer to the *Identity Management Suite Integration* white paper for more information.

---

## Data Synchronization

---

**NOTE:** The information in this section applies to the integration with Oracle Utilities Meter Data Management and/or Oracle Utilities Network Management System.

---

## Defining Sync Requests

This section highlights functionality provided to help an implementation manage the communication of data to an external system.

## The Big Picture of Sync Requests

Your implementation may need to communicate certain data to external systems. This may be part of a data warehousing requirement or an integration effort. The synchronization process has two main parts. First, the change to the data must be detected and captured. Once that is accomplished, the next step is to manage the communication of that change to the external systems involved. The changes must be captured in chronological order so as to avoid systems going out of sync.

## Capturing the Change

The base product has provided the **Audit** plug-in spot on the MO to allow for processing to be called by the system upon detecting a change to an MO. The framework calls the algorithm defined on this plug-in spot in the event a change to the MO has been detected. The base product comes with a generic change data capture algorithm **F1-GCHG-CDCP** that makes use of this plug-in spot to create the sync request BOs. The **Sync Request BO** MO Option is used to define these sync request BOs. The generic algorithm loops through this list of BOs and instantiates them, if no instances of those BOs yet exist in the initial state. The MO and primary key used for these sync requests will be the MO being modified and its primary key.

Sometimes it is necessary to instantiate a sync request whose MO is different from the MO being modified. As an example, a change to the Person record may need to instantiate an SA sync request. In this case, your implementation will have to create your own Audit algorithm that captures this idiosyncratic logic. For examples on how this is done, please refer to the *Oracle Utilities CCB-MDM Integration Implementation Guide*.

## Sync Request Maintenance Object

The sync request maintenance object (MO) is used to communicate data elements to an external system. It holds the data elements that are to be synchronized or extracted out of your system and also holds the MO and primary key constituents that drive the sync request. If your organization wishes to use this MO, you can either use the business object (BO) supplied in the base product or configure your own BO.

This MO provides the following functionality:

- A business object option **External System** is provided for business objects of this type. The value specified on this option is used by the base product to create an entry in the JMS Queue.
- A business object option **Outbound Message Type** is provided for business objects of this type. The value specified on this option is used by the base product to create an entry in the JMS Queue.
- An **Information** algorithm is provided (**F1-SYNC-INFO**).
- The standard **Determine BO** algorithm is provided.
- The standard logs are provided.
- The standard characteristics collection is provided.

## Where Used

Follow this link to open the data dictionary where you can view the tables that reference F1-SYNC REQ.

For more information about this MO and to review the business objects defined for this MO, navigate to **Admin > System > Maintenance Object** and view the MO **F1-SYNC REQ**.



## Sync Request Business Object

The sync request business object (BO) contains the rules that govern the processing of a sync request. The algorithms in this BO control what data elements are synchronized to an external system; how this data is to be communicated to the external system; and how the communication of this information is controlled chronologically to ensure that changes are processed in the right order. The base product provides the BO **F1-SyncRequest**, which contains the following life cycle.

- **Pending.** The initial state of the sync request BO. At this point, a pre-processing algorithm would have already taken a snapshot of what the relevant information looked like prior to the change. A deferred monitor algorithm is specified on this state so as to accumulate all changes prior to sending the sync request to the external system. Furthermore, in order to prevent the sync request from transitioning to the next state while another sync request may not have completed its processing yet, the base package provides the monitor algorithm **F1-WAITRELSR** that skips transitioning if it detects another instance of the same BO in a non-final state for the same MO and primary key combination.
- **Determine If Sync Needed.** This transitory state will contain the algorithm that takes the final snapshot of the relevant information so that it may be compared against the initial snapshot to see if the change affected any of the relevant data elements. The base product provides the algorithm **F1-COMPSNAPS** for this purpose. Your implementation may also add your own rules here to discard the sync request if certain criteria are met.
- **Discarded.** If the comparison of the initial and final snapshots yields no changes, or if it fails any other checks your implementation has introduced, the sync request is transitioned to this state. If any processes are dependent on the sync request transitioning to a final state, your implementation may plug in an enter algorithm on this state to kick off the dependent process.
- **Send Request.** This transitory state is used to hold the algorithm that facilitates the sending of the sync request to the external system. The algorithm used here may write a message to a JMS Queue or write a record to the general process MO or perform any other task that is suited to how your implementation communicates the change.
- **Awaiting Acknowledgement.** This state can be used to hold the sync request from further state transitions until an acknowledgement is received from the external system. We strongly recommend designing some form of acknowledgement that the external system has processed the information as this helps control the chronological flow of information. The base package provides the business service **F1-UpdateSyncRequest** that transitions the sync request to either the next default state (in this case the Synchronized state) if a positive acknowledgement is received; or the state associated with the Rejection transition condition (in this case the Error state) if a negative acknowledgement is received.
- **Synchronized.** Once the information has been processed and acknowledged by the external system, the sync request is transitioned to this state. If any processes are dependent on the sync request processing successfully, your implementation may want to plug in an enter algorithm on this state to kick off the dependent process.
- **Error.** If a negative acknowledgement is received or if the acknowledgement is not received within the prescribed period of time, the sync request is transitioned to this state. Your implementation may wish to create a to do entry when the sync request transitions to this state. The base product provides the enter algorithm **F1-TDCREATE** for this purpose. The exit algorithm **F1-TODOCOMPL** has also been provided to complete any to do entries when leaving this state.
- **Canceled.** A user may transition the sync request to this state from either the **Awaiting Acknowledgement** state or the **Error** state if he wishes to discontinue processing of the sync request for any reason. Your implementation will have to configure your own cancel reasons to document the transition. If any processes need to be triggered when the sync request reaches this state, your implementation can plug in an enter algorithm containing such logic.

## Designing Your Sync Requests

The starting point for designing sync requests is the analysis of the external system's data requirements. Your implementation will need to analyze the external system's data model and compare it against your system's data model. Where the relevant data reside in your system will need to be identified. If the data reside in more than one MO, a decision must be made to either combine the information in one sync request or to break up the sync request into several ones.

For each sync request identified, the driving MO must be determined. The BO for each sync request can then be created as a sub-class of the base BO **F1-SyncRequest**. For an example of how these BOs are configured, please refer to *Oracle Utilities CCB-MDM Integration Implementation Guide* or *Oracle Utilities CCB-NMS Integration Implementation Guide*.

## Sync Requests

The Sync Request portal is used to view information about data synchronizations from Oracle Utilities Customer Care and Billing to Oracle Utilities Meter Data Management, or from Oracle Utilities Customer Care and Billing to Oracle Utilities Network Management System. The menu location of the portal depends on your specific edge product.

### Sync Request Query

Use the [query portal](#) to search for an existing Sync Request. Once a Sync Request record is selected, you are brought to the maintenance portal to view and maintain the selected record.

### Sync Request Portal

This portal appears when a sync request has been selected from the Sync Request Query portal.

The topics in this section describe the base-package zones that appear on this portal.

#### Actions

The actions zone only appears for sync request business objects that define an actions map. For sync requests that are designed to show the actions directly in the display map, this zone is suppressed.

#### Sync Request

The Sync Request zone shows display-only information about the sync request. Refer to the zone's help text for additional information about this zone's fields.

### Sync Request - Log

The sync request log page contains a standard [Log Zone](#).

## Configuration Migration Assistant (CMA)

---

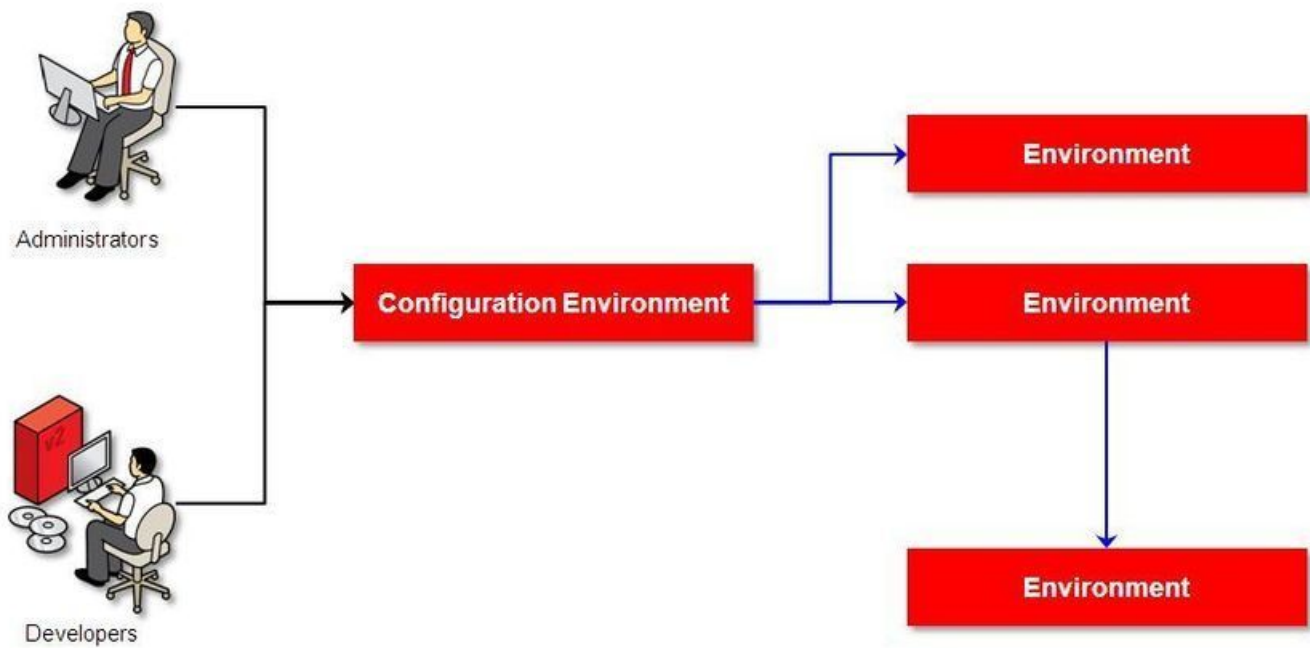
This chapter describes the Configuration Migration Assistant (CMA), a facility to enable the export of customer-owned configuration data from one environment to another.

**CAUTION:** This chapter is intended for users responsible for testing configuration data and performing "what if" analysis in non-production databases.

---

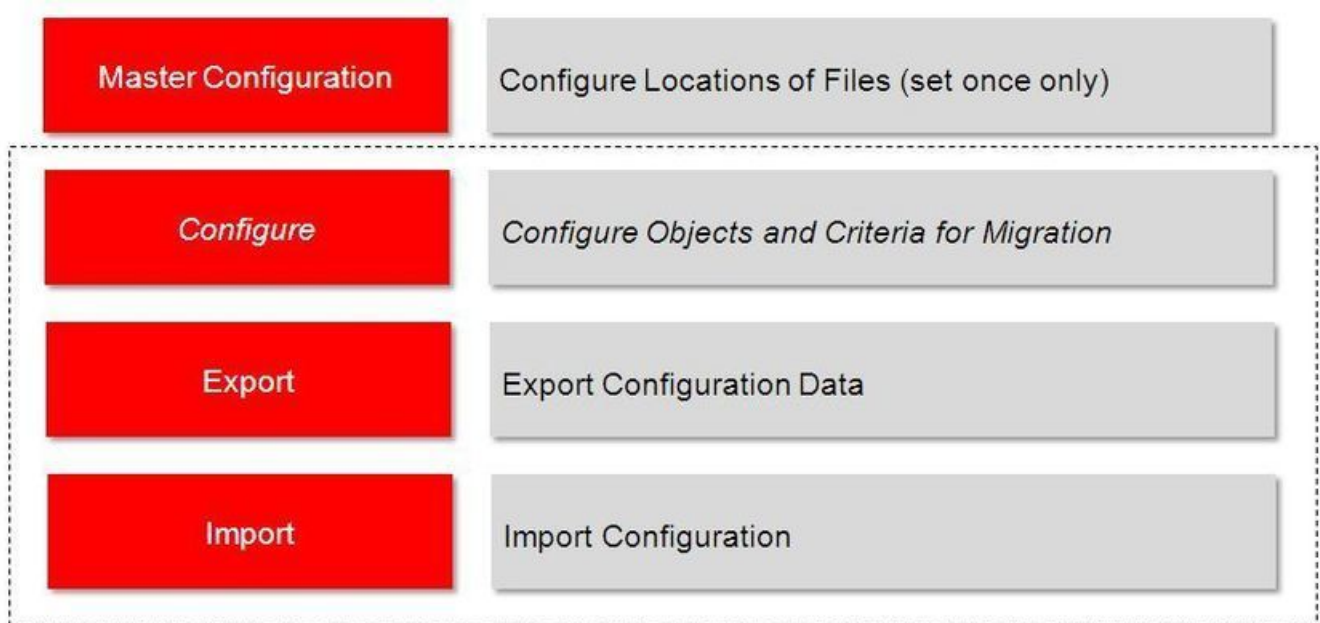
## Understanding CMA

The Configuration Migration Assistant (CMA) provides implementers with a flexible, extensible facility for migrating configuration data from one environment to another (e.g., from a development environment to a production environment). Data is exported from the source system to a file. The file can then be checked in to a version control system for reuse, or can be immediately imported into the target system and applied.



**NOTE:** As used in this chapter, *source* systems are those on which export-related activities are conducted and *target* systems are those on which migration updates are to occur. The two system preparation tasks described in [Migration Assistant Configuration](#) must be performed on both source and target systems.

## The CMA Process Flow



The high-level CMA process comprises the following tasks and flows. Each is described in more detail later in this section.

- **One-time tasks**, which must be applied on both source and target systems:

- Specify the source and target paths and a file extension for import and export data files in the **Migration Assistant Configuration** master configuration record.
- Create a thread pool that turns off the L2 caching. This is required to support the import step where cached data is being added or updated. It's recommended for all steps (including export and compare) to ensure that the algorithms are working with the data in the database rather than the cache. Refer to [Caching Considerations](#) for more information.
- **Configuration** steps are used to define the data to migrate. This task is performed on the source system and may be defined at any time. Note that the products provide base delivered configuration that may be used as is or used as a template for building more specific configuration for a given implementation.
- Each type of record that may be copied requires a **Migration Plan**. The migration plan is used to identify the maintenance object (MO) for the record (using a business object) and allows for instructions to specify related records that may be included in the migration. Multiple migration plans may exist for a given maintenance object if there are different requirements for migrating records in that MO under different circumstances.
  - The primary instruction in the migration plan could define the physical BO of the record. This signals to CMA that all records for the MO are eligible to be migrated.
  - The primary instruction may define a specific BO, in which case only records that reference that BO in its parental hierarchy are considered.

---

**NOTE:** Refer to [Understanding the BO Filtering Process](#) for more information.

---

- Subsequent instructions may include references to related records. There may be some circumstances where a migration should include subsequent records and some circumstances where they shouldn't based on requirements.
- A **Migration Request** is used to define a set of migration plans to be included together in a single migration attempt. In addition, the migration request may define selection criteria for each migration plan to limit the migration to a subset of data for each MO. Selection may be defined using SQL, an algorithm or explicit primary key values.

---

**FASTPATH:** For more information, refer to [CMA Configuration](#).

---

- The **Export** process includes all the steps needed to select records to be exported from the source environment and create the export file.
  - A **Migration Data Set Export** object is created for a specific migration request.
  - The lifecycle of the Migration Data Set Export business object includes algorithms that select the appropriate records according to the migration request, determine dependencies between records to build groupings of related objects and create the export file.
  - Because there may be a large volume of data in the export, many of the steps in the lifecycle of the migration data set export are executed via the **Migration Data Set Monitor**.

---

**FASTPATH:** For more information, refer to [Exporting a Migration](#).

---

- The file created by the export, which is a BINARY file, needs to be transferred from the export directory to the import directory. The transfer needs to be done in such a way as to preserve the file structure. Refer to [Migration Assumptions, Restrictions and Recommendations](#) for more information.
- The **Import** processes include all the steps needed to read an imported file, compare the data in the file to the data in the target, review the proposed changes and apply the updates. The following is a high level overview of the process.
  - A **Migration Data Set Import** object is created for a given file to import.

- The next step reads the import file and creates **Migration Transactions** and **Migration Objects** based on the information in the import file. A migration object is created for every maintenance object record to potentially be created or updated. The migration transaction is a grouping record that groups together related migration objects.
- The next step is for the objects to **Compare** the data being imported against the data for that record in the target environment. If it is found that the migration object is new or represents a change to the existing record in the target environment, appropriate SQLs are generated. If no changes are found, the object is marked “unchanged” and doesn’t progress.
- Once all the objects are compared, the user may review the objects for acceptance or rejection.
- When the migration objects are all accepted or rejected, the next step is to apply the objects and update the target environment.

---

**FASTPATH:** For more information, refer to [Importing and Applying a Migration](#).

---

## Migration Assumptions, Restrictions and Recommendations

The following sections describe miscellaneous topics that are important to learn with respect to CMA.

### Type of Data Migrated

CMA is designed to migrate configuration data only. The tool cannot be used to migrate master or transactional data that contains system-generated primary keys.

The comparison step of the import process will generate appropriate insert or update SQL statements for the following data found in the export:

- Configuration data in a maintenance object with no owner flag. This is purely implementation data.
- Configuration data in a maintenance object with owner flag, where the owner is **Customer Modification**. For example, implementer-specific business objects.
- Configuration data in a maintenance object with owner flag, where the main record is owned by the product but where a child record exists with an owner of **Customer Modification**. For example, implementer-specific algorithms added to a base owned business object.
- Customizable fields in a record that is owned by the product. For example, the priority of a based owned To Do type.

### No Record Deletion

The CMA process allows users to define records to copy from a source environment to a target environment. In that way, the import process for the migrated records is able to identify objects to add and objects to change. There is no mechanism for indicating that records in the target environment should be deleted. The absence of those records in the import is not enough because the migration may be only importing a subset to add or update. If data on the target system must be deleted, users must delete the records in the target accordingly.

Note that CMA does support the deletion of child rows of an object as a result of a comparison. This is only applicable to child records that are owned by the implementation.

### File Transfer Considerations

When moving the export file between systems, use the binary transfer option of whatever tool you use to move the file so that line-end characters are not converted from Linux-style to Windows-style or vice versa.

It is recommended to avoid using ‘.txt’ for the export file’s extension (defined in the [master configuration](#)). That file extension by default implies a non-binary file and tools that perform file transfer may treat this as a non-binary file unless explicitly stated. The recommendation is to define ‘.cma’ as the extension. This is not a recognized file extension and most file transfer tools will transfer the file as is.

Note that if the file gets converted, there are two likely outcomes - either a numeric conversion error, or a buffer under-run error may be received when attempting to import the file.

## CMA Configuration

The following sections describe tasks required for CMA configuration.

### Master Configuration - Migration Assistant

Before proceeding with your first migration, system wide configuration must be defined. This is captured in the **Migration Assistant Configuration** master configuration record. This must be defined in both the source environment and the target environment.

Navigate using **Admin > Master Configuration**. In the **Master Configuration** list, scroll to **Migration Assistant Configuration** and click the add icon if there is no record yet or the edit icon to modify an existing record.

For more information about specific fields in the master configuration, refer to the in-line help.

---

**NOTE:** This record can be updated at any time to change details. The new configuration takes effect on all subsequent exports and imports.

---

### Migration Plans

A migration plan defines one or more types of objects that are eligible for migration. It is essentially a set of instructions describing how the data to be exported is structured, allowing objects to be migrated together as a logical unit to ensure consistency and completeness.

The following topics describe defining a migration plan as well as other topics for a migration plan.

### Defining a Migration Plan

To view or define a migration plan, navigate using **Admin > Migration > Migration Plan**.

Use the **Migration Plan Query** portal to search for an existing migration plan. Once a migration plan is selected, you are brought to the maintenance portal to view and maintain the selected record.

---

**CAUTION:** Important! If you introduce a new migration plan, carefully consider its naming convention. Refer to [System Data Naming Convention](#) for more information.

---

The following points provide information about defining **Instructions** for a migration plan.

The **Instruction Sequence** uniquely identifies the instruction. The recommendation is to use increments of 10 to allow insertion of other instructions in the future.

Select **Primary** for the first **Instruction Type**. All migration plans must contain one and only one primary instruction. All subsequent instructions require a **Subordinate** instruction type. In this case, the **Parent Instruction Sequence** must be entered. This number, used to maintain the defined relationships in the exported data, must match an instruction sequence number at a higher level in the hierarchy.

The instruction **Description** provides a business description of the instruction.

Select a **Business Object (BO)** to define the type of object from which data will be derived.

---

**NOTE:** Though BOs are specified in each instruction, it's important to understand that each BO is used only for filtering purposes. The migrated data set comprises the complete contents of the *maintenance object* that the business object structure is defined against. For a more detailed explanation of this, see [Understanding the BO Filtering Process](#).

---

**NOTE:** Refer to [Identifying Tables to Exclude From Migrations](#) for information about defining child tables to always exclude from a migration.

---

**Traversal Criteria** is used to define the relationship between each of the objects in a migration plan. The system provides three options to define how the child object is connected to the parent object so the system knows how to traverse from one object to another. **Traversal Criteria Type** options are **Constraint**, **SQL** and **XPath**. The following points explain each option:

- **Constraint** allows you to select a table constraint that represents a given record's relationship to another record in the system via a foreign key constraint defined in the meta-data. If **Constraint** is selected, the following additional fields are enabled:
  - **Constraint ID** is a unique identifier for the constraint. The search will show the valid table constraints for the MO of the instruction's BO and the MO of the parent instruction's BO.
  - **Constraint Owner** is used to define the owner of the constraint. This is populated automatically when selecting a constraint from the search.
- **SQL** lets you specify SQL join criteria between the parent instruction's object and the child object in the **SQL Traversal Criteria**. The syntax of the the traversal criteria is a WHERE clause (without including the word WHERE). When referring to a field on the parent instruction's object, use the syntax `#PARENT.TABLE_NAME.FIELD_NAME`. When referring to a field on the current instruction's object, use the syntax `#THIS.TABLE_NAME.FIELD_NAME`. For example, the following statement is used on a migration plan for Business Object, where the parent instruction is the BO and the subordinate instruction is used to reference the UI Map that is referred to as a BO option with the option type "F1DU":  
`#PARENT.F1_BUS_OBJ_OPT.BUS_OBJ_OPT_FLG = 'F1DU' AND @trim(#THIS.F1_MAP.MAP_CD) = @trim(#PARENT.F1_BUS_OBJ_OPT.BUS_OBJ_OPT_VAL)`.
- The **XPath** option lets you apply syntax in an XPath expression referencing elements in the instructions' referenced business objects. This is entered in the **XPath Traversal Criteria**. For example, the display map collection statement in the SQL example noted above would be written as follows in XPath: `#this/mapCd = #parent/businessObjectOption/businessObjectOptionValue AND #parent/businessObjectOption/businessObjectOptionType = 'F1DU'`. This technique allows foreign key references that are mapped inside a CLOB to be referenced.

---

**NOTE:** The `#parent` expressions may access elements that are stored in the CLOB and described using `mapXML` and `mdField`. However, the `#this` expressions must refer to fields available in the business object using the `mapField` reference.

---

Defining **Next Migration Plan** provides the ability to indicate that in addition to copying the object defined in the instruction, any additional instructions included in that referenced migration plan will also be included in an export.

The **Algorithms** grid contains algorithms associated with each instruction. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the Sequence in which they should execute.

System Event	Optional / Required	Description
Import	Optional	Algorithms of this type may be used to adjust the data after it is moved to the target system.



System Event	Optional / Required	Description
		Refer to <a href="#">Adjusting Imported Data</a> for more information. Click <a href="#">here</a> to see the algorithm types available for this system event.

---

## Where Used

Follow this link to open the data dictionary where you can view the tables that reference [F1\\_MIGR\\_PLAN](#).

## Understanding the BO Filtering Process

Migration plan instructions require the definition of a business object to provide CMA with information about the record related to the instruction.

If the business object is the physical business object for the maintenance object, then CMA assumes that the instruction applies to all records that satisfy the traversal criteria. CMA recognizes the physical BO by comparing the BO to the value defined in the maintenance object option. If the business object defined is not the physical BO, then CMA will limit the records in the instruction to those that explicitly reference this BO or reference a child of this BO as its [identifying BO](#) value. (In other words, this BO must be in the parentage hierarchy of the records to be included in the instruction.)

---

**NOTE:** Unlike Bundling, CMA does not use the BO schema to drive what data is copied for a given record. The BO is only used as a filtering mechanism for selecting records. Refer to [Identifying Tables to Exclude from Migration](#) for information about how to ensure a child table is not included in a migration.

---

For example, if you define a migration plan for Master Configuration and use the physical business object for the instruction (**F1-MstCfgPhysicalBO**) then all master configuration records are considered for the instruction. If instead the business object you define is Migration Assistant Configuration (**F1-MigrationAssistantConfig**) then only the record related to this business object is included in the instruction.

## Migration Plans for Objects with CLOB-Embedded Links

When migrating objects where foreign key references are captured in the object's XML based field, subordinate instructions are needed to define the foreign key references in order for CMA to understand the relationships. This is in contrast to direct foreign keys where CMA can determine the relationships using constraints. The instructions provide two purposes. For wholesale migrations, where all data (or a large amount of data) is being migrated, the instructions allow CMA to group related objects into transactions. This helps in the apply process at import time to ensure that related objects are grouped together. However, the apply process includes iterative steps to try to overcome dependencies like this so defining the instructions is not critical for this type of migration. For piecemeal migrations, defining instructions ensures that the related objects are included in the migration, if appropriate.

The following are options for creating migration plans with XML-embedded links:

- One option is to use the specific logical (business) BO in the primary instruction to define the object you are copying. With this option, the subordinate instructions may use XPath criteria to define the related foreign key. When this approach is used, a separate Migration Plan must be created for each logical BO. (Refer to [Understanding the BO Filtering Process](#) for more information.) This option would only be used in isolated cases.
- Another option is to create a migration plan that uses the Physical BO as the primary instruction, and then include a subordinate instruction for the real logical BO, using SQL Traversal to join the object to itself by its primary key. Note that with this technique, the records that reference the logical BO will still only be included in the export file once. At this point further subordinate instructions may use XPath notation to define the foreign key data. Using the physical BO as the primary instruction ensures that all records in the MO are considered. The subordinate instructions with the logical BO and XPath notations will only apply to the records that are applicable to that BO. This option is useful for MOs that have a small number of logical business objects with disparate foreign keys.



- Another option is to use the physical BO in the primary instruction and use raw SQLs in the subordinate instruction's traversal criteria to identify the foreign keys using substring commands. A separate Subordinate Instruction is needed for each SQL corresponding to each element occurrence. Using this technique has the same advantages of the previous in that all records for the MO are included in the migration. However, this technique may be useful for maintenance objects with a larger number of business objects expected where each has one or more foreign keys. It's especially useful if many business objects reference the same foreign key. Then only one instruction is required for that foreign key. Note that a single migration plan may use this technique and the XPath technique for different elements.

A migration request may have multiple migration plans for the same maintenance object. That allows for some flexibility and long term maintainability in that the above techniques may be used in multiple migration plans. Consider the following example:

- A product provides base business objects with foreign keys defined in the XML field and provides the appropriate migration plan with instructions. An implementation extends this business object or perhaps creates their own business object for the same maintenance object and includes different additional foreign keys in the XML. Rather than duplicating the base migration plan and adding additional instructions for the additional foreign keys, the implementation can create a second migration plan for the MO with the additional foreign keys defined. A migration request should be defined to include both migration plans. In this case if the implementation has only one custom BO, they can choose to use the custom BO as the primary instruction as described above in the first option.

## Defining a Migration Request

Migration Requests are unordered lists of Migration Plans that are to be migrated together. Algorithms, SQL statements, or specific keys are used to specify the set of objects you want to export. When complete, the request describes the complete data set that is extracted to the migration export file when the request is executed.

To view or define a migration request, navigate using **Admin > Migration > Migration Request**.

Use the **Migration Request Query** portal to search for an existing migration request. Once a migration plan is selected, you are brought to the maintenance portal to view and maintain the selected record.

Define one or more **Migration Plans** for the migration request. Note that the order doesn't matter. During the export process, CMA looks at all objects that qualify for the export and will group them together based on their relationships.

For each option, define an appropriate **Selection Type**, which is used to filter which records in the migration plan's object should be selected. The valid values are **Algorithm**, **SQL Statement**, and **Specific Key**.

- For the option **Algorithm**, specify an **Algorithm** for the Key Selection. The algorithm is responsible for returning a list of key values to include in the migration.

---

**NOTE:** The algorithms that are eligible to be plugged in here are those valid for the Migration Request — Select system event. Click [here](#) to see the algorithm types available for this system event.

---

- For the option **SQL Statement**, specify an **SQL Statement** for the key selection. may be used to limit the keys to include. To migrate all records in the table, the SQL statement "1=1" may be used. Refer to the inline help for other examples.
- For the option **Specific Key**, enter one or more primary keys to individually to define the records to include. Click the "+" to enter each additional key. Each row represents one key and supports a multi-part primary key.

### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [FI\\_MIGR\\_REQ](#).

## Wholesale and Piecemeal Migrations

The Configuration Migration Assistant is used for two general types of migrations: wholesale and piecemeal. The following sections provide some additional information about these concepts.

## Wholesale Migrations

Wholesale migrations are used when migrating all the configuration and/or administrative data from one environment to another. For example, a wholesale migration might be used when migrating administrative data from a development or test environment to a production environment.

A wholesale migration may be comprised of one or more migration requests that in total include all the administrative data to move. Whether one migration request is used or multiple are used depends on the following considerations:

- For each migration request used, a separate migration data set export record (and therefore separate migration data set import record) is needed. The multiple records may cause more user interaction with the data to progress all the records to their final state.
- On the other hand, splitting data into multiple migration requests, grouping information logically together may allow for more reuse.
- In addition, you should consider that the framework product provides base migration requests and your specific edge product may provide base migration requests as well that may or may not include framework migration plans. Using the product provided migration requests is beneficial with respect to maintenance. As features are added to the product (including new administrative maintenance objects), any impact on CMA should be included in the product owned migration requests. If your implementation introduces new custom administrative maintenance objects that should be included in CMA, then custom migration plans and a custom migration request should be added. Your implementation may choose to include only migration plans for your custom MOs (and simply migrate the objects in this table as a separate step in the process) or include other migration plans from the product in your custom migration plan to have a consolidated plan.

Migration plans used in wholesale migrations may be designed to omit subordinate instructions related to explicit foreign keys that are identified through constraints as they are not needed, assuming that the data they are referring to will also be included in the migration.

---

**NOTE:** Refer to [Framework Provided Migration Objects](#) for information about migration requests provided in the framework product. Refer to your specific product's documentation for information about addition base provided migration requests.

---

## Piecemeal Migrations

A piecemeal migration refers to migrating a targeted subset of data from one environment to another. Examples of piecemeal migrations include:

- Migration of a new Business objects with its options and algorithms.
- Migration of a new maintenance portal, its zones, and its application service.
- Migration of all outbound message types.

Administrative users can define a migration request with the specific types of objects that should be copied and use selection criteria to limit the records to copy as desired.

The migration plans included in a 'piecemeal' migration request should be reviewed carefully to verify what subordinate instructions are included. When copying a specific type of record, there may be related records that should be copied as well and other related records that should not be copied. For example, when copying a custom To Do Type, perhaps any algorithms it refers to should be copied (along with their algorithm types) and its message should be copied, but not its navigation option or its drill keys, which would probably refer to base values.

## Identifying Tables to Exclude From Migrations

Some maintenance objects that are eligible to be migrated may include child tables that should not be included in the migration. For example, if an object includes log tables, the entries in the log should reflect the actions on the object in that system, and will be different between the source system and the target system. If you have a custom Maintenance Object that includes tables you don't wish to migrate (such as a log table), use the **Non-Migrated Table** option on the MO to specify this table. All child records for this table will also be ignored during migration.

Another use case to consider is a child “many-to-many” table that connects two administrative objects and exists in the maintenance object of both tables. The child table may be in both MOs for convenience sake, but it may be that one MO is considered more of a “driver” object and the other more of a subordinate. If you are doing a targeted (piecemeal) migration where you want to copy a subset of objects, you may want to only copy the driver object and its children and their data but not their children. For example, in Oracle Public Sector Revenue Management, a Form Type includes a collection of valid Form Change Reasons and in turn the Form Change Reason refers to its Form Types. If an implementation wants to do a targeted migration of a form type and include all its related information, including form rules and form change reasons, it does not want the migration of a form change reason to in turn copy all its form types (and their data).

---

**NOTE:** The MO option must be set in both the Source and Target systems for a given MO.

---

## Configuring Custom Objects for Migration

During the implementation and extension of the product, new custom administrative maintenance objects may be introduced. If your implementation would like to migrate records in those maintenance objects using the Configuration Migration Assistant, additional steps must be performed, which are highlighted in the following sections.

### Physical Business Object

As described in [Understanding the BO Filtering Process](#), the migration plan requires a business object for its instruction. The business object is used to identify the records eligible for inclusion in the migration. Assuming your custom tables use one or more “logical” business objects for their processing, your implementation must decide if these business objects are appropriate for use by the migration plans, or if a physical BO is warranted. If so, create an appropriate physical BO.

### Review MO Option Configuration

The following points highlight maintenance object (MO) configuration that should be reviewed or updated to support CMA:

- If a physical BO was created (above), link it to the MO as an option using the appropriate option type.
- Be sure that your MO defines an appropriate [FK Reference](#) and includes an Option on the MO that identifies the FK Reference. This is used by various portals and zones for CMA when showing detail about records being imported into the target region.
- As described in [Identifying Tables to Exclude From Migrations](#), an MO option is used to identify child tables for an MO that should never be included in a migration. If your custom maintenance object includes a standard Log table, then the recommendation is to list that table as an excluded table. Depending on the specific design of the maintenance object, there may be other child tables to define.

### Characteristic Type Configuration

The CMA import process will attempt to create a log record for any administrative object that includes a log table. If your implementation has introduced any custom administrative tables that you plan to include in a migration request and it includes a log table, you must, to ensure that the log creation is successful, add your log table as a valid characteristic entity to the characteristic type **F1-MGO** (Migration Object).

Navigate using **Admin > Characteristic Type > Search** and select the characteristic type **F1-MGO**. Navigate to the **Characteristic Entity** tab and add a row to include the characteristic entity for your custom maintenance object's log table.

## Standard CMA Configuration

Create one or more migration plans for the new object, depending on the type of data in the maintenance object and the types of migrations you envision:

- If you have implemented only one “logical” business object used to define the data in the MO, then a single migration plan that references the this BO (or the maybe the MO’s physical BO) is appropriate.
- If you have implemented more than one “logical” business object, would the data for multiple business objects get copied together? Then perhaps a single migration plan that references the MO’s physical BO is appropriate.
- Are there additional foreign keys defined using mapXML in the business object(s) for the MO? If so, then it is recommended to include sub-instructions to define the links. At this point, if multiple “logical” BOs exist, your implementation may choose to define all the additional elements in the same migration plan or choose to define separate migration plans for each logical BO.
- Your implementation may decide to define more than one migration plan for the same type of record based on the types of migrations you plan to include. For example, you may decide to include a migration plan that copies only the records in this maintenance object. You may decide to define another migration plan that copies the records in this MO along with related records in another MO (for a special type of migration). Having said that, be sure to design the migration plan with reuse in mind to minimize maintenance efforts.

If your implementation has a template migration request to use for piecemeal or wholesale migrations, include the new migration plan(s) as appropriate.

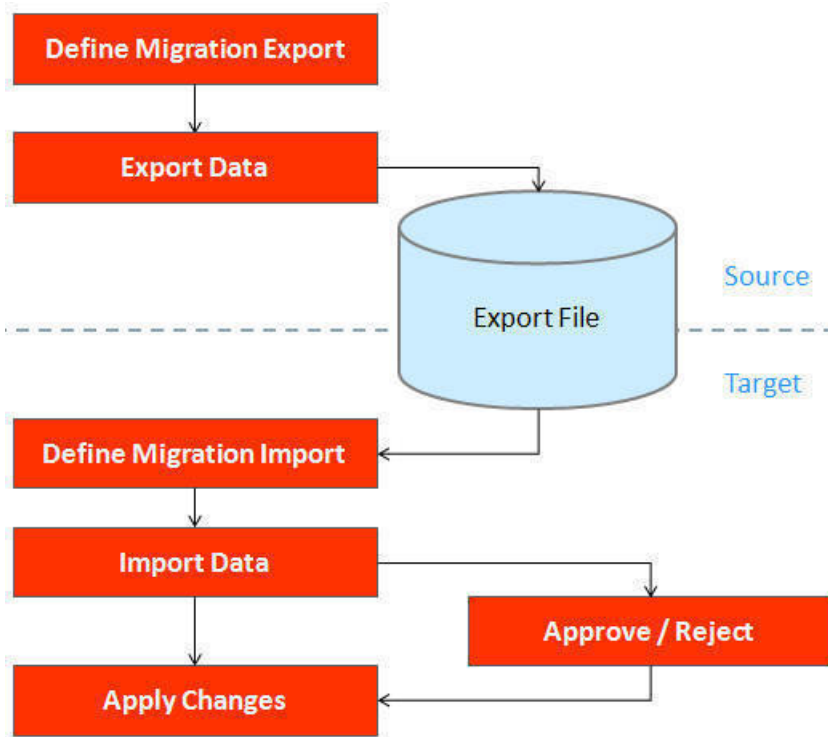
---

**CAUTION:** Important! New migration plans and migration requests should follow naming conventions. Refer to [System Data Naming Convention](#) for more information.

---

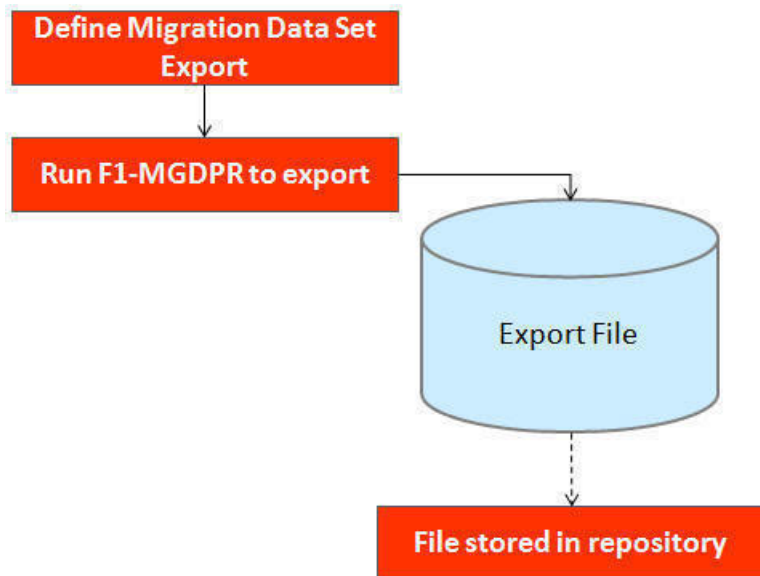
## The CMA Execution Process

The following diagram illustrates a high-level view of the Configuration Migration Assistant execution process. The subprocesses illustrated here are described in more detail in the following sections.



## Exporting a Migration

The migration export process begins in the source environment by defining a **Migration Data Set Export**, which specifies a defined **Migration Request** and provides a file name and description for the export file. After the data set is defined and saved, the **Migration Data Set Monitor** batch job can be submitted generate the export file.



The following topics provide more detail about this process.

## Migration Data Set Export

To migrate data from one region to another, define a **Migration Data Set Export**. This establishes the export file name and identifies the migration request that includes the migration plans and selection criteria for the objects to include in the migration.

To view an existing migration data set export, navigate using **Admin > Migration > Migration Data Set Export**. Use the query criteria to locate the desired data set.

Note that you can also initiate the creation of an export data set from the [Migration Request](#) portal using the **Export** button.

The export requires the name of an existing **Migration Request**.

Enter a unique **File Name** for the export. Do not use spaces in the name, and do not enter the file extension or a path. The output location and file extension of the intended export file, which should appear in the **Export Directory** and **File Suffix** labels, are defined in the [Migration Assistant Configuration](#) master configuration record.

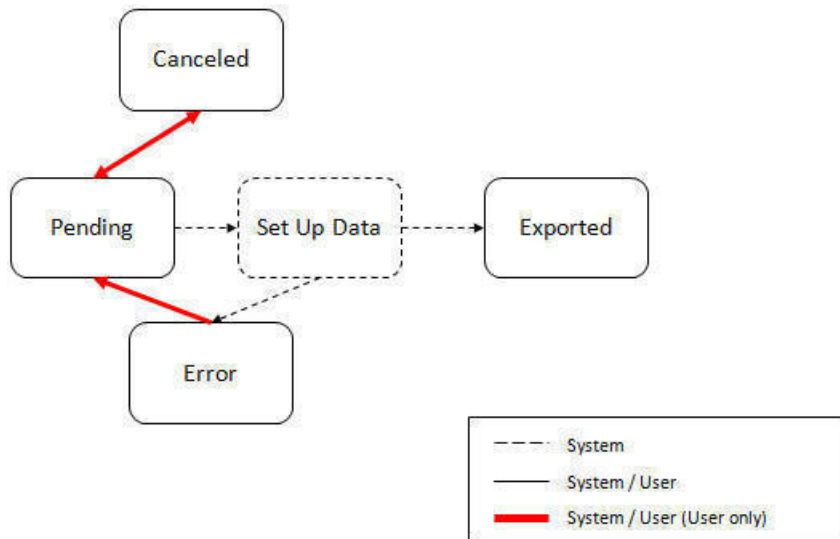
Enter an **Export Description** for the data set.

The **Source Environment Reference** is for information purposes. It should be populated with text that provides a meaningful description of the source environment. The default value is the URL of the source environment.

When the data set is complete, click **Save**.

## Export Lifecycle

The following diagram describes the lifecycle of a Migration Data Set Export (data set).



The following points describe the lifecycle.

- The data set is created in **Pending** status.
- A user may choose to temporarily **Cancel** a pending data set to prevent it from being processed. The user can later return it to the Pending state when desired.
- The record remains in Pending until its monitor batch job is submitted. The **Migration Data Set Export Monitor** (F1-MGDPR) selects pending records and transitions them to **Set up data**. Refer to [Running Batch Jobs](#) for more information.
- Set up data is a transitory state that includes the algorithm that does the work of determining the objects to include in the export and group related objects together into a transaction. If everything is successful, the export file is written to the

appropriate file location and the record transitions to **Exported**. If an error is detected, the process stops and the record transitions to **Error**.

- If a record is in error and it is possible to correct the error, the record may be transitioned back to **Pending** to try again.

When the process is marked as **Exported**, the export file can be imported into the target system.

---

**NOTE:** The export process creates a file, providing the benefits of having a standalone file. It can be stored in a version control system for audit purposes or provided to others for import purposes.

---

**CAUTION:** Under no circumstances should exported data files be edited manually. Doing so could cause data corruption when the file is applied to the target environment.

---

**NOTE:** The export functionality is supported using the business object **Migration Data Set Export** (F1-MigrDataSetExport). The expectation is that implementations will use the delivered base business object and its logic and will have no reason to implement a custom business object for the CMA export process.

---

## Importing and Applying a Migration

The import process is broken down into four general steps: Import, Compare, Approve, Apply. The following points provide an overview of the steps.

- **Import.** The first step covers importing the file and creating appropriate Migration Import records in the target environment to facilitate the subsequent steps.
- **Compare.** The compare step reviews each object that is in the import file and compares the object in the import against the equivalent record in the target environment. The comparison step results in noting which objects are unchanged, which are new (and the appropriate SQL to insert them) and which objects are changed (and the appropriate SQL to update them). Based on user configuration at import time, the objects that qualify for the import may be in a state that requires review or may be pre-approved.
- **Approve.** Once the comparison is complete, the user should review the results. There may be records marked for review. All of these records must be approved or rejected before the import can proceed. When the user is satisfied with the results of the comparison and has completed the review, the import is marked to proceed to the Apply step.
- **Apply.** This is the final step and is the step where the records in the target environment are added or updated. Because of potential high volumes of data and because of possible dependencies between records, this step supports two levels of attempting to apply the records. There is an apply step at the object level and an apply step at the transaction level. This will be described in more detail below.

## Import Step

The import process starts with verifying the import directory configured in the [Master Configuration — Migration Assistant](#) for the target environment and ensuring that the exported file is located in that directory. Then, in the target environment, a **Migration Data Set Import** record should be created. The user indicates the file name.

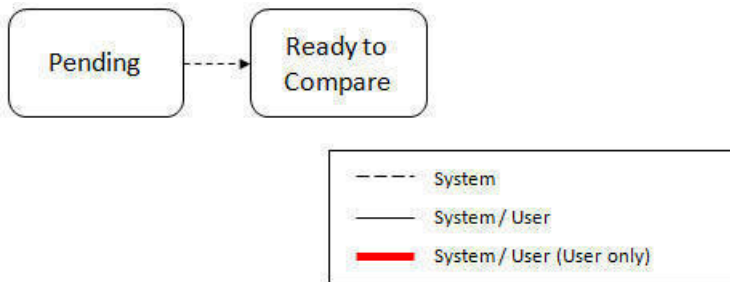
In addition, the user decides what the default status should be for resulting objects.

- The **Default Status for Add** sets the default status for objects that are determined to be *new* during the import comparison process. The default is to automatically set new objects to **Approved** status. Other options are to set any new objects to **Rejected** or **Needs Review** status.
- The **Default Status for Change** sets the default status for objects that are determined to be *changed* during the import comparison process. As with new objects, the default for changed objects is **Approved**, with **Rejected** or **Needs Review** options available.



The file to import contains a list of all the objects included in the export. Any objects that the export step determined to be related have been grouped into “transactions”. Once the Migration Data Set Import (data set) is created, the next step is to read in the file and create Migration Transactions and Migration Objects.

The following is a portion of the Migration Data Set Import lifecycle as it pertains to the import step.



The following points describe the lifecycle.

- The data set is created in **Pending** status.
- The record remains in Pending until its monitor batch job is submitted. The **Migration Data Set Import Monitor** (F1-MGDIM) selects pending records and transitions them to **Ready to Compare**. Refer to [Import Process Summary](#) for more information.

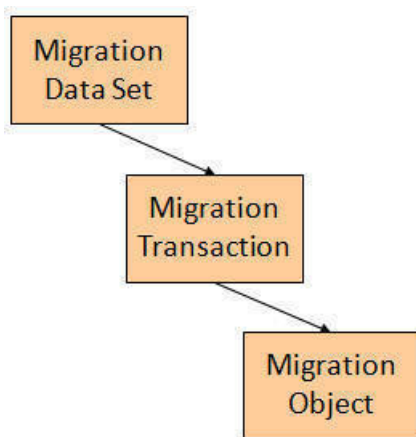
The Ready to Compare state has an algorithm that is responsible for reading the related import file and creating the migration transactions and migration objects. The data set remains in this state until the comparison step is complete.

---

**NOTE:** A user may choose to **Cancel** a data set. Refer to [Cancelling a Data Set](#) for more information.

---

The following diagram highlights the relationships of the resulting migration import records.



The migration transaction and migration object each have their own lifecycle that will help manage the subsequent compare and apply steps. At the end of the import step, the status values of the three types of records are as follows:

- Migration Data Set Import is in the **Ready to Compare** state.
- Migration Transaction is in the **Pending** state.
- Migration Object is in the **Pending** state.

---

**NOTE:** The import functionality is supported using business objects supplied by the base product. The expectation is that implementations will use the delivered base business objects and their logic and will have no reason to implement a custom business objects for the CMA import process. The base business objects are **Migration Data Set Import** (F1-MigrObjectImport), **Migration Transaction** (F1-MigrTransactionImport) and **Migration Object** (F1-MigrObjectImport).

---



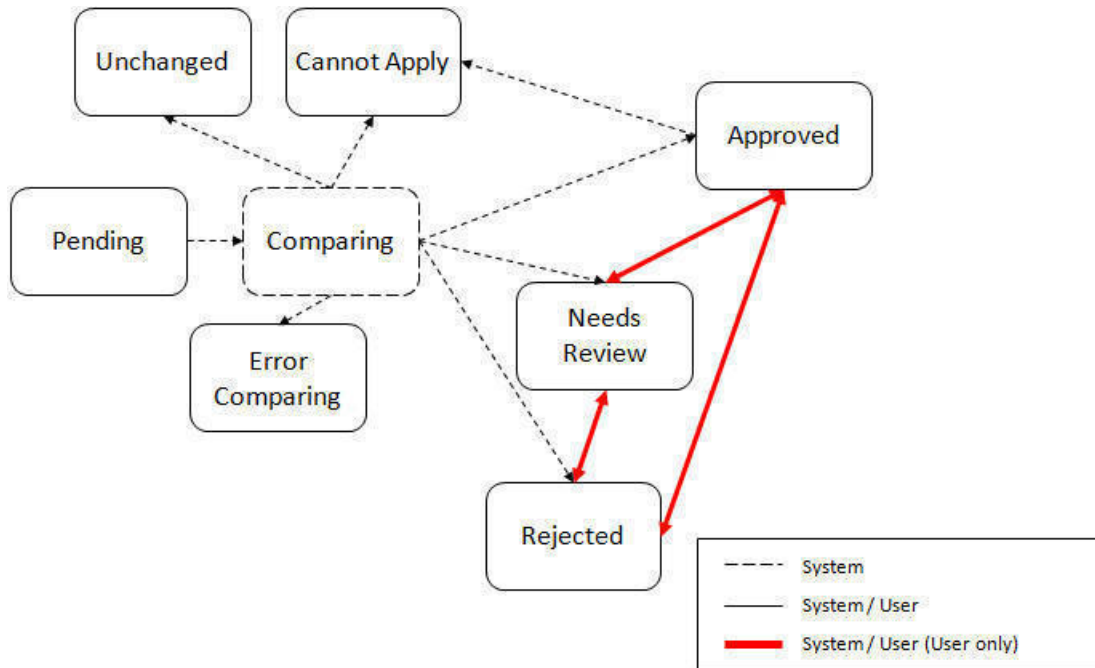
## Compare Step

The import step results in the creation of one or more migration objects, one for each record selected in the export based on the export's migration request and its configuration. Related objects are grouped together in migration transactions. The next step in the import process is the Comparison step. In this step, the data captured by the import file for each object is compared to the view of that object in the target environment.

To cater for a possible large volume of objects, the comparison is done via a batch monitor. To aide in performance of the process, the monitor is performed on the migration objects so that it can be run multi-threaded. Once the objects are finished with the comparison, the migration transactions and the migration data set should be updated with an appropriate overall status before continuing to the next step. As a result, the comparison actually requires three steps: Migration Object Comparison, Migration Transaction Status Update and Migration Data Set Export Status Update. The steps are explained in detail in the following sections.

### Migration Object Compare

This is the main step of the comparison. The **Migration Object Monitor** (F1-MGOPR) selects pending migration object records and transitions them to **Comparing**. This is a transitory state that includes the algorithm that does the work of comparing. There are various possible outcomes that could occur based on the logic in the algorithm. The following diagram illustrates a portion of the migration object lifecycle that pertains to comparison.



The following points describe the lifecycle.

- When **Pending** records are selected by the monitor batch job, it transitions to **Comparing** where the algorithm will determine the appropriate next state.
- If the record in the migration object is found in the target environment and the data is exactly the same, the record transitions to **Unchanged**.
- If the record in the migration object is found in the target environment and the data is different, the algorithm generates the appropriate SQL to be used later in the Apply step to update the record and then transitions to **Approved**, **Needs Review** or **Rejected** based on the Default Status For Change setting captured on the Data Set.

- If the record in the migration object is not found in the target environment, the algorithm generates the appropriate SQL to be used later in the Apply step to insert the record and then transitions to **Approved**, **Needs Review** or **Rejected** based on the Default Status For Add setting captured on the Data Set.
- If there is any issue with attempting to parse the object data from the import, the record transitions to **Error Comparing**.
- If there is any reason that the imported object is not valid for import, the record transitions to **Cannot Apply**. The log will be updated with the error that caused the record to transition to this state. An example is that perhaps the record was exported in a different version of the product and has additional elements that are not recognized in this version.

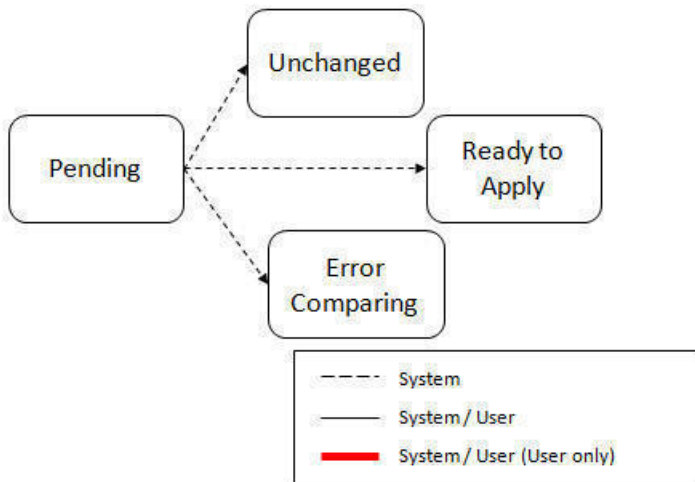
---

**NOTE:** Refer to [Cancelling a Data Set](#) for information about cancelling a data set and its impact on its related objects.

---

## Migration Transaction Status Update

After the import step, the migration transaction remains in the Pending state until all its objects have completed the comparison step. At that point, the status of the transactions should be updated based on the results of their objects. The **Migration Transaction Monitor** (F1-MGTPR) selects pending migration transaction records and runs its monitor algorithms. There are various possible outcomes that could occur based on the logic in the algorithms. The following diagram illustrates a portion of the migration transaction lifecycle that pertains to comparison.



The following points describe the lifecycle possible next states after Pending.

- If any related migration object is in the Error Comparing state, the transaction transitions to **Error Comparing**.
- If all related migration objects are in the Unchanged state, the transaction transitions to **Unchanged**.
- Otherwise, the transaction transitions to **Ready to Apply**. This means that at least one object is in an “apply-able” state.

The transaction remains in the **Ready to Apply** state until a user has approved the data set to move to the Apply step and the transaction’s related objects have attempted to apply themselves. This is described in more detail below.

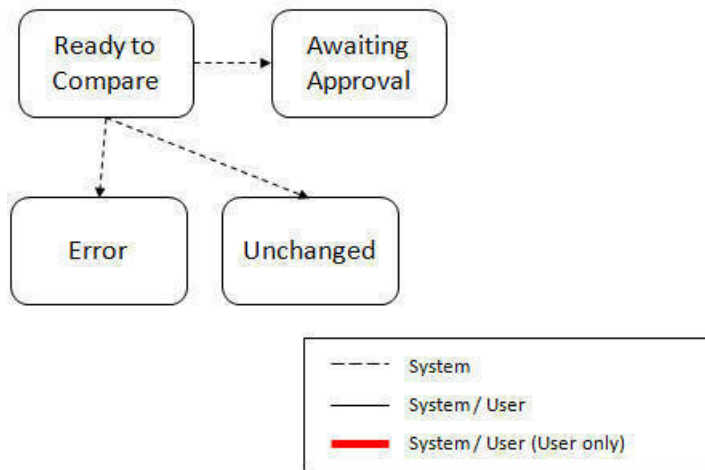
---

**NOTE:** Refer to [Cancelling a Data Set](#) for information about cancelling a data set and its impact on its related objects.

---

## Migration Data Set Import Status Update

Once all the objects and all transactions have been updated via the previous two steps, the migration data set export must be updated based on the results of their transactions. The **Migration Data Set Import Monitor** (F1-MGDIM) selects Ready to Compare data sets and runs its monitor algorithms. Note that this is the same monitor process that is used to select Pending data sets. There are various possible outcomes that could occur based on the logic in the algorithms. The following diagram illustrates the portion of the migration transaction lifecycle that pertains to comparison.



The following points describe the lifecycle possible next states after Ready to Compare.

- If any related migration transactions is in the Error Comparing state, the data set transitions to **Error**.
- If all related migration transactions are in the Unchanged state, the data set transitions to **Unchanged**.
- Otherwise, the transaction transitions to **Awaiting Approval**. This means that there are no errors and at least one object is in an “apply-able” state.

The data set remains in the **Awaiting Approval** state until a user decides that the data set and all its records are ready to progress to the Apply step.

---

**NOTE:** A user can choose to cancel a data set at any time while it is in progress. Refer to [Cancelling a Data Set](#) for more information.

---

## Approval Step

Once the comparison is complete and the data set transitions to the Awaiting Approval state, a user needs to progress the data set to **Apply Objects** to trigger the Apply step. The following points describe steps a user may take during the approval step.

- If the data set configuration for the Default State for Add and Change was set to **Approved**, then any migration object that is determined to be eligible for the Apply step will be in the Approved state. In this situation, a user may want to review the data set and its transactions and objects to see verify that the results make sense. At that time, the user is able to move an object to Needs Review or Rejected as appropriate.
- If the data set configuration for the Default State for Add and Change was set to **Needs Review** for either option, then each migration object in the Needs Review state must be reviewed and the user must either Reject or Approve each object before moving to the Apply step.
- If the data set configuration for the Default State for Add and Change was set to **Rejected** for either option, the assumption is that the rejected records don’t need to be reviewed. But if a user finds a rejected record that shouldn’t be rejected, it may be transitioned to Approved (or Needs Review) as appropriate.

Once the user is comfortable with the data set’s results and no more objects are in the Needs Review state, the user should transition the record to **Apply Objects**. This will initiate the Apply step.

---

**NOTE:** Refer to [Maintaining Import Data](#) for details about the pages provided to help the user review a data set and its transactions and objects to help in the approval step.

---



---

**NOTE:** A user can choose to cancel a data set at any time while it is in progress.

---

## Apply Step

The apply step is the step where records in the target environment are added or updated. Like the comparison step, the apply step is actually multiple steps to optimally handle high volume and dependencies between records as smoothly as possible. Before explaining the apply steps in detail, the following points highlight the type of data that may be included in a given data set.

1. Records that have no foreign keys and therefore no dependencies on other records. Examples: Message, Display Profile.
2. Records that have foreign keys that may already be in the target. Examples: Algorithms for existing algorithm types, To Do Roles for existing To Do Types.
3. Records that have foreign keys that are new records but also part of the migration. CMA detected the relationship at export time and grouped the objects in the same transaction. Example: Script-based Algorithm Type where the script is also in the migration.
4. Records that have foreign keys that are new records but also part of the migration. CMA did not detect the relationship. This may occur if the reference the foreign key is in a CLOB or parameter column and the migration plan did not include an instruction to explicitly define the relationship. Example, a Zone that references a visibility script.
5. Records that have circular references where both records are new and are part of the migration. CMA detected the relationship at export time and grouped the objects in the same transaction. Example: plug-in Script for a BO plug-in spot. The script references the BO and the BO references an algorithm for the script's algorithm type.

To handle high volume data, the first step in the apply process is to perform the apply logic at the migration object level via a multi-threaded batch job. This should result in all records in categories 1 and 2 above being applied successfully.

For records in categories 3 and 4 above, if a record with a foreign key is added or updated before its related record, the validation will fail. However, if the related record is added first and then the record referring to it is added, validation will pass. Because the migration objects are not ordered, the multi-threaded batch process may not process records in the desired order. To overcome this potential issue, the Apply step has special functionality, described in detail below.

For records in category 5 above, the circular reference will mean that the apply process at the object level will not successfully add or update these records. The apply process at the transaction level will cover these records. This is described in detail below.

## Apply Objects

Once the Data Set is in the state of **Apply Objects**, the **Migration Object Monitor — Apply** process (F1-MGOAP) runs to attempt to apply the objects. The background process in conjunction with the Apply algorithm have special functionality to ensure records in categories 3 and 4 (above) successfully apply during this step:

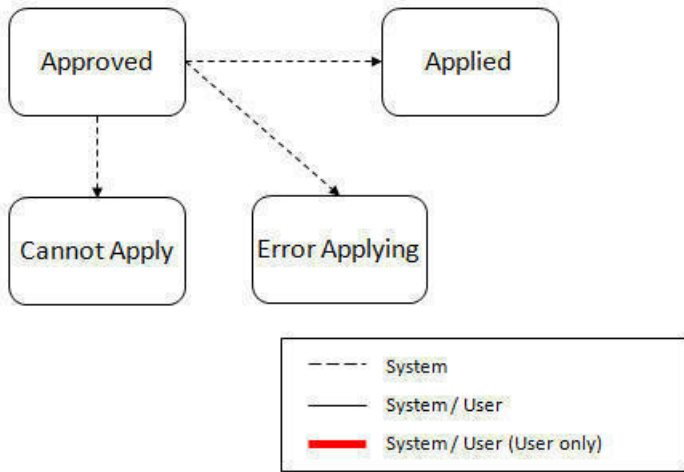
- The **Migration Object Monitor — Apply** process is a special one that continually re-selects records in the **Approved** state until there are no more eligible records to process.
- When an error is received in the Apply Object algorithm, the algorithm increments an “iteration count” on the migration object record. If the iteration count does not exceed a maximum count (noted in the algorithm), the object remains in the **Approved** state and is eligible to be picked up for processing again. If the iteration count exceeds the maximum defined in the algorithm, the record transitions to the **Error Applying** state.

---

**NOTE:** When submitting this Apply batch job, be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker. Doing this will cause the ‘excess’ threads to wait for the supported number of threads to finish, erasing the benefit of the iteration processing.

---

The following diagram is the portion of the migration object lifecycle that pertains to the Apply step. Note that this diagram is not complete. A subsequent section provides more information about resolving errors.



At the completion of the Apply monitor process, typically the objects will be in the **Applied** state or the **Error Applying** state. The records in the Error Applying state are in that state for one of two reasons.

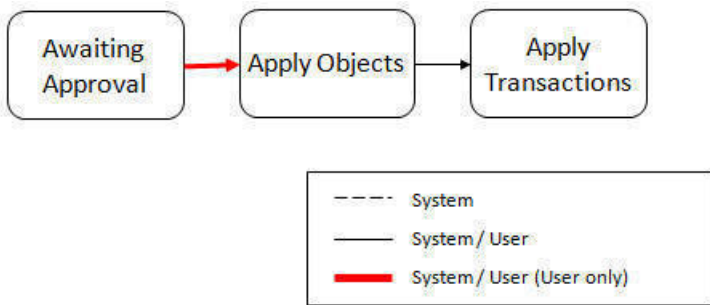
- They are in category 5 described above where the records have a circular reference with another record. For this scenario, the Apply Transactions step described below should successfully apply the records.
- There is some other error that is unrelated to the records in the current migration. In this case, manual intervention is required. Refer to the Resolving Errors section below for more information.

As shown in the diagram, the Apply Objects algorithm may also detect a reason that the object cannot be applied. This may occur if the object in the target environment has been updated since the comparison step, making the SQL captured at that point no longer applicable. If this occurs, after the current migration is fully applied, the original file may be imported again, and new comparisons can be generated and applied.

### Apply Transactions

Ideally, after the Apply Objects step, all the objects are **Applied** or are in **Error Applying** due to the “circular reference” situation. The typical next step is to turn over responsibility to the transactions. The migration transactions can then attempt to apply their objects in bulk.

In order to ensure that multiple background processes are not trying to select migration objects to run the Apply step, the transactions are only eligible to attempt to “apply my objects” if the Data Set is in the **Apply Transactions** state.



A monitor algorithm (executed by the data set monitor batch process) on the Apply Objects state checks to see if all migration objects are no longer **Approved** and do not have any records in **Error Applying**. If so, it automatically transitions the record to the **Apply Transactions** state.

If any objects are in **Error Applying**, the monitor algorithm does not automatically transition the record. In that case, a user must decide if the record should transition to **Apply Transactions**. The Resolving Errors section below describes alternative steps that the user may take if there are errors.

Once the Data Set is in the state of **Apply Transactions**, the **Migration Transaction Monitor** — **Apply** process (F1–MGTAP) runs. It attempts to apply the transaction’s objects. If no migration objects are in error, the migration transaction

simply transitions to **Applied**. If any of the migration objects are in **Error Applying**, the background process and the Apply algorithm have special functionality to try to overcome dependencies in migrated objects:

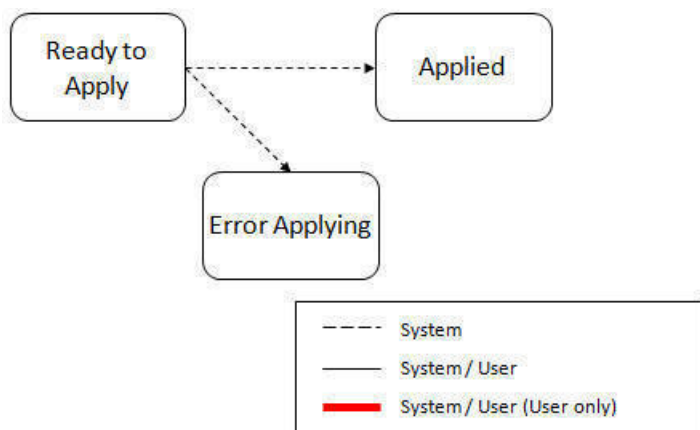
- The Apply algorithm selects all migration objects in error and performs all their SQL, then validates all the records. If there are objects in the transaction with circular references, they should pass validation at this point.
- Because there may still be some dependencies across transactions, similar error handling described in the Apply Objects step occurs here. When an error is received in the Apply Transaction's Object algorithm for any of the objects in the transaction, the algorithm increments an "iteration count" on the migration transaction record. If the iteration count does not exceed a maximum count (noted in the algorithm), the transaction remains in the **Ready to Apply** state and is eligible to be picked up for processing again. If the iteration count exceeds the maximum, the record transitions to the **Error Applying** state. Note that if any objects in the transaction are in error, none of the objects are applied. They all remain in error.
- The **Migration Transaction Monitor — Apply** process is a special one that continually re-selects records in the **Ready to Apply** state until there are no more eligible records to process.

---

**NOTE:** When submitting this Apply batch job, be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker. Doing this will cause the 'excess' threads to wait for the supported number of threads to finish, erasing the benefit of the iteration processing.

---

The following diagram is the portion of the migration transaction lifecycle that pertains to the Apply step illustrating the points above. Note that this diagram is not complete. A subsequent section provides more information about resolving errors.



If at the end of the transaction level Apply process there are transactions in error (and therefore there are still objects in error), a user must review the errors and determine how to fix them. Refer to the Resolving Errors section below for more information.

## Resolving Errors

As mentioned in the previous sections, errors may be received after the Apply Objects process runs and the system will not automatically transition the data set to the **Apply Transactions**. Rather, a user must make the decision. This allows for the user to review the errors and make a decision:

- If the errors appear to be dependency related, the user can decide to let the "transactions apply their objects" and transition the data set to **Apply Transactions**, described above.
- If the errors appear to be related to an outside issue that can be manually resolved, the user may choose to fix the issue and redo the Apply Objects step.
- The user may also decide to reject one or more objects to remove them from the migration.

After the Apply Transactions step, if there are still errors, a user must review the records and determine how to proceed

- The user may decide to reject one or more objects to remove them from the migration.
- The user may manually resolve an issue external to the migration and then decide to do one of the following:
  - Redo the **Apply Objects** step. This is recommended if there are a large number of Objects still in error and not a large number of dependencies expected. The benefits of running the Apply Objects multi-threaded will ensure that the process runs efficiently.
  - Redo the **Apply Transactions** step.

Because the objects and transactions are in Error Applying, in order to “retry” the Apply step after manually fixing an error, the system needs to move the records back to the state that allows them to be picked up by the appropriate Apply process. For migration objects, records need to be moved back to **Approved**. For migration transactions, records need to be moved back to **Ready to Apply**. This is accomplished with a Retry Count. This is different from the Iteration count that is used during the apply steps to handle the iterative selection logic. The following points describe the Retry logic for migration objects.

- When the Migration Data Set Import record first enters the **Apply Objects** state, its Object Retry Count is set to 1.
- When Migration Objects first enter the **Approved** state, their Object Retry Count is set to 1.
- When migration objects transition to **Error Applying**, it remains in this state as long as the object’s Retry Count matches the data set’s Object Retry Count (or it is successfully applied via the Apply Transactions step).
- If a user decides to **Retry Objects** (using an action button on the Migration Data Set Import page), the data set’s Object Retry Count is incremented and the Data set returns to the **Apply Objects** state.
- At this point, the monitor on the **Error Applying** state for the objects detects that the object’s Retry Count doesn’t match the data set’s Object Retry Count and that triggers the transition back to **Approved**. This increments the object’s retry count to match the data set’s. Now the objects are eligible to be picked up by the object level Apply process.

Analogous logic exists for the migration transactions.

- When the Migration Data Set Import record first enters the **Apply Transactions** state, its Transaction Retry Count is set to 1.
- When Migration Transactions first enter the **Ready to Apply** state, their Transaction Retry Count is set to 1.
- When migration transactions transition to **Error Applying**, it remains in this state as long as the transaction’s Retry Count matches the data set’s Transaction Retry Count.
- If a user decides to **Retry Transactions** (using an action button on the Migration Data Set Import page), the data set’s Transaction Retry Count is incremented and the Data Set returns to the **Apply Transactions** state.
- At this point, the monitor on the **Error Applying** state for the transactions detects that the transaction’s Retry Count doesn’t match the data set’s Transaction Retry Count and that triggers the transition back to **Ready to Apply**. This increments the transaction’s retry count to match the data set’s. Now the transactions are eligible to be picked up by the transaction level Apply process.

## Finalize Apply Step

Once all the migration objects for a migration transaction are in a final state (**Applied**, **Rejected** or **Cannot Apply**), the migration transaction transitions to the **Applied** state. Once all the migration transactions are in the **Applied** state, the Migration Data Set record transitions to the **Applied** and the import is complete.

---

**NOTE:** To review the full lifecycle for each record, refer to the Business Object — Summary tab in the application metadata for the base business objects **Migration Data Set Import** (F1-MigrObjectImport), **Migration Transaction** (F1-MigrTransactionImport) and **Migration Object** (F1-MigrObjectImport).

---

## Adjusting Data While Importing

Some records may have data that is specific to the environment it is in and won't apply in the target environment. In such cases, an algorithm plugged into the migration plan instruction may be used to adjust the data when importing. This algorithm is executed in the apply step after the record is inserted but before the validation is executed.

Some examples of records that may require import algorithms.

- Batch Control references its next batch sequence number. This number is only relevant with respect to its environment. The instruction for a batch control can include an algorithm to not overwrite the batch sequence number when copying a batch control.
- Some products include administrative objects that reference a master data object. Master data objects are not copied as part of CMA. An import algorithm may be used to adjust the referenced master data foreign key when importing, for example to reset it (or not overwrite when updating). If the algorithm knows how to find the appropriate master data record to link, that may also be included.

## Import Process Summary

The following table summarizes the steps required to complete the import process from start to finish. It highlights what steps are manual and what steps are performed by a batch monitor process. For the Apply steps, there are two parts where multiple next actions are possible. The possible next actions have the same sequence with a letter following the sequence. For each step, the table highlights the Next Action sequence that would occur.

**NOTE:** When running the Apply batch jobs, be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker.

Also note that a sequence and action marked in bold is considered the “normal path”.

Step	Seq	Action	Manual / Batch	Portal — Action Batch Control	Record Impacted — Resulting Status	Next Action Sequence
Import	<b>10</b>	<b>Create Import record</b>	Manual	Migration Data Set Import - Add	Migration Data Set Import - Pending	11
Import	<b>11</b>	<b>Import file</b>	Batch	F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Ready to Compare Migration Transaction - Pending Migration Object - Pending	20
Compare	<b>20</b>	<b>Migration Object Compare</b>	Batch	F1-MGOPR - Migration Object Monitor	Migration Object - Approved, Needs Review, Rejected, Unchanged or Error Comparing	21
Compare	<b>21</b>	<b>Migration Transaction Status Update</b>	Batch	F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply, Unchanged	22



Step	Seq	Action	Manual / Batch	Portal — Action	Batch Control	Record Impacted — Resulting Status	Next Action Sequence
						or Error Comparing	
Compare	22	<b>Migration Data Set Export Status Update</b>	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Awaiting Approval, Unchanged or Error	30
Approval	30	<b>Review comparison results, approve / reject as needed</b>	Manual	Migration Data Set Import, drill in to the Transactions / Objects as necessary.		Migration Object - Approved or Rejected (no records should be in Needs Review) Migration Data Set Import - Apply Objects	40
Apply	40	<b>Apply Objects</b>	Batch		F1-MGOAP - Migration Object Monitor - Apply	Migration Object - Applied or Error Applying	41
Apply	41a	Data Set Transition - auto transition to Apply Transactions.  Only applicable if no migration objects are in Error Applying	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Transactions	42
Apply	41b	<b>Data Set Transition - manual transition to Apply Transactions.</b>  Occurs if user reviews errors and determines that they may be resolved in the Apply Transaction step.	Manual	Migration Data Set Import - click Apply Transactions		Migration Data Set Import - Apply Transactions	42
Apply	41c	Data Set Transition - manual transition to Retry Objects.  Occurs if user decides to fix an external error and wants to try the batch level Apply again.	Batch	Migration Data Set Import - click Retry Objects		Migration Data Set Import - Apply Objects	44
Apply	42	<b>Apply Transactions</b>	Batch		F1-MGTAP - Migration Transaction Monitor - Apply	Migration Transaction - Applied or Error Applying Migration Object -	43

Step	Seq	Action	Manual / Batch	Portal — Action	Batch Control	Record Impacted — Resulting Status	Next Action Sequence
						Applied or Error Applying	
Apply	43a	<b>Data Set Transition - auto transition to Applied</b> Applicable if all transactions are Applied	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Applied.	N/a
Apply	43b	Data Set Transition - manual transition to Retry Objects. Occurs if user decides to fix an external error and wants to try the batch level Apply again at the Object level.	Manual	Migration Data Set Import	- click Retry Objects	Migration Data Set Import - Apply Objects	44
Apply	43c	Data Set Transition - manual transition to Retry Transactions. Occurs if user decides to fix an external error and wants to try the batch level Apply again at the Transaction level.	Manual	Migration Data Set Import	- click Retry Transactions	Migration Data Set Import - Apply Transactions	45
Apply	44	Move Objects from Error Applying back to Approved. Occurs if user chose to Retry Objects.	Batch		F1-MGOPR - Migration Object Monitor	Migration Object - Approved	40
Apply	45	Move Transactions from Error Applying back to Ready to Apply. Occurs if user chose to Retry Transactions.	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply	42

The following table summarizes the batch monitor jobs that are used in the import process. You can see that there are special monitor processes for the Apply step for both the Object and Transaction. However, for all other states that have monitor logic, the standard monitor process for that MO is used.

Batch Control	Description	Comments
F1-MGDIM	Migration Data Set Import Monitor	Processes data set records in the following states:

Batch Control	Description	Comments
		<ul style="list-style-type: none"> <li>• Pending</li> <li>• Ready to Compare</li> <li>• Apply Objects</li> <li>• Apply Transactions</li> </ul>
F1-MGTPR	Migration Transaction Monitor (Deferred)	Processes transaction records in the following states: <ul style="list-style-type: none"> <li>• Pending</li> <li>• Error Applying</li> </ul>
F1-MGTAP	Migration Transaction Monitor - Apply	Processes transaction records in the Ready to Apply state where the data set is in the Apply Transactions or Canceled state. <p><b>NOTE:</b> Be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker.</p>
F1-MGOPR	Migration Object Monitor	Processes object records in the following states: <ul style="list-style-type: none"> <li>• Pending</li> <li>• Needs Review (check for Data Set cancellation)</li> <li>• Rejected (check for Data Set cancellation)</li> <li>• Error Applying</li> </ul>
F1-MGOAP	Migration Object Monitor - Apply	Processes object records in the Approved state where the data set is in the Apply Objects or Canceled state. <p><b>NOTE:</b> Be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker.</p>

Refer to [Running Batch Jobs](#) for more information about managing the batch jobs.

## Cancelling a Data Set

A user may choose to **Cancel** a data set to prevent it from being processed at any point during the process.

If related migration transactions or migration objects have already been created, they will not be canceled as part of the data set getting canceled (due to possible high volumes of related records). They will be canceled the next time an appropriate monitor batch process runs. The child records checks to see if the data set has been canceled prior to any state transition.

## Additional Note Regarding Imports

The following points describe miscellaneous comments related to Migration Import.

- CMA relies on the fact that database referential integrity constraints are not in place, and that the SQL statements can be run in any order within the transaction. Any archiving solution that requires referential integrity constraints (such as Information Lifecycle Management) would not be possible on this data. Given that CMA migrations comprise administrative data and not transactional data, this should be a reasonable exception.
- The validation that is performed is only via the **Page Validate** service. BO validation algorithms are not executed. Page validation does not include validation of the business object against the schema (for example, for required fields, field sizes, etc.).

- All migration requests can be exported at the same time. On the import side, however, you should consider importing, reviewing, and applying an entire file/data set before moving on to the next one. The reason is that if objects are included in more than one file, two sets of "inserts" will be generated, but only the first will succeed. The second will cause an insert error on the object, and the transaction would be marked with status "Error Applying". If instead you wait until the first file is applied before importing the second, the second data set will not generate any SQL for the object, since it has already been inserted. It's a matter of efficiency: If you first import all files and then try to apply all, you'll have to identify the duplicated object as an error and then mark the object as rejected before applying the transaction.

## Caching Considerations

Because CMA updates administrative data that is usually read from a cache, a thread pool with the L2 cache disabled is required during the execution stages of migration. Contact your system administrator and ask for the name of a thread pool with the L2 cache disabled. This thread pool name must be entered in the batch controls used for CMA processing in the **Thread Pool Name** parameter.

Refer to [Running Batch Jobs](#) for more information about running CMA batch jobs.

The batch section of the *Server Administration Guide* also contains information on preparing and implementing thread pools and turning off the L2 cache.

Also note that after a successful migration, the target region now has new administrative data which needs to be part of the L2 cache. The existing thread pool workers should be restarted by your system administrators. In addition, it is recommended to restart the application server.

## Maintaining Import Data

This section describes the portals provided to add, view and maintain migration import data.

### Migration Data Set Import

Use the Migration Data Set Import portal to view and maintain migration data set import records. Navigate using **Admin > Migration > Migration Data Set Import**.

### Migration Data Set Import Query

Use the query portal to search for an existing Migration Data Set Import record. Once a record is selected, you are brought to the maintenance portal to view and maintain the selected record.

In addition, the query provides an option to specifically search for data sets that have either objects in error or transactions in error.

### Migration Data Set Import Portal

This page appears when viewing the detail of a specific migration data set import record.

The topics in this section describe the tabs that appear on the portal.

### Migration Data Set Import - Main

This portal appears when a migration data set import record has been selected from the query portal. The topics below describe the base-package zones that appear on the portal.

## Migration Data Set Import

The Migration Data Set Import zone contains display-only information about the selected record.

Please see the zone's help text for information about this zone's fields.

## Migration Data Set Transactions

This zone is visible once the [Import Step](#) has occurred and lists all the transactions that are related to the data set. To see more information about a specific migration transaction, click the hypertext for its ID. This brings you to the [Migration Transaction](#) portal.

## Migration Data Set Impacted Object Summary

This zone is visible once the [Import Step](#) has occurred and lists the objects that are related to the data set. To see more information about a specific migration object, click the hypertext for its ID. This brings you to the [Migration Object](#) portal.

## Migration Data Set Objects in Error

This zone is only visible if the data set's status is **Apply Objects**, and there are objects in the status **Error Applying**. It indicates the error for each object. A user may use this zone to review errors after the monitor batch job to apply objects completes. Using the error information shown, the user can choose to transition the record to **Error Applying** or manually fix the cause of the error and click **Retry Objects**.

---

**NOTE:** Refer to [Apply Step](#) for more information about resolving errors.

---

## Migration Data Set Transactions in Error

This zone is only visible if the data set's status is **Apply Transactions**, and there are transactions in the status **Error Applying**. It indicates the error for each object. A user may use this zone to review errors after the monitor batch job to apply transactions completes. The errors received when attempting to apply objects at the transaction level may differ from those received when attempting to apply objects at the object level. A transaction log is created for each object error received and these exceptions are shown in this zone.

---

**NOTE:** Refer to [Apply Step](#) for more information about resolving errors.

---

## Migration Data Set Import - Log

Navigate to the Log tab to view the [logs](#) for the migration data set import record.

## Migration Transaction Portal

This page appears after drilling into a specific migration transaction from the migration data set portal or from the migration object portal.

The topics in this section describe the tabs that appear on the portal.

## Migration Transaction - Main

The topics below describe the base-package zones that appear on the portal.

## Migration Transaction

The Migration Transaction zone contains display-only information about the selected record.

Please see the zone's help text for information about this zone's fields.

## Migration Transaction Objects

This zone lists the objects that are related to the data set. To see more information about a specific migration object, click the hypertext for its ID. This brings you to the [Migration Object](#) portal.

## Migration Transaction - Log

Navigate to the Log tab to view the [logs](#) for the migration transaction.

## Migration Object Portal

This page appears after drilling into a specific migration object from the migration data set portal or from the migration transaction portal.

The topics in this section describe the tabs that appear on the portal.

## Migration Object - Main

The topics below describe the base-package zones that appear on the portal.

### Migration Object

The Migration Object zone contains display-only information about the selected record.

Please see the zone's help text for information about this zone's fields.

### Migration Object - Log

Navigate to the Log tab to view the [logs](#) for the migration object.

## Running Batch Jobs

There are several batch jobs that are part of the CMA process, especially the import step. And in some cases, a single batch jobs may process multiple states in the same business object lifecycle. Implementations must decide the best way to manage the batch job submission depending on how they plan to work.

- **Batch scheduler.** If an implementation wishes to put these batch jobs in the batch scheduler, a given job may need to be included several times to manage progressing the records to completion.
- **Timed Batches.** The batch controls can be configured as timed batches so that they run every N minutes based on the setting. This allows for the batch jobs to run periodically and process whatever is ready. A user doesn't have to manually submit a batch request. Navigate to **Admin > Batch Control > Search** and select the appropriate batch controls. For each one, change the Batch Control Type to **Timed**. Fill in the additional information that appears for timed batches.
- **Manual submission.** The user managing the CMA import process submits the appropriate batch jobs on demand when a particular step is ready. Navigate using **Menu > Batch > Batch Job Submission > Add**, select the appropriate batch control and fill in the parameters as needed.

Note that for the CMA batch processes, especially the "apply" batch processes, the setting of the **Thread Pool Name** parameter is important. This parameter specifies the thread pool that will be executed by the batch. Since the CMA apply processes update administrative data that is usually read from a cache during batch processing, the thread pool named here must be one with the L2 cache disabled. In addition, after successfully applying the migrated data, the L2 cache of all thread pools must be refreshed. For more information, see [Caching Considerations](#).

---

**CAUTION:** Be sure that the Thread Count set when submitting the batch job does not exceed the number supported by the thread pool. Otherwise the extra threads will simply wait until the supported number of threads are finished.

---

Refer to the parameter descriptions in the batch control metadata for more information about filling in the parameters.

For additional details on submission controls, refer to the topic [Batch Job Submission - Main](#) in the Batch Jobs section.

## CMA Reference

This section provides additional reference information.

## Framework-Provided Migration Configuration

This topic describes special information relating to migration objects provided for use by CMA in the product. Additional objects may be provided by your specific product. Any special information for objects is provided separately in each product's documentation.

You can see all currently available objects and their descriptions by performing a search in the **Migration Plan Query** or **Migration Request Query** pages.

The following points highlight some information about the Framework-provided migration plans and migration requests:

- Fields and characteristic types are not migrated with the object unless specifically indicated.
- The **Application Service** used by an object is migrated only if it is CM-owned.
- The **F1-Language** migration plan should **never** be used in a migration request with other objects. Defined constraints will cause CMA to link all objects with a language table—which is almost all of the system admin data—into one large transaction. Attempting to update and commit all the changes for this one large transaction may cause performance issues and/or out of memory errors. For this reason, this MO has its own migration request, **F1-Languages**.
- The migration request **F1-SchemaAdmin** defines migration plans related to data that is commonly used in schema-based objects.
- The **Batch Control** object optionally references a User. If this user does not exist on the target system, CMA cannot apply the requested changes.
- The base migration plans for MO and BO include instructions to copy option types that use foreign key references to refer to other objects. Note that the data stored in the options are not validated, so defining these instructions is not required when doing wholesale migrations. However, including subordinate instructions for foreign key references is useful for piecemeal migrations to ensure that the related data is included in the migration. If you add additional MO or BO option types that use foreign keys and you want to support piecemeal migrations, you must create custom migration plans and requests for MO and BO, respectively to include these referenced objects in the migration plan. Note that you do not need to duplicate the instructions in the base migration plans. You may define the additional migration plans to only have the additional custom option types. When submitting a migration request for MO or BO you must include both the base migration plans and the custom migration plans in the request.
- The migration plans provided by CMA migrate scripts, schema-based objects and zones, and, through constraints, some of the associated data associated with them. However, data specified through alternate formats (such as through **Edit Data** steps in scripts, referenced in schemas for schema-based objects, or data from mnemonics in zone parameters, etc.) are not identified and combined in the same transaction. It's important to note that this could cause validation errors during **Apply**, and may require retrying or migrating the additional data separately. See [Apply Step](#) for a description of how to reapply the import should an **Applied With Errors** result occur.
- There are two migration plans for **Scripts**. The migration plan **F1-ScriptOnly** migrates just the script and its **Application Service** (provided the Application Service is CM-owned). This is used in the **F1-FrameworkAdmin** migration request, which migrates all admin objects. The migration plan **F1-Script** includes most related objects, but does not migrate any objects referenced in the edit data area steps. It does not move the **Function** maintenance object

(which has been deprecated). This migration plan is not included in any base migration requests. It may be included in any appropriate custom piecemeal migration request where scripts and related data should be migrated.

- If your implementation includes a **Feature Configuration** setting for the **F1\_DBCONINFO** entry that will be included in a migration request, be sure that the import user on the target region has the appropriate security rights to this entry (**Super User** access mode for the Feature Configuration application service (**CILTWSDP**)).
- All of the provided Framework-owned migration requests include the SQL statement "1=1" so that all records in the plan are migrated. If an implementation wants to limit a migration to a subset of records, the prepackaged migration request may be duplicated and then customized to modify the key selection criteria.
- Many of the XAI-related maintenance objects include references to environment-specific data, and data should be migrated with extreme care. Migration plans are not provided for MOs that contain mostly environment-specific data. Migration plans are provided for the following XAI-related maintenance objects, but these plans are not included in the **F1-XAI-Admin** migration request: **External System**, **XAI Receiver**, **Message Sender**, and **XAI Route Type**.

## Migration Requests for Wholesale Migrations

The following framework provided migration requests may be used in a wholesale migration. Refer to your specific product's CMA documentation for its recommendation on which migration requests to use for a full migration of framework and product administrative tables.

- The **F1-SchemaAdmin**(Framework Schema Admin) migration request. This request contains migration plans for objects needed by schema based objects, such as Field and Lookup. It has few dependencies on its data and many other objects include dependencies to it.
- The **F1-FrameworkAdmin** (Framework Admin) migration request. This includes migration plans for most widely used maintenance objects Your product may have chosen to incorporate migration plans from this migration request directly into one of its owned migration requests.
- The **F1-GeneralSystemOptions** (General System Objects) migration request. This includes migration plans for business related objects that may or may not be applicable to all products. Your product may have chosen to incorporate a subset of the migration plans from this migration request directly into one of its owned migration requests.

For each migration request determined to be needed for a wholesale migration, create a migration data set export record and process it as documented in **Exporting a Migration**.

## Executing Piecemeal Migrations

Piecemeal migrations involve copying a targeted set of data. It may be that it includes a subset of maintenance objects (for example all Activity Types or all Asset Types) or it may further limit the records within one or more maintenance objects (for example, all 'electric' rates). Many piecemeal migrations are ad-hoc, depending on the specific requirements at the time.

If your product has provided migration requests for piecemeal migrations out of the box, it would they would be defined to copy the appropriate type of data. However, the selection criteria would most likely not be limited to a type of record. As such, it could be used as is for a "copy all data in these maintenance objects" migration. For example, the framework provides a migration request for Security Configuration. This may be used to copy all the data in the various security tables. Oracle Public Sector Revenue Management provides a migration request to copy form types and its related data.

If your implementation wants to copy only a subset of data for maintenance objects covered by a base provided migration request, do the following:

1. Duplicate the base migration request.
2. Find the migration plan for the object or objects where you want to limit the data. Enter appropriate Key Selection criteria to select the desired data.
3. Use that migration request to request a migration data set export request.



If your piecemeal migration requirement is not satisfied by a base migration request that may be used as a starting point, review the base migration plans for the records that should be copied. Determine if the migration plans have the desired subordinate instructions to copy (or not copy) the related data as desired.

- If existing migration plans satisfy your requirements, create a migration request that includes those plans (and desired key selection as needed). Use that migration request to create a migration data set export request.
- If existing migration plans do not satisfy your requirements, create migration plans as needed. Create a migration request that includes those plans (and desired key selection as needed). Use that migration request to create a migration data set export request.

## Configuring Facts

---

Fact is an optional configuration tool for simple workflow-type business messages or tasks. The base package does not provide a dedicated Fact user interface because fact is generic by design. Implementations configure their own user interface to visualize the desired custom business process. The topics in this section describe the generic Fact entity and how it can be customized.

### Fact Is A Generic Entity

The Fact maintenance object is a generic entity that can be configured to represent custom entities and support automated workflows for a variety of applications. Each fact references a business object to describe the type of entity it is. A status column on the fact may be used to capture its current state in the processing lifecycle controlled by its business object.

The maintenance object also supports a standard characteristic collection as well as a CLOB element to capture additional information.

#### Where Used

Follow this link to open the data dictionary where you can view the tables that reference [F1\\_FACT](#)

### Fact's Business Object Controls Everything

A fact's business object controls its contents, lifecycle and various other business rules:

- Its schema defines where each piece of information resides on the physical Fact maintenance object.
- It may define a lifecycle for all fact instances of this type to follow. Each fact must exist in a valid state as per its business object's lifecycle definition.
- It may define validation and other business rules to control the behavior of facts of this type.

---

**FASTPATH:** For more information about business objects, refer to [The Big Picture of Business Objects](#).

---

### Fact Supports A Log

The Fact maintenance object supports a log. Any significant event related to a Fact may be recorded on its log. The system automatically records a log record when the fact is created and when it transitions into a new state. In addition, any custom process or manual user activity can add log entries.

---

**FASTPATH:**

---

Refer to [State Transitions Are Audited](#) for more information on logging. For more information about the various configuration tools available, refer to [Configuration Tools](#). For more information about user interfaces, refer to [Portal and Zone Features](#).

# Chapter 2

---

## Oracle Utilities Work and Asset Management Administrative User Guide

---

The topics in this section describe how to administer Oracle Utilities Work and Asset Management.

### System Wide Oracle Utilities Work and Asset Management Configuration

---

This section provides information on general system wide settings which fall outside of the major functional areas within the application.

#### Defining Home Page Configuration

The Oracle Utilities Work and Asset Management home page provides a comprehensive dashboard where users can access all of the information they need to review and complete their work.

Please refer to [About the Home Page](#) in the Business User Guide for a conceptual overview of home page functionality.

Beyond entering **Master Configuration** settings for the home page, you might also consider the following:

#### Configure User Settings for the Home Page

Users can add or remove zones on the **Home Page** by navigating to **Preferences** and selecting the **Portal Preferences** tab. Here, under the various Home Page sections, you can select which default zones display, whether or not they are initially collapsed or expanded, and the order in which they appear. Depending on your business practices, you can choose to disable this ability for users by modifying their access to these settings by assigning a **Template User** for **Portal Preferences** to each **User** record. Refer to [The Big Picture of Portals and Zones](#) for more information.

## Add Zones to the Home Page

You may find that it is beneficial to include additional zones on the **Main** tab of the **Home Page** such as a zone to show **Saved Searches**, **Bookmarks** or **Favorites**. You may also want to add organizational information such as a company news zone or additional custom zones. You can configure the Home Page, or any portal, to display the zones that you choose by configuring zone settings as described under [The Big Picture of Portals and Zones](#).

## Defining General System Master Configuration

This section provides information about general system master configurations.

Use the following to locate information about master configuration:

- Refer to the embedded help for descriptions of the specific settings on the **Master Configuration** pages for functional areas.
- Refer to the [Defining Master Configurations](#) section for more technical details on using master configurations.

---

**NOTE:** Note that for most system functionality, master configurations must be set before other settings or configurations. However, there are some master configurations that depend upon initial configuration of admin settings. Please refer to the embedded help for each individual master configuration for more information.

---

## About Access Control

Access control provides a way for your organization to limit or classify record information by "plant" or varying work locations. For example, you may have an asset type that is only used at your northern location, so you would use an access type to designate that asset type as restricted to that location. You can also use access controls to hide or show areas of the application to users based on any other business requirements as needed.

Access groups are defined in the **Access Group** portal.

Most objects in the system have an optional reference to an **Owning Organization** which allows you to choose which work location applies for the entity being created. Some administrative "type" objects which have a corresponding transactional object also have an **Instance Organization** which is used as the default **Owning Organization** on the transactional object. For example, the **Instance Organization** identified on an Asset Type is used as the **Owning Organization** for Assets of that type.

Users must be granted rights to **Owning Organizations** in order to use them. If your implementation needs to restrict access to rows based on the Owning Organization value, you will need to implement database policies in conjunction with using the **Owning Organization** values and rights.

Refer to [Access Groups](#), [Data Access Roles and Users](#) for more information.

## About Statistics

The **Statistics** portal captures metrics that are collected via batch processing. These embedded analytics are not present in real time, but rather are accumulated and presented as they are processed by batch. These analytics are displayed on the **Home Page**.

This portal is mainly used as a method to speed up the processing associated with presenting complex statistics or statistics that represent a large amount of data. Processing these in batches rather than in real time significantly increases system performance.

Authorized users can also use the portal to manipulate the statistical data that is displayed on the **Home Page** such as selecting crews or dates. The **Log** tab tracks who made changes to the data, and when.

Refer to the section or the [Defining Home Page Configuration](#) section for more information.

During configuration, you will want to review this portal to ensure that the metrics are set up to reflect your business requirements.

## Communication

This section provides the configuration guidelines related to communication types.

### About Communication Types

Communication types define the information required for the system to receive messages from external systems.

#### Communication Data Mapping

Communication Data Mapping information is used to populate communication business object elements from information within the inbound communication's raw message.

#### Attachment Suppression

When a message is received from an external system, the message may contain attachment data which is very large. The substitution text entered in this section can be used in place of that large data in the completion event list on the inbound communication. The attachment element xpath identifies the data to match it to the substitution text. This serves to prevent transfer delays in cases where large amounts of data might overload the browser.

### Defining Communication Types

Navigate to the **Communication Types** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## BI Configuration Administration

This section provides information about requirements for defining Oracle Utilities Advanced Spatial and Operational Analytics configurations.

BI configuration administration involves creating, maintaining and working with Oracle Utilities Advanced Spatial and Operational Analytics configurations. Refer to the *Oracle Utilities Advanced Spatial and Operational Analytics* documentation for more information and the specific data mapping and metric reference guidelines for business intelligence.

### Master Configuration for Business Intelligence

Ensure that you define extract parameters as required for your implementation using the **Master Configuration** portal.

#### BI Configuration Portal

The **BI Configuration** portal holds information about the configuration tasks related to Oracle Utilities Advanced Spatial and Operational Analytics.

This is a display-only portal that provides a bird's eye view of the configuration for Oracle Utilities Advanced Spatial and Operational Analytics. The portal also provides links and guidelines for the areas that need configuration to successfully run the extract transfer and load processes from Oracle Utilities Advanced Spatial Analytics.

Users can leverage the reporting capabilities of Oracle Utilities Advanced Spatial and Operational Analytics to view various measures such as:

- How many Assets have failed acceptance testing
- How many Assets have been in store for a given period
- How many Assets have been installed for a given period
- and so on.

## Bucket Configuration

Key performance indicators can be defined to measure the age of assets grouped into ranges, or buckets, which are defined in the **Bucket Configuration** portal.

Analysts can use these metrics to review the overall asset ages classified into various ranges, or buckets, such as 0 - 100 days, 100 - 300 days or more than 300 days, and so on.

Use the **Bucket Configuration** portal to define age buckets, such as:

- Asset age
- Asset installation age
- Asset in storage age
- Asset criticality
- Asset condition score
- Number of cycles overdue
- Work priority

These buckets can be defined for a specific asset type or as a default set of buckets that will be applicable to all assets that do not have any specific buckets defined on the asset type.

# Designing Approval Processing

---

The following sections provide information on configuring approval processing.

## Understanding Approval Processing Configuration

This section provides the configuration guidelines related to approval processing.

One of the ways that the system simplifies and speeds up work is by routing records when they require approval. For example, purchase requests, work requests, employee leave, and other functions can be configured to require approvals.

Approval requests use approval profiles to determine the properties and requirements for the approval. The approval profile contains the rules for a certain kind of approval process, including the To Do Type to send out, the Approval Hierarchy with To Do Roles and approval limits, the type of objects that the profile pertains to, and the work calendar to use. A common structure defines the approval process, users, and limits for each given functional area using approval processing.

Please refer to the [Approval Processing Overview](#) section in the Business User Guide for a conceptual overview of approval functionality.

## About Approval Profiles

Approval profiles are used to define the approval hierarchy within your organization.

Based on **To Do Roles**, approval profiles define:

- the types of documents the role can approve
- the working calendar to use when determining approval time lines and when to send To Dos
- dollar amount maximums that the role is authorized to approve

An approval profile can have one or many **To Do Roles**, and it is possible for an approver to be assigned to more than one To Do Role in the same approval profile.

Depending on the notification setting designated on the approval profile, the approver receives an email and/or an alert in their To Do List.

Note: Individual approvers can choose to modify their notification settings for approvals. Navigate to the **User** portal and access the **Characteristic** tab to set the **Approval Profile E-Mail Override** setting to disable email notifications for approval requests. When this is set, the approver does not receive any emails and must rely on the approval dashboard or To Dos.

## Configuring Approval Processing

This section provides configuration tasks for approval processing administrative support records.

Please refer to the [Approval Processing Overview](#) section in the Business User Guide for a conceptual overview of approval functionality.

Refer to the [Understanding Approval Processing Configuration](#) section in the Administrative User Guides for configuration guidelines.

## Defining an Approval Profile

Navigate to the **Approval Profile** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** Work calendars and related document types must be created in order to reference them on the approval profile.

The base product provides a **To Do Type** for each base maintenance object that needs approval. Your implementation must add **To Do Roles** and assign users to those roles.

**To Do Types** are associated on the approval profile to determine which **To Do Types** are responsible for that profile.

Establish the approval hierarchy for the profile by entering **To Do Roles**, approval types, notification methods and amount limits for each **To Do Role**.

## Setting the Approval Profile for a Record Type

Your organization should configure your system with the **To Do Types**, **To Do Roles**, and **Approval Profiles** in combination so that when a user selects the **Submit for Approval** option, the list of possible approval profiles is limited to a finite selection of approvers based on the type of document they are trying to approve.

## Defining the Approval Hierarchy

Approval profiles can reference one or many **To Do Roles**. The way in which you structure your approval profiles depends on how regulated the structure needs to be for your organization.

Your business practices may be such that your approvals can go to a limited number of Supervisor and Manager roles, or they can be more advanced having detailed levels. To this end, the To Do Roles you create for can be shared across Document Types or they can all be kept discrete to business process and functionality. This is also true for the No Approval user group. There can be one to be reused across your approval processing configuration, or you can set more to address different functions.

One type of record, such as work order, might require several different **Approval Profiles** to address different situations. For example,

- PM work might be “auto-approved” since the work needed has already been analyzed, planned and approved.
- Emergency work might bypass normal approval processing since it might come in at off hours or will otherwise be subject to different approval than regular work.
- Regular work will likely require general approval processing.

## Approval Types

Approval Types indicate whether the approval role is standard, mandatory or notification only:

- **Standard:** A standard approver might appear on the approval hierarchy with other approvers with equal approval authority. Any standard approver can record the decision before the record moves on to the next approver in the hierarchy.
- **Mandatory:** Indicates that this approver must provide an approval decision on the record. Approval processing is not complete until this decision is recorded.
- **Notification Only:** Notification only approvers are only sent a To Do that the approval is pending. The system does not validate an amount limit and the approver does not have to respond in order for the record to be approved.

## Additional Approval Profile Settings

- **Initial Notification Method:** Indicates whether the approver receives an e-mail, an alert in their To Do List, both or none.
- **Authorized to Approve:** (Non-Amount Based only) When multiple To Do Roles are added to the profile, this field identifies which is authorized to approve. At least one to do role must be added with this value set to Yes.
- **Amount Limit:** Provides the maximum amount that the approver in the indicated role is allowed to approve. If the value on the record in need of approval exceeds this limit for the role, the approval path must move on to the next approver or the requester is notified that no one on the approval profile has sufficient authority to approve the request.
- **Timeout Threshold:** Indicates the amount of time that the approver has to respond to the approval request, in days. For example, if the threshold is set to “2”, the approver has 2 days to approve requests. If the approver does not respond within the set time, another alert is sent. Additional alerts are sent up to the set number of retries.



- **Maximum Retry:** Indicates the number of times that the system will retry sending the approval request to approvers with the ToDo Role before timing out the request. For example, if the Timeout Threshold is set to “2” and the Maximum Retry is set to “3” the system will try three times for a total of 6 days resending the alert or email every 2 days. If the approvers fail to respond, the request is routed to the next approver in the hierarchy and that person receives the email or alert.

The following table provides examples of the types of roles you may need to create to support your approval hierarchy:

Admin Object	Example of Data to Create
To Do Types	<p>These default types are delivered with the base package with the ToDo Type Usage set to <b>Automatic</b>:</p> <ul style="list-style-type: none"> <li>• Approval Type — Employee Unavailability</li> <li>• Approval Type — ODC</li> <li>• Approval Type — Project</li> <li>• Approval Type — Purchase Order</li> <li>• Approval Type — Purchase Requisition</li> <li>• Approval Type — Timesheet</li> <li>• Approval Type — Work Order</li> <li>• Approval Type — Work Request</li> </ul>
To Do Role	<ul style="list-style-type: none"> <li>• General Approver</li> <li>• General Director</li> <li>• General Manager</li> <li>• General Notification Only Approver</li> <li>• General Supervisor</li> <li>• General Work Management Supervisor</li> <li>• General Work Management Manager</li> <li>• General Purchasing Manager</li> <li>• General Purchasing Supervisor</li> </ul>
User Groups	<ul style="list-style-type: none"> <li>• Authorized for No Approval</li> </ul>
Approval Profiles	<ul style="list-style-type: none"> <li>• Approval Amount <ul style="list-style-type: none"> <li>• General Purchasing Approver</li> <li>• High Value Purchasing Approver</li> <li>• General Work Request Approver</li> <li>• General Work Order Approver</li> <li>• High Value Work Order Approver</li> <li>• Project Approver</li> <li>• Other Direct Charges Approver</li> </ul> </li> <li>• No Approval <ul style="list-style-type: none"> <li>• Emergency Work Approval</li> </ul> </li> <li>• No Amount <ul style="list-style-type: none"> <li>• Employee Unavailability Approver</li> </ul> </li> </ul>

Admin Object	Example of Data to Create
	<ul style="list-style-type: none"> <li>Timesheet Approver</li> </ul>

Using a scenario of a moderate structure with a separate approval profile for each document type, refer to the following table for an example of the hierarchy:

Seq	To Do Role	Approval Type	Initial Notification Method	Amount Limit	Time Out Threshold	Maximum Retry
10	Supervisor	Standard	To Do List Only	\$1000	2 (days)	1 (retry)
20	General Manager	Mandatory	Both (email and To Do List)	\$5000	2 (days)	2 (retries)

This example establishes a hierarchy where the first role to receive the approval request is at the supervisor level with a minimal approval limit. This supervisor would approve the request only to inform the general manager that he or she approves, but since the General Manager has been set with a mandatory approval type, he or she must approve the request for the processing to be considered complete.

## About No Approval Required Profiles

User groups are referenced on **No Approval Required** approval profiles to indicate the user groups that can create an approval request routed to the indicated approval profile.

This helps to control which users are allowed to submit “no approval” requests. User groups are defined as a listing of user profiles in the **User Group** portal.

## About No Amount Profiles

No amount profiles are used for documents that do not have explicit amounts/costs associated - for instance a Work Request would not support Amount-based Approval since there is no cost estimate associated with it. In this case you would use the no amount based approach in configuring To Do Roles.

## Defining To Do Types, To Do Roles, and To Do Entries for Approvals

It is recommended that you follow the delivered naming convention and preface your **To Do Types** and **To Do Roles** with “Approval” so that they are easy to find when adding them to approval profiles.

When an object, such as a work order, is submitted for approval, an approval request record is created showing the related document, the person who submitted it and when, and which roles need to approve. Background processing creates a **To Do Entry** for each step of the approval process, and the entry gets linked to the approver when action is taken. Administrators can review **To Do Entries** to audit how approval processing is working.

Please refer to the Oracle Utilities Application Framework section titled [To Do Processing](#) for more information about To Do functionalities.

# Designing Asset Management

---

The following sections provide information on concepts related to managing assets and components.

## Understanding Asset Management Configuration

This section provides configuration guidelines for asset functionality.

The Oracle Utilities Work and Asset Management system provides functionality to track the purchase, repair and maintenance of assets and asset components using general work management processing, preventive maintenance techniques, purchasing processing and inventory tracking.

Please refer to the [Asset Management Overview](#) section in the Business User Guide for a conceptual overview of asset functionality.

## About Asset Types, Assets and Components

Asset types define the attributes for assets of a certain type.

This includes determining information such as the valid location types and service history types for the assets and whether or not such assets can have attached components. Average cost statistics are displayed graphically on the **Asset Type** record. An asset is an object in an infrastructure, such as a smart meter, poles, pipes, transformer or any other material item owned or managed by an organization. Components are devices or other objects that are attached to another device, usually an asset.

### Assets and Components

The application provides a number of utilities specific business objects that can generally be used to create assets and components for specific purposes. However, your implementation will likely require several business objects to define assets and components of other types.

### Specifications

The base package specifications provide a general starting point for your specification set-up. Specifications can include a wide variety of possible fields, so the definitions of your specifications should be considered carefully.

For example, your implementation may want to include additional fleet/vehicle specification details by extending the data area - W1-VehicleDetail. Specification detail values may also be extended by adding values to the following look-up fields:

Specification detail values may also be extended by adding values to the following look-up fields:

- **MODEL\_YEAR\_FLG** - model year
- **VEHICLE\_COLOR\_FLG** - vehicle color
- **DRIVE\_TYPE\_FLG** - drive type
- **FUEL\_TYPE\_FLG** fuel type

### Tracked Assets, Stock Items and Inventory

Link a specification to stock items to associate the items to the tracked assets that are also associated with the specification. This allows the stock item to be used to track the asset. Only specifications with asset types configured with an 'inventory controlled storeroom' node type can be added.

## Extending Asset Statuses

If your implementation needs to create a new Asset Lifecycle BO with a status which differs from the base package, you can choose to create them by extending W1-AssetDisposition. With this extension, note the following:

- ‘assetRelationship’ on the Asset Disposition has 3 values: Installed, Attached and Not Installed. Installed is applicable to assets when they are installed in service.
  - **Attached** is applicable to components when they are attached to an asset.
  - **Not Installed** is applicable when assets or components are in other statuses (in store, in receipt, in repair, rejected or retired).
- Since the assetRelationship is in a clob, the base package uses a naming convention using the asset disposition code. The first two characters should be ‘IN’ for installed, ‘AT’ for attached and ‘NI’ for not installed.

## Linear Assets

Setting up your assets so that they are recognized as a “system” can help you to manage maintenance. If your organization uses GIS, this functionality can also help to position your assets on the mapping system that you have in use. Please consider the following when configuring linear assets:

- The Asset Location Type and Asset Location determine where the linear asset is connected.
- Linear assets include “Connected to” fields which allow you to define the connections between assets

For example, a pipe system would consist of a pipe connected to a manhole connected to the next pipe, and so on.

## About Asset Location Types

Asset location types define the details of asset or component locations of a certain type.

Valid asset locations might be service points, other locations for installed assets, or any other location as defined by your organization, for example:

- General asset location types define a type of asset location that is used to indicate the physical location where assets or components are installed.
- Plant asset location types define a type of asset location for a fixed organizational structure such as a building, factory, or other facility.
- Linear asset location types define a type of asset location that is part of a system such as pipes and manholes or poles and cables.

## Out of Service Location Types

When assets are changed to statuses such as “In Repair” or “Pending Disposition” the asset location is typically transitioned to an out of service location. This may be a warehouse or garage where the asset is stored, dispositioned or repaired.

## About Service History Types and Service History

Service history types define the main attributes for service history of a certain type. Types might include maintenance, inspection, service, test, measurement, and so on.

The base package service history business objects can be used out of the box, however it is likely that your implementation will want to configure special business objects to capture specific service history events.

## Planned Service History Types

Your implementation can create “planned” service history types for use with template activities. These service history types can be selected by users when creating the completion event to add service history on an activity.

## About Condition Assessment Questions

Questions and related answers can be entered to define “inspection templates” which guide asset inspections. Questions are associated to service history types which are then associated to asset types.

When it comes time to complete an inspection, the previously defined set of questions and answers guides the process for inspectors as they complete the inspection out in the field. Answers may be presented as a check box, multiple choice, or free-form text.

### Sample Questions and Answers

What is the condition of the pipe?

- **Good** no severe cracks and less than 2 small defects per 100 ft.
- **Acceptable** no severe cracks and less than 5 small defects per 100 ft.
- **Needs Attention** no severe cracks but 5 or greater small defects per 100 ft.
- **Needs Immediate Attention** One or more severe cracks detected.

---

**NOTE:** The bulleted items would appear as radio buttons in the actual questionnaire as answers.

---

Questionnaires also support a "fill in the blanks" format. A similar question to that above might appear as follows with “fill in the blanks” t:

Assess the condition of the pipe:

- Small Defects per 100 ft.: |\_\_\_\_\_|
- Number of severe Cracks: |\_\_\_\_\_|

---

**NOTE:** In this case, each blank would appear as text boxes in the actual questionnaire to capture the answers.

---

## About Condition Assessment Groups

Condition assessment groups are used with asset assessment/condition rating functionality. They group assessment service history types (i.e. inspection types) and to factor in asset aging factors to comprise a weighted combination of elements that affect how condition ratings are calculated.

Use the Service History Type section to establish a list of valid service history types that can be used with this assessment group.

- **Weight:** Represents the importance of the service history type as it relates to the overall group. For example, you might add 2 types, one for visual inspection and one for mechanical inspection. The visual inspection might have a weight of .25 where the mechanical inspection is weighted more heavily at .75. The overall total weight must equal 1.
- **Calculation:** Indicates whether or not the service history type should be included in the overall scoring calculation for the inspection.

---

**NOTE:** Any service history type that is used for scoring must have a weight. The sum weight of all service history types used for that scoring must be equal to 1.

---

Use the Aging Factors section to define the parameters for calculating the confidence rating for inspections related to the assessment. Confidence rating is a percentage scoring method for assets where 100% means complete confidence that the asset is in good condition. The percentage drops as asset condition data ages. Note that the older the data, the lower the percentage because the less reliable the inspection data. The confidence rating is shown on the related asset record.

- **Sequence**
- **Start Range/End Range:** Define an age range, in number of days.
- **Aging Factor:** A numerical value which, when applied, decreases the validity of the inspection score as the inspection date becomes less relevant.

The ranges and aging factor can be determined as best derived by your business practices. As an example, you may break the ranges down into quarters:

- 1.00 for 0 to 90 days
- .75 for 91 to 180 days
- .50 for 181 to 270 days
- .25 for 271 to 99,999 days

Note the final end range is entered as “99,999” since the system requires a fail safe end range. The confidence rating resets each time a new, “fresh”, inspection is completed.

## About Condition Assessment Classes

Assessment classes are sets of condition assessment groups, and are necessary for asset assessment/condition rating functionality. They are used to manage assets according to a set of inspection criteria and are used to define an inspection rating system for inspections. The purpose of this rating system is to measure the deterioration of the assets and the effect of the deterioration on the service levels that are expected of the asset.

Inspections are performed to assess the asset's current physical condition via a scoring method that grades the asset based on severity and number of deficiencies found. The results of the inspection program are used to determine the residual life (i.e. effective remaining life) of an asset and to support the review of preventative maintenance activities to ensure optimization of the asset's life. Please note the following with respect to assessment classes:

- Assessment classes are associated to assets on the asset type.
- Assessment classes are formed by a grouping of condition assessment groups. The total of all weights must equal 1.

The **Assessment Class Summary Information** section displays a pie chart to represent the percentage of assets by score and a cluster chart to show the number of inspections, per type, per month (for the most recent months).

The **Group Weight** field represents the importance of the assessment group as it relates to the class. For example, you might add 2 classes, one for visual inspection and one for mechanical inspection. The visual inspection might have a weight of .25 where the mechanical inspection is weighted more heavily at .75. The overall total weight must equal 1.

## About Asset Inspections (Assessments)

This section provides information on the setup needed to configure asset inspections.

Each **Asset Type** can be associated with many **Service History Types**, each of which falls into a **Service History Category** (the lookup values are downtime, repair, inspection, maintenance).

Each **Service History Type** can define a set of **Questions** that are used to standardize the work that is performed. A Question can be set up as single or multiple choice, with predefined Question Answers for the user to select. Assessment Class and Assessment Group are additional categorization mechanisms to define how the Assets of an Asset Type are to be maintained and reviewed.

**Preventive Maintenance Schedules** link Service History Types with (template-like) Work Orders.

Comprehensively, an asset inspection is configured with the following:

- **Condition Assessment Questions and Answers**  
Questions reference a service history subclass which is based upon the types of service history that can be created such as deficiency, general purpose, general weighted, operational device, percent of new, and questionnaire. These represent the types of inspection scoring that is used to inspect and evaluate assets. The weight of each question is defined on the question itself. The overall scoring is a calculation of the overall weights on every condition assessment group which is included on every service history type included on the inspection. The **Calculation Flag** in the question is used to determine if the answer is used in the calculation of asset condition rating.
- **Service History Type** (referencing questions and answers and an inspection condition inspection group)  
Refer to [About Service History](#) for information on inspection types.
- **Condition Assessment Group** (referencing the service history type and defining the aging factors)
- **Condition Assessment Class** (referencing the condition assessment group)
- **Asset Type** (referencing an inspection service history type)
- **Preventive Maintenance Schedule** (referencing the asset type with the inspection service history)
- **Preventive Maintenance Trigger** (referencing the maintenance schedule, a template work order, and the activity type)  
When the maintenance is triggered, the system uses the template work order to generate the work order then activity to complete the inspection. Service history is generated from the completion of the inspection, and the inspection calculations are created as processing transitions through the inspection lifecycle. Refer to the Sample Asset Inspection Processing section for examples.

---

**NOTE:** These concepts can also be used to configure general preventive maintenance.

---

## Aging Factors

**Aging Factors** define a numerical value which, when applied, decreases the validity of the inspection score as the inspection date becomes less relevant. Confidence rating is a percentage scoring method for assets where 100% means complete confidence that the asset is in good condition. The percentage drops as asset condition data ages.

The ranges and aging factor can be determined as best derived by your business practices. As an example, you may break the ranges down into quarters:

- 1.00 for 0 to 90 days
- .75 for 91 to 180 days
- .50 for 181 to 270 days
- .25 for 271 to 99,999 days

## Sample Data Configuration

### Example of Assessment Class Data

Assessment Class	Assessment Group	Service History Type	Description	Group Weight	Participate in Calculations?	Assessment Weight
Rotate Equipment	Electrical			0.25	Yes	
		Insulation	Assess the insulation		Yes	0.60
		Conductivity	Assess the conductivity		Yes	0.40
	Performance			0.75	Yes	

Assessment Class	Assessment Group	Service History Type	Description	Group Weight	Participate in Calculations?	Assessment Weight
		Performance	Curve test		Yes	0.50
		Reliability	Reliability testing		Yes	0.50

### Example of Service History Type (General Weighted) Data

Service History Type	Question	Weight	Answer	Point Value
Rotating Equipment Electrical Insulation	How is the appearance?		Apparently near infinite insulation resistance	1
			Good condition, only slight discoloration	2
			No cracks or deterioration. Aging and discoloration okay.	3
			Detectable cracks / deterioration - non-penetrating	4
			Cracked / deteriorated to conductors	5
	Heat distribution	0.50	Appears "new", with no heat imbalance between winding	1
			No heat imbalance between windings or at the junction box	2
			Factors indicate equipment is in later half of normal life	3
			Detectable heat imbalance between windings or junction box	4
			Significant heat imbalance between windings or junction box	5

### Sample Asset Inspection Processing

This section provides sample processing examples for a local asset and an asset worked from an external system using inspection as an example. This general processing concept can be applied to other maintenance and activities performed on assets.

#### Example: Configuring and Performing an Inspection for a Local Activity

1. Create a scheduled activity type configured with:
  - Local Activity BO
  - Inspection Service History Type (e.g. Percent of New, General Weighted or Deficiency Inspection Type)



- Service History from Activity Type Completion Event Type (configured with Create Service History Using Payload Completion Event)
2. Create an activity for an asset using the activity type. Include the following:
  3. Create a **Create Service History Using Payload** Completion Event
  4. Perform the inspection and record the answers in the service history.
  5. Set the activity to complete to execute the completion event.  
This triggers the following:
    - Create the service history from the payload.
    - The service history total inspection points are calculated and normalized to a grade by looking up the grade based on the total points using extendable lookup.
    - The condition rating and confidence rating are calculated and the asset is updated with the values.

### **Example: Configuring and Performing an Inspection from an External System**

1. Create an outbound activity type configured with :
  - Outbound Activity BO
  - Inspection Service History Type (e.g. Percent of New, General Weighted or Deficiency Inspection Type)
  - Service History from Activity Type Completion Event Type (configured with Create Service History Using Payload Completion Event)
2. Create an activity for an asset using the outbound activity type.
3. Send the activity to external system.  
This triggers the following:
  - WAM creates the outbound message containing questions and answers from the service history type on the outbound activity type
  - External system receives message from WAM, does the inspection and sends the answers to the questions back to WAM
  - TFW/Integration layer creates the inbound communication containing the raw message from external system
  - Inbound communication gets processed
  - Based on the completion event type on the activity type, a Create Service History Using Payload Completion Event gets created (Pending state)
  - The inbound communication transitions the activity to next default state
  - The outbound activity transitions from Sent to Execute Completion Events
  - The completion event gets transitioned to the next state
  - Completion event transitions from Pending to Executed
  - The completion event creates the service history based on the service history type on the activity type using the payload on the completion event
  - Service History moves from Pending to Compute Score (via FW auto transition monitor)
  - The service history total inspection points are calculated and normalized to a grade by looking up the grade based on the total points using extendable lookup
  - The condition rating and confidence rating are calculated and the asset is updated with the values

## About Measurement Types and Measurements (Operational Readings)

Measurement types define the units to use for a particular measurement and establish the values needed to support the evaluation of measurements such as upper and lower reading thresholds, maximum reading value, and so on. Asset measurements, or operational data, might include mileage, hours of uptime, number of start-stops, , and so on.

Please note the following with regard to of implementing measurement types.

- Each asset is assigned a list of valid measurement types to regulate the types of measurements take for that asset.
- Typically measurement types would be metered reads or gauge reads. Gauge reads are usually compared to a threshold or valid range of values. For example, tire pressure for a typical truck tire should be between 34 and 36 psi. These would be defined as the lower limit and upper limit for the measurement type.
- Measurement type identifiers can be used if the measurement type needs to be identified in an external system.

### Measurement Type Exception Handling

The exception handling fields define the parameters for how the system should respond if there are issues with the reading such as the To Do Type and Role that should receive alerts, retries, and discard reason. For example, if the reading is outside of the measurement range, the system sends an alert to the indicated To Do Role.

### Measurements

Measurements manage the asset operational and runtime data which is collected and tracked for assets. Asset measurements might include mileage, hours of uptime, number of start-stops, and so on.

Configure the extendable lookup for **Gauge Reason** as needed.

Measurement readings are entered in the Measurement maintenance portal via the **Measurement Quick Entry** zone.

## About Measurement Unit of Measurement

Measurement unit of measurements (UOM) establish the types of units that can be used to measure assets. Each UOM identifies a definite magnitude of a measurement reading taken for an asset. This might include tire pressure (psi), temperature ratings (degrees), , and so on.

## About Asset and Component Configuration Types

(Oracle Utilities Operational Device Management only) Configuration types define the information required for various asset configurations with other assets, components, firmware, and other aspects of a functioning configured system.

Configurations are referenced on specifications which are then referenced on assets and components to indicate the best way that these entities can work with the other entities that they are connected to in the field.

In general configuration classes define the types of connections that are being configured. The generic classes that are available include:

- Single Asset
- Asset/Component
- Asset to Asset
- Asset/Component to Asset
- Asset to Asset/Component

- Asset/Component to Asset/Component

## Configuring Asset Management

This section provides configuration tasks for asset management administrative support records.

Please refer to the [Asset Management Overview](#) section in the Business User Guide for a conceptual overview of asset functionality.

Refer to the [Understanding Asset Management Configuration](#) section in the Administrative User Guides for configuration guidelines.

## Defining Measurement Unit of Measurements (UOM)

Navigate to the **Unit of Measure — Measurements** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Measurement Types (Operational Readings)

Navigate to the **Measurement Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the measurement type record:

Note: This list does not represent required fields.

- Measurement Unit of Measure

## Defining Questions

Navigate to the **Question** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Service History Types

Navigate to the **Service History Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the service history type record:

Note: This list does not represent required fields.

- Questions

## Defining Condition Assessment Groups

Navigate to the **Condition Assessment Group** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the condition assessment record:

Note: This list does not represent required fields.

- Service History Type

Service history type defines the questions and answers used when completing an inspection. Set up the condition assessment group to define which service history types are valid for the condition *assessment*. This will then be referenced on the condition assessment class to provide the comprehensive criteria for an inspection.

## Defining Condition Assessment Classes

Navigate to the **Condition Assessment Class** portal under **Admin**, select **Add** mode, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the assessment class record:

Note: This list does not necessarily represent required fields.

- Condition Assessment Group

A condition assessment class may consist of one or many condition assessment groups which define the comprehensive criteria for a type of inspection.

## Defining Asset Location Types

Navigate to the **Asset Location Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the asset location type record:

Note: This list does not necessarily represent required fields.

- Location Class/Subclass
- Service Type
- Parent Location or Organization Type (as applicable)

## Defining Out of Service Location Types

Navigate to the **Out of Service Location Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the out of service location type record:

Note: This list does not necessarily represent required fields.

- Location Class/Subclass
- Parent Location Type (as applicable)
- Organization Type (as applicable)

You can also enter a parent location or organization to establish the position of storerooms of this type in the location hierarchy.

## Defining Asset Types

Navigate to the **Asset Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the asset type record:

Note: This list does not represent required fields.

- Assessment Class
- Asset Location Type or Out of Service Location Type
- Service History Type
- Failure Profile

## Defining Component Types

Use this procedure to define component types.

**Prerequisites:** The following are referenced by the asset type record:

Note: This list does not represent required fields.

- Assessment Class
- Asset Location Type
- Storeroom
- Service History Type

1. Navigate to the **Asset Type** portal under the **Admin** menu and select **Add**.
2. Complete the required fields according to your business practices and **Save**
  - **Valid Location Types** where the component can be received, stored, installed, repaired, and so on.
  - **Valid Attached to Asset Types** to indicate the asset types where the component can be attached/installed.
  - **Valid Service History Types** which define the types of service that can be performed on components of this type.

You can now create components based on the component type.

## Defining Asset Configuration Types

Navigate to the **Configuration Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

# Designing Preventive Maintenance

---

The following sections provide information on configuring for preventive maintenance functionality.

## Understanding Preventive Maintenance Configuration

This section provides configuration guidelines for preventive maintenance functionality.

Please refer to the [Preventive Maintenance Overview](#) section in the Business User Guide for a conceptual overview of asset functionality.

## About Preventive Maintenance Configuration

Preventive maintenance helps manage the creation of work activities to maintain assets as per defined calendars and schedules. A Maintenance Schedule is associated with an Asset Type. A Maintenance Schedule provides a series of pre-defined Maintenance Triggers which define when to perform specific activities on the asset to ensure its continued operation.

Set up preventative maintenance for your assets by completing the following general steps:

1. Decide on the type of maintenance schedule needed for the asset:
  - There are four methods of calculating the maintenance schedule, as defined by the Trigger Type — calendar anniversary, calendar interval, runtime anniversary and runtime interval. The triggers are defined in terms of either time duration (for calendar-based methods) or runtime values.
  - Assets inherit their maintenance schedule from their specification. One specification can be linked to multiple maintenance schedules: one for each applicable asset situation. For example, you might configure a maintenance schedule for “in Service” and another for “in Stores”.
2. Plan for the preventive maintenance work by setting up a template work order.
3. Schedule the preventive maintenance work plan by setting up a maintenance schedule and maintenance triggers.
4. Trigger, execute and closeout the work.
  - Monitor algorithms generate Asset Maintenance Trigger Forecast records with the date at which the asset is due for the preventive maintenance.
  - The trigger creates the preventive maintenance work order and activity, (using the template work order) which then becomes part of the asset’s service history.
  - The forecast is based on the horizon days set on the asset monitor.
  - Forecast information is displayed on the **Asset Maintenance Trigger Forecast** zone in **Asset** portal.
5. Evaluate results and modify the preventive maintenance work plan as needed.

## About Reliability Centered Maintenance (RCM)

Reliability centered maintenance is a process which helps to ensure that assets run without fail. Define failure types, failure modes, failure repairs, failure components and failure profiles within the system, and use these elements together to form a comprehensive RCM system. Effective use of RCM can increase cost effectiveness, improve machine uptime, and help your organization to understand the level of risk you have undertaken with certain assets.

Generally an RCM process includes the following:

- Define Failure Modes Effects and Critically Analysis (FMEA):  
FMEA is a design tool to systematically analyze component failures and identify the resulting effects on system operations.
  - Identify assets, asset functionality, and the standards of performance for those assets.
  - List all possible ways that the assets could fail and provide the required functions
  - Provide detail on the types of events that could cause these failures
  - Identify the effects on the system when failures happen
- Apply RCM
  - Indicate the maintenance that can be performed to prevent or avoid such failures. This maintenance contains cost effective tasks to preserve functions that are important to the system.

In using RCM, your organization must define the following according to your business practices:

- **Failure Cause** — Represents the specific manner or way by which a failure occurs in terms of failure of the part. This may generally describe the way a failure occurs such as due to corrosion, debris, or distortion.
- **Failure Mode** — Describes what has failed.
- **Failure Repairs** — Indicates the repair that will be performed on the asset in case of failure.
- **Failure Component** — Indicates the part of the asset for which the failure has occurred.
- **Failure Profiles** — Groups valid failure types, failure modes and failure repairs into a profile which are attached to an asset type to define valid failure information for assets of that type.

## Designing Inventory Management

---

The following sections provide information on configuring inventory management.

### Understanding Inventory Configuration

This section provides guidelines for configuring inventory functionality.

Inventory is tracked, ordered, and received from vendors and allocated in the Inventory system. Materials are purchased and received, then as that inventory is allocated to work, the system passes this information back to the Work Management system so that maintenance managers and crews know that their parts are available.

Please refer to the [Inventory Management Overview](#) section in the Business User Guide for a conceptual overview of inventory functionality.

### About Stock Items, Stock Item Details and Storerooms

Stock items are defined as material resources that are held in storerooms and issued to activities that require the materials to be completed. The stock item record determines whether or not the type of stock can be purchased, repaired, tracked, and so on. Storerooms define the physical location where stock items are *stored*.

The Stock Item Detail (SID) record determines whether or not the type of stock can be purchased, repaired, tracked, and so on. An SID is a particular stock item in a particular storeroom. For example, in one storeroom, a particular stock item might be stored whereas in another storeroom that same stock item might only be direct purchased. The item would have a different SID in each storeroom. In cases where an item can only be direct purchased for activities, the storeroom only acts as the receiving dock and courtesy storage location until the workers from the activity come to pick them up.

## Types of Stock Items

Stock Item Details manage various types of stock items. The type determines how the stock is handled and managed, and when and how it can be ordered.

- **Regular Stock Items:**

Regular stock items are kept in inventory until they are needed.

These types of items are controlled by the following rules:

- created with the **trackable** flag set to “false” and the **lot managed** flag set to “false” on the stock item record
- can be repairable
- can be expensed at receipt
- can be designated as direct purchase items
- can be configured for reorder review, trackable items have a “safety stock” flag

- **Direct Purchase Stock Items:**

Direct purchase stock items can only be directly purchased when needed for work.

These types of items are controlled by the following rules:

- created with the **trackable** flag set to “false” and the **lot managed** flag set to “false” on the stock item record
- can be repairable
- can be expensed at receipt
- can be designated as direct purchase items
- can be configured for reorder review, trackable items have a “safety stock” flag

- **Lot Managed Stock Items:**

Lot managed items are tracked in a special way in the system. These types of items are fabricated, sold, and delivered in “lots” or “batches” of the item. This is significant when there are qualities or details about the item that might vary between lots. If this item is lot managed, the system tracks a lot number with any transaction that changes the quantity of the item for the storeroom. Since each received lot also has its inventory tracked independently, lot managed item details all have “child” stock item details for each lot that is received. This means a “master” stock item detail that is classified as “lot managed” on its stock item detail with many “lot” stock item details (one for every lot that is received). When an activity references lot managed items as a resource requirement, the master stock item detail is referenced. Lots only become relevant when material is issued from or received into inventory. This means the material issue for lot managed stock items reference a lot stock item detail, but the planned activity references the master stock item detail. During receiving of a lot managed stock item, the system verifies whether or not the storeroom has a lot specific SID for that lot number and creates one if one does not exist.

These types of items are controlled by the following rules:

- created with the **trackable** flag set to “false” and the **lot managed** flag set to “true” on the stock item record
- cannot be expensed at receipt
- cannot be designated as direct purchase items

- **Trackable Stock Items:**

Trackable stock items merit being individually identified and followed from initial receipt on-site through complete usage. These are typically large, expensive or specialized assets or materials which are inspected when they are received into inventory, repaired if they break, can be installed on other assets, and are otherwise carefully dispositioned in the system.

These types of items are controlled by the following rules:



- created with the **trackable** flag set to “true” and the **lot managed** flag set to “false” on the stock item record.
- can be repairable
- cannot be expensed at receipt
- cannot be designated as direct purchase items
- when configured for reorder review, trackable items have a “safety stock” flag
- indicate the specification for the trackable item on the stock item record

When defining stock items, crafts, equipment and miscellaneous resources, there is an option, Default Values to Activity, to define whether or not the information should be defaulted onto activities when resource requirements reference the resource. If this option is set to No for a stock item, then the item can only be added to stock item details as a direct purchase item.

## ABC Classification

Represents a system used to classify stock items based on their frequency of use and value. Class A items have the highest dollar usage value and are the most often counted while Class C items have the lowest dollar usage value and are the least often counted.

## Material Lifecycle

Materials are typically purchased and received into storerooms. Material Issue and Return tracks the movements of stock items as they go from storerooms to work activities and potentially back again. When materials are requested for work, and if the storeroom can fulfill the request, then material issue lines are created to specify one stock item each. Stock Transactions and Financial Transactions are created at time of issue to record both the inventory impact and the accounting for the use of the items.

There will be occasions when the material that was issued is not actually used and material returns are created and processed to return the items into inventory. This again will create reversing transactions on inventory and accounting.

## Material Disposition

Material Disposition is used to log and track the repair of assets. Typically there are two Material Disposition Lines, one for the old asset (Initial Relationship of ‘In Service’) that needs repair and the other for the new asset (Initial Relationship of ‘Issued’) that replaces the old asset.

## Physical Inventory

Physical inventory is the act of counting actual numbers of items on hand to reconcile with numbers calculated by the system. A Physical Inventory is related to a storeroom, and is one of three types of inventory (inventory, lot inventory, tracked inventory). The user defined Stock Category lookup can be used to group Stock Items together as needed.

## Inventory Adjustments

Inventory adjustments allow authorized users to adjust not only the inventory quantity, similar to Physical Inventory, but also inventory value such as average unit price.

## About Commodity Codes

Commodity Codes are an alternate way to describe stock items which can be helpful when you need to locate a specific stock code because they form a ‘logical’ segmented code for a stock item. Codes are segmented by commodity category, name, type, and composition. The **Stock Item** portal can be searched by commodity code where you can filter by a variety of criteria.

Commodity codes cannot be created directly. Instead they are constructed as a combination of the following:

- **Commodity Category** — Commodity Category is the most general description of the item. A sample category code might be “M” for a piece of mechanical equipment.
- **Commodity Name** — For each category, there is a unique list of Commodity Names. A sample commodity name for mechanical equipment might be a “Check Valve”.
- **Commodity Type** — For each category and name combination, there is a unique list of Commodity Types. A sample commodity type for a check valve might be a high-pressure pneumatic check valve.
- **Commodity Composition** — For each category/name/type combination, there is a unique list of Commodity Compositions. The composition code is used to indicate the primary material in the item (e.g. titanium alloy).

These attributes must be created in sequence within the commodity code hierarchy itself.

## Configuring Inventory Management

This section provides configuration tasks for inventory processing administrative support records.

Please refer to the [Inventory Management Overview](#) section in the Business User Guide for a conceptual overview of inventory functionality.

Refer to the [Understanding Inventory Configuration](#) section in the Administrative User Guides for configuration guidelines.

## Defining Storeroom Types

Navigate to the **Storeroom Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the storeroom type record:

Note: This list does not necessarily represent required fields.

- Location Class
- Parent Location Type (as applicable)
- Organization Type (as applicable)

You can also enter a parent location or organization to establish the position of storerooms of this type in the location hierarchy.

Storeroom type is referenced when creating Storerooms.

## Defining Commodity Codes

Use this procedure to create a commodity code with the four levels of category, name, type and composition.

**Prerequisites:** It is a good idea to have a detailed spreadsheet or plan for creating commodity codes for your organization.

1. Navigate to the **Commodity Code** portal.
2. Select the **Add Category** link.
3. Create the **Commodity Category** with a description.
4. Create a **Commodity Name** as needed and **Save**.

It is recommended that you create one name at a time. If you need to create additional names, create them after completing this process through the commodity composition for each name.

5. From the hierarchy tree, select the **Expand** icon next to the **Category** you just created.
6. Select one of the commodity names just created and click the **Edit** button.

7. Enter a **Commodity Type** as needed and **Save**.

It is recommended that you create one type at a time. If you need to create additional types, create them after completing this process through the commodity composition for each name and type.

8. From the hierarchy tree, select the **Expand** icon next to the **Category** you just created then expand the full hierarchy down to the **Type**.

9. Select the **Type** you just created and click the **Edit** button.

10. Enter **Commodity Compositions** as needed and **Save**.

This is the last level of the hierarchy, so you can create as many as you need for each category, name, and type.

## Designing Purchasing

---

The following sections provide information on configuring purchasing.

### Understanding Purchasing Configuration

This section provides configuration guidelines for purchasing functionality.

Oracle Utilities Work and Asset Management uses Purchasing functions to plan, initiate, and process purchases and purchasing contracts. It also manages vendors, designated buyers, and stock items.

Please refer to the [Purchasing Overview](#) section in the Business User Guide for a conceptual overview of purchasing functionality.

### About Buyers

Buyers are persons designated within your organization to make purchases and purchasing decisions.

Buyers can be referenced on purchase requisitions and purchase orders to provide an official point of contact for the purchasing transaction. Depending on your business process and configuration settings, buyers may also be able to make changes on approved purchasing documents that affect price and quantity without re-obtaining approval.

Buyers also manage automatic reordering.

### About Purchase Commodities

Purchase commodities are used to assign a specific buyer to a classification of stock items in the storeroom.

A buyer is specified on the purchase commodity so that when the commodity is then referenced on a stock item, the buyer association to that item is made. This allows buyers to be associated with a specialization over specific materials, equipment, etc. When items of that type are needed, the purchase commodity helps you to identify the buyer with that specialty.

### About Capability Types

Capability types refer to the types of products and services provided by a vendor. They are referenced on vendor locations to help identify what the vendor can provide, per location.

Since they are primarily informational, your organization may use capability types in a variety of ways. For example, you might run reports on which vendors provide particular goods or services to have this information handy for buyers' reference when making purchasing decisions.

## About Payment Terms

Payment terms represent the payment and credit agreement between your organization and the vendor such as the number of days your organization has to pay the invoice or the forms of payment that are accepted.

## About FOB (Freight on Board or Free on Board)

Freight on Board or Free on Board (FOB) represents the delivery charge terms negotiated for purchase orders from the vendor. This also includes the vendor or the seller's obligation to deliver goods at a designated place.

This information is referenced on vendor locations, blanket contracts and expedite orders to help define delivery terms.

## About Invoicing

Configuration parameters for invoicing functionality are set in Purchasing Master Configuration. Please also note the following:

### Master Configuration

Ensure that you create master configuration settings in Purchasing Master Configuration and Financial Master Configuration to process invoicing transactions.

### Default Approval Profile for Invoice Submission

If your implementation would like to define a default approval profile and status reason when users submit an invoice for approval processing, you must configure each individual user record with an approval title.

- Navigate to the user and click the **Characteristics** tab.
- In the **Characteristic Type of Invoice Approval Profile** field, enter an approval profile for characteristic value.  
To have the invoice go straight to approved status, select an approval profile based on a **No Approval Required** type of profile.

### Invoice Total Cost Validation

The Check Invoice Total algorithm validates that any manually entered invoice total matches the system calculated invoice total. If your implementation wants to enforce matching totals, you can introduce two new algorithms to enforce matching totals:

- Create a validation algorithm to make the Manually Entered Total field on the invoice required.
- Create a copy of W1-CKINVCTTL and modify it so that it produces an error rather than a warning.

This will force the user to correct the invoice before the system allows it to be processed. Once this copy is created, you should inactivate W1-CKINVCTTL.

### A/P Interface

If WAM owns invoice in your implementation and you need to interface invoices to your A/P system, the base product provides support to batch your invoices. You need to define an algorithm based on the W1-ADENTRYGP algorithm type and plug this algorithm onto the appropriate status of the invoice header BO, e.g. 'Posted'. This algorithm writes a record in **F1\_GEN\_PROC** table for that invoice with a batch code, which should be defined in the 'Invoice A/P Extract' parameter of the Purchase Master Configuration, and batch run number.

Please note that the base product doesn't deliver the actual batch process that extracts and formats the data. It's the implementation team's responsibility to develop such batch process. You can use the base GL batch extract as a template. The batch control for the GL extract is 'W1-GLDL'.

## Tax Rate Schedule Selection

As the default, the Tax Rate Schedule drop-down does not include the option to select a blank value. If your implementation requires the option to NOT select a tax rate schedule, you must create a Tax Rate Schedule such as "No Tax Rate" with Tax Rate Components containing factors with zero values. Users can then select this option to not include taxes on an invoice.

## Configuring Tax Characteristics for Invoice Lines

Your implementation must complete the following to ensure that tax component characteristics are captured on Invoice line items:

- For each required tax component, create a characteristic type with an entity flag of **Tax Component**.
- Introduce a flattened characteristic element for the characteristic type in the Invoice Line BO. In order to include them in the base BO's, the corresponding Invoice Info data area included in the BO can be extended to add the characteristic element(s).
- Configure a pre-processing algorithm for the BO to set default values for the tax characteristic element, if necessary.
- Use the **Retrieve Characteristic Value from Input Factor Characteristic List** characteristic source algorithm for the factor that will be referenced for the tax component.

## Cost Allocation

Note that if any of the cost allocation settings for invoicing are set as "general" the system creates a financial transaction with the following entries:

- **Freight:**
  - Debit Freight Expense; cost center/expense code from Purchasing Master Configuration
  - Credit Payables Clearing; cost center/expense code from Financial Master Configuration
- **Extra Cost:**
  - Debit Extra Cost Expense; cost center/expense code from Purchasing Master Configuration
  - Credit Payables Clearing; cost center/expense code from Financial Master Configuration
- **Discount:**
  - Debit Payables Clearing; cost center/expense code from Financial Master Configuration
  - Credit Discount; cost center/expense code from Purchasing Master Configuration

## Configuring Purchasing

This section provides configuration tasks for purchasing administrative support records.

Please refer to the [Purchasing Overview](#) section in the Business User Guide for a conceptual overview of purchasing functionality.

Refer to the [Understanding Purchasing Configuration](#) section in the Administrative User Guides for configuration guidelines.

## Defining Buyers

Navigate to the **Buyer** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the buyer record:

Note: This list does not represent required fields.

- User ID

Make sure that you activate the buyer after the record is created. Buyers are not available to reference on other records until they are active. Click the **Activate** button on the **Buyer** record.

## Defining Purchase Commodities

Navigate to the **Purchase Commodity** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the purchase commodity record:

Note: This list does not represent required fields.

- Buyer

## Defining Capability Types

Navigate to the **Capability Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Vendor Payment Terms

Navigate to the **Payment Terms** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining FOB (Freight on Board or Free on Board)

Navigate to the **FOB** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Exchange Rates

To properly set up exchange rates for alternate currencies, you must configure the following:

---

**NOTE:** Refer to the demo environment for the full example.

---

- Define the currency in the **Currency** portal.  
Demo Example:

- Canadian Dollars (CAD)
- Configure a **Characteristic Type** and **Value** to define a single value.  
Demo Example:
  - Single Value
- Configure a **Characteristic Source Algorithm** for the factor which will define the conversion.  
Demo Example:
  - Factor Retrieve Single Value Characteristic
- Configure a factor to define the exchange rate to use.  
Demo Example (U.S. to Canadian exchange rate):
  - **Factor:** USD to CAD Exchange Rate
  - **Factor Characteristic Type:** Single Value
  - **Factor Characteristic Value:** Single Value
  - **Effective Date/Time:** 01-01-2015
  - **Factor Value:** 1.2200000  
This value represents the value of your base currency converted to the foreign currency. In this case, 1 US dollar converted to Canadian.
- Add the factor to the **Purchasing Master Configuration** under **Alternate Currency Exchange Rates**.

When the currency is referenced on a vendor location and processing completes from issued purchase order through to receipt, the system calculates the converted unit price and total. The purchase order displays both values: In this example, **Order Total** in USD and **Converted Total** in CAD.

## Designing Resource Management

---

The following sections provide information on concepts related to managing resources.

### Understanding Resource Management Configuration

This section provides guidelines for configuring resource management functionality.

Resources encompass any object that contributes to the running of your organization. This includes hierarchies, employees, contacts, documents, standard notes and attachments, managing wage earning types and timesheets, and so on.

Please refer to the [Resource Management Overview](#) section in the Business User Guide for a conceptual overview of resource functionality.

### About Organization Types and Organizations

Organization types define the main attributes for organizations of a certain type.

Organizations are used for categorization and reporting for locations.

For example, your reporting may include a hierarchy that groups locations by a structure of parent and child organizations.

#### Valid Parent Types

Parent organization types define the set of valid location types and organization types that can be used to define a parent location or parent organization.

## About Contact Types and Contacts

Contact types establish parameters for contact of a specific type.

Contacts are referenced on locations to provide an associated person or business. The main attributes for contacts are managed in the **Contact** portal. For example, a contact might be referenced as the main person on site for assets or storerooms.

## About Document Types and Documents

Document types establish a basis for the types of documents that can be added within the system. This is referenced when creating a new **Document** record.

Document types might be used to classify the types of documents being used such as “work order doc”, “purchase order doc” to help users find the documents they need by type. Your business process might also dictate that document descriptions include such information to help with searching for documents to attach.

The **Documents** portal is used to track, store and manage electronic documents for use throughout the system. Each document record contains information such as file location, file type and a document description. Refer to [About Documents \(Attachments, Standard Notes and URL Attachments\)](#) in the User Guide for more information on the specific document types.

## About Service Classes

Service classes categorize activity types and indicate the type of work a shift is responsible for. Every crew shift is populated with a list of service classes that the crew shift is allowed to work. This is matched up with the service class indicated on the activity to help determine which crew shift can work the activity.

Every crew shift is populated with a list of service classes that the crew shift is allowed to work. This is matched up with the service class indicated on the activity to help determine which crew shift can work the activity.

### Renewal Work

The service class includes a flag to indicate whether or not the work type is work that is completed to restore or replace an asset towards its original size, condition, or capacity. The system separately calculates costs for renewal work.



## About Service Areas

Service areas define a logical boundary of your organization's territory. Every crew shift is populated with a list of service areas that the crew shift is allowed to work. This is matched up with the service area indicated on the activity to help determine which crew shift can work the activity.

## About Resource Unit of Measurement

Resource units of measurement (UOM) establish the acceptable units used to quantify resources such as stock items, materials, equipment, or other used by your organization.

## About Business Units

Business units are used as cost reporting entities for timekeeping expenses.

Employees are associated with a specific business unit so that when they report time charges, the costs are charged against that business unit.

The system uses double entry accounting. For example, when time is booked against an activity, the system debits the cost centers on the activity and credits an offset account. In this case the offset account would be the payroll cost center defined on the crew member/employee's business unit.

## About Employee Types and Employees

Employee types define the kinds of employees that can be defined for your organization. For example, your organization might use employee types to define regular employees, skilled employees, and novice employees.

Employee types are referenced on **Employee** records when they are defined.

Employee records are used to manage human resource level information for the workers within your organization with details such as name and personal information, work status, the main craft worked by the employee, their business unit, payroll information, hire date, default timesheet information, and so on. The employee record can also tie the employee to a service area where they are allowed to work as well as track training completed by the employee.

### Employees

Employees are defined with the Employee Type, and belong to a Business Unit (which defines a Payroll Clearing Cost Center). An employee who performs work will have a Principal Craft. The Employee can be set up with effective dated Employee Billable Rates, and the Employee Training keeps a record of courses the employee has taken. Employee Unavailability is used to indicate times when the employee is not available to work, due to leave, training or other Employee Unavailability Type.

## About Employee Unavailability Types and Employee Unavailability

Employee unavailability types define kinds of situations where employees may be unavailable such as general leave, jury duty, vacation, sick leave, and so on.

Employee unavailability types are referenced on **Employee Unavailability** records.

Employee unavailability records establish periods of time where an employee has been given leave from his or her regular job duties. This might be due to circumstances such as vacation, sick leave, attending a training, jury duty, and so on.

Logging employee unavailability allows the system to know when an employee is not available to perform work on the crews that he or she is associated with.

There are two classes of employee unavailability; leave (general leave, jury duty, vacation and sick leave) and work function (meetings and training). The base system has two employee unavailability type business objects, one for each class with the employee unavailability class defaulted in the schema. There are two corresponding employee unavailability business objects as well. Unlike leave, work function employee unavailability can have a list of employees defined on a particular employee unavailability record.

In terms of scheduling, employees indicated on the attendees list are not available as work resources. This means that during scheduling, the crafts available to the shift are reduced so that the employee on leave cannot be included. Timesheet generation is completed for the leave according to your settings and business practices.

## About Crew Shift Types, Crew Types, Crew Shifts and Crews

Crew shift types establish the kind of crew shift such as day, night, swing, and so on.

Please note the following with respect to configuring crews and shifts:

- Crew types define the kinds of crews that can be defined for your organization. For example, your organization might use crew types to define regular crews, skilled crews, and novice crews.
- A crew is a group of workers that can be assigned to complete work. Crews are used to group workers, schedule work, and help organize work schedules.
- A crew shift is a planned period of time in which a crew (one or more workers) is scheduled to perform work. For example, a crew shift might be defined as Crew A, consisting of Jack and his van, working on a specific date from 9 am – 5 pm.
- Crew shifts are associated with a crew shift type, that defines attributes common to all crew shifts of that type, including whether the shift is daily or weekly. For example, night shifts would have a different start time and duration than morning shifts, so they would have different shift types.
- Once a crew is created, shifts can be generated. Work order labor requirements can be assigned to the crew shift, and to a particular individual on the crew.
- Shift plan templates are used to automatically generate crew shifts based on a set of pre-defined shift criteria such as the shift type, the shift rotation, the defined calendar, and so on.
- Schedule Plan is a container for many daily schedules.
- Schedule Period defines if the crew shift type is a daily schedule or a weekly based planned schedule.
- Daily crew shift type determines whether crews receive extra pay (and the amount) when working such a shift.

## About Crafts, Equipment and Miscellaneous Resources

Crafts define a skill or work specialty for crew members such as mechanic, electrician, plumber, arborist, and so on.

When crafts are defined, a secondary craft can be added as part of the craft definition. Secondary crafts are regarded as additional or complementary skills to the primary craft.

Equipment is defined by your organization to be referenced as a resource type available for use to complete work. This might include machinery, tools or other equipment required to work activities (crane, bulldozer, drill, etc.).

Other resources are defined by your organization to be referenced as a resource type available for use to complete work.

Crafts, equipment and miscellaneous resources are referenced on activities as a material resource requirement. The system tracks usage and reports the costing associated with that usage against the total cost of completing the activity.

When defining stock items, crafts, equipment and miscellaneous resources, there is an option, Default Values to Activity, to define whether or not the information should be defaulted onto activities when resource requirements reference the resource.

# Configuring Resource Management

This section provides configuration tasks for resource management administrative support records.

Please refer to the [Resource Management Overview](#) section in the Business User Guide for a conceptual overview of resource functionality.

Refer to the [Understanding Resource Management Configuration](#) section in the Administrative User Guides for configuration guidelines.

## Defining Resource Unit of Measurements (UOM)

Navigate to the **Unit of Measure** — **Resource** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Organization Types

Navigate to the **Organization Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

You might define an organization type as a child organization type. In this case, the parent organization needs to be defined first.

Organization types are referenced when creating organizations.

## Defining Service Classes

Navigate to the **Service Class** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Service Areas

Navigate to the **Service Area** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Contact Types

Navigate to the **Contact Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

Contact type is referenced when creating contacts.

## Defining Document Types

Navigate to the **Document Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Business Units

Navigate to the **Business Unit** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the business unit record:

Note: This list does not represent required fields.

- Payroll Cost Center

## Defining Employee Types

Navigate to the **Employee Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

An employee type is referenced when creating an employee.

## Defining Employee Unavailability Types

Navigate to the **Employee Unavailability Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the employee unavailability type record:

Note: This list does not represent required fields.

- Cost Center
- Expense Code

An employee unavailability type is referenced when entering employee unavailability.

Employee unavailability is typically created for leave time or for work functions such as training or meetings. When you create the employee unavailability type, this is specified by the business object that is selected on the initial screen.

## Defining a Crew Type

Navigate to the **Crew Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining a Crew Shift Type

Navigate to the **Crew Shift Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Designing Work Management

---

The following sections provide information on concepts related to managing work including work requests, work orders, activities, and so on.

### Understanding Work Management Configuration

This section provides guidelines for configuring work management functionality.

Work management is the process by which work orders are requested, approved and activated, work is performed and then work is finished and closed. While the work is being performed, materials, equipment assets and labor are tracked.

Please refer to the [Work Management Overview](#) section in the Business User Guide for a conceptual overview of work management functionality.

### About Planners and Maintenance Managers

Planners and maintenance managers are defined for your organization as personnel authorized to perform certain functions or who are responsible for overseeing specific work. For example, planners are usually responsible for planning work and materials for assets. Maintenance managers are typically more specifically involved with an aspect of work such as managing hazardous materials or specialized equipment. The usage of these roles depends entirely upon your business practices.

Every planner and maintenance manager references the User ID of the assigned person. Work statistics on the **Home Page** can be configured to display by planner or by maintenance manager.

### About Activity Types and Activities

Activity types define the main attributes for activities of a certain type and act as a model and template for **Activity** records.

The activity type specifies the applicable service class, scheduling information as applicable, default requirements, how to handle errors or exceptions, and completion details for activities of the type. The **Service History Type** linked to the Activity Type acts as the 'default' for creating Service History records as a result of an Activity. This is used when reviewing an asset's service history for preventive maintenance purposes.

Identifier fields on an activity type define additional attributes, determined by your organization, that can be assigned to further characterize an entity. They are used on the search screen to add additional search criteria.

An activity is a task which needs to be completed. Activities manage asset changes, work, inspections maintenance and other actions that are performed on assets or components.

Each activity on a work order is defined with one or more "resource requirements" which define resources that are needed to perform the activity. Typically planned work orders specify resource requirements on the related activities, and approvers can verify the requirements before the work order is approved. Requirements may be categorized as materials, labor, equipment or other. Refer to [About Activity Resource Requirements](#) for more information.

## Cost Tracking

Cost tracking is enabled, as default, for all work activities. For field activities (activities sent to an external system), the option to track costs can be enabled or disabled on each activity type. To configure the default for this setting, use the **Schema Constants Feature Configuration**.

---

**NOTE:** For Oracle Utilities Operational Device Management users also using Oracle Utilities Work and Asset Management: Activity types are delivered in ODM with the cost tracking default set to No on all ODM activity types.

---

## Capital Work

Work orders that are worked by outside contractors are considered “capital work”. If your implementation will use contractors to complete activities, ensure that you configure:

- a work category of “capital”
- external system to the contractor
- outbound activity type with the external system set and at least one required service history completion event type

## About Work Priority and Criticality Fields

Work priority and criticality custom lookups require special configuration since they can be searched by range and by greater than and/or less than values.

Your organization may also implement the ability to do a match function on these fields. For example, you can configure the system to multiply location criticality by activity work priority to derive a usable value.

To this end, you must configure these lookups, `WORK_PRIORITY_FLG` and `CRITICALITY_FLG`, with the following considerations:

- The **Field Value** or code must be a 2-digit number.  
This is required even if your codes do not exceed 09. They must be 2 digits for sorting to function properly.
- The description of the value must have the same two digits at the beginning. Since lookup values show up as drop-downs and drop-down values are sorted by description, starting the description with the code will ensure proper sorting.

Refer to the demo environment and navigate to these lookups to review the demo example.

## About Completion Event Types

Completion event types are used to define completion events of a particular type.

For example, completion event types might be configured for several types of asset processing such as asset installation, asset removal, replication, service history creation, and activity creation.

Completion event types are referenced on activity types to indicate which completion events should be included as part of the lifecycle of activities of that type. As examples, this can affect when service history is created for an activity, and the disposition of the assets on an activity as the activity is processed.

Note that if you add activity types to the completion event type, you cannot set whether or not the completion event type is required for the activity type. Add the completion event type to the activity type using the **Activity Type** portal to also set the completion event types as optional or required.

The base package completion event type business objects can be used out of the box, however, it is likely that your implementation will want to configure special business objects to capture specific system events.

It is also important to note where there are optional completion event types might need to be used to capture incidental or unexpected events such as:

- Create follow-up activity

- Create service history
- Create attachment (asset and/or location)

Your implementation may want to carefully consider these types of events and possibly add them as optional on every completion event type where they might be needed.

## About Inbound Communications

The **Inbound Communication** portal provides details related to the information received from an external system such as Oracle Utilities Mobile Workforce Management. Use inbound communication to create a completion event using information defined within the raw message coming from an external system.

Inbound communication events receive completion event information from the external system and then process the necessary transactions to see the activity or other event to completion.

### Inbound Communication Configuration Notes

An inbound communication business object must be created for each type of message to be received from an external system. For most external systems, this is based on the types of messages the system is designed to send.

You must also create an XAI Inbound Service for each type of message to be received from an external system. XAI inbound services define the details of how messages are received from an external system, including the inbound communication business object (or business service or service script) to be invoked when the response message is received. As in the case of inbound communication business objects, the set of XAI inbound services you need to create is based on the types of messages the system is designed to send.

## About Projects and Subprojects

Projects provide a way to group work so that it can be planned and charged against large efforts. Typically, a project will be a task that is out of the ordinary, or that requires extra planning and resources. Examples would be adding new facilities or conducting a major overhaul of an assembly line.

### Projects in Context

Please note the following with respect to using projects in your implementation:

- Projects can be linked to many of the transactional records in the system. This is primarily used for cost reporting. They can be linked to many cost centers.
- Projects can be organized into a hierarchical structure.
- Projects can be configured to require approval processing.
- Key data components on a project include the estimated and actual start/end dates, budgeted and actual amounts, and the project supervisor.

### Project Categories

Your organization can define project categories to create classifications for projects such as maintenance, capital, construction, and so on.

## About Timekeeping

Timesheets are used to record the amount of time that workers (labor resources) spend on activities or work orders. Once charges are entered, processing allows employees to receive proper compensation for their work and labor charges are applied to the appropriate cost buckets.

Please note the following with respect to using scheduling in your implementation:

### Time Periods

Your organization must create time periods to define valid from and to dates for labor costing and timesheet reporting. Time periods have to be distinct and cannot overlap. When a new timesheet is created, the system defaults to the current time period.

### Labor Earning Types

Labor earning types define earnings codes for your organization which includes an adder and a multiplier to calculate labor costs.

### Overtime Types

Overtime types define overtime codes for your organization which includes a multiplier to calculate the premium rate that workers earn for overtime.

### Deferred Monitor

Once a timesheet is approved, user will need to run batch job Monitor Move Timesheet to Next Default Status to transition the timesheet to Posted status. The Timesheet will transition all the Timesheet Details in the initial state to Posted and when all the non cancelled timesheet detail records have been posted and the timesheet's time period end date is less than current date. Reason for this is that we don't want to post future timesheet detail records and we shouldn't show the timesheet as posted until all the timesheet detail records for the timesheet have been posted.

## Configuring Work Management

This section provides configuration tasks for work management administrative support records.

Please refer to the [Work Management Overview](#) section in the Business User Guide for a conceptual overview of work management functionality.

Refer to the [Understanding Work Management Configuration](#) section in the Administrative User Guides for configuration guidelines.

## Defining Maintenance Managers

Navigate to the **Maintenance Manager** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the maintenance manager record:

Note: This list does not represent required fields.

- User ID



## Defining Planners

Navigate to the **Planner** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the planner record:

Note: This list does not represent required fields.

- User ID

## Defining Activity Types

Navigate to the **Activity Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the activity type record:

Note: This list does not represent required fields.

- Service Class
- Completion Event Type (should be configured for the specific type of activity)

## Defining Completion Event Types

Navigate to the **Completion Event Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the completion event type record:

Note: This list does not represent required fields.

- Activity Type

## Defining Time Periods

Navigate to the **Time Period** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Labor Earning Types

Navigate to the **Labor Earning Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Overtime Types

Navigate to the **Overtime Type** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Project Categories

Navigate to the **Project Category** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

The project category is referenced when creating project.

## Designing Financials

---

The following sections provide information about the portals and objects related to financial transactions throughout the system.

### Understanding Financials Configuration

This section provides the configuration guidelines related to financial entities throughout the system.

Financials management involves creating and managing entities such as expense codes, distribution codes, cost centers, cost categories, tax rate schedules, accounting calendars, , and so on.

Financial Transactions (FTs) are created in quite a few situations, in general associated with the purchase or use of resources, such as the time cost of labor, the invoicing or receipt of materials, etc. There is always some system transactional record that drives the creation of the FT, which is referred to as the **Creating Sibling** record. The system also captures that sibling record on the FT. Each FT belongs to a single Accounting Period, which must be open at the time of creation, and is related to a particular currency.

Each FT consists of two or more balanced financial transaction general ledger (FT GL) lines (at least one debit and one credit). This information can be sent to your organization's general ledger. On the FT GL record, the system can capture an associated Project, Cost Center and Expense Code, as applicable. Each transaction is booked to a particular GL Account code via a distribution code and its associated GL Account Detail. The distribution code specifies how the GL Account string is to be constructed via the **GL Construction Algorithm**. The **Financial Master Configuration** specifies which **Distribution Code** to use for particular scenarios.

Please refer to the [Financial Transactions Overview](#) section in the Business User Guide for a conceptual overview of financials functionality.

### About Accounting Calendars

Your accounting calendar determines the accounting periods where financial transactions are posted. An open accounting period is the time period during which costs accumulate. You can configure these periods to match the accounting periods used by your general ledger application.

As part of your general configuration your organization will define the accounting calendar in the **Accounting Calendar** portal then designate that calendar as the default in **Financial Master Configuration**. Only one accounting calendar is allowed for the entire system. Accounting calendars include a series of accounting periods with a start and end date. They also specify to and from dates when the calendar is “open” or when transactions can be posted to that calendar.

## About Currency

The overall system uses the base currency defined in **Installation Options**, and all financial transactions use the base currency. However, the purchasing system supports other currencies which can be defined in the **Currency** portal.

Enter standard currency codes such as USD or GBP, a description, symbol and the number of decimal places to use. The currency code will then appear as an option wherever a currency code is required on purchasing records, such as on a blanket contract or purchase order. Any currency codes used in the purchasing system which are different from the base currency are converted into the base when the system creates financial transactions.

## About Colors

Your organization can use the **Colors** portal to define custom colors used to represent transactions and other entities in various portals. For example, colors defined here are used to distinguish cost categories as they are represented in graphs and/or pie charts.

The **Color** record provides a hex value, description and color sample. Once this is defined, the color can be specified on the Cost Category record as the designated color for that category. Each color should only be referenced by one cost category.

## About Cost Categories

Cost categories group expenses or costs into logical buckets for reporting purposes.

The following details apply for cost categories:

- Each expense code references a cost category.
- Each cost category can be assigned a specific color so that the category is represented by that color in graphs and pie charts throughout the system. Each color should only be referenced by one cost category.

## About Cost Centers

As costs are incurred, the system tracks them and accumulates them at different levels or cost centers within your organization. Cost centers are part of the accounting hierarchy.

Cost center records display financial statistics such as a budget amount, year to date actual amount, and committed amount for the cost center. These amounts represent the fiscal year rather than the calendar year. A monthly statistics section also allows you to display more detailed information by month in a graph.

General ledger account numbers are a concatenation of the cost center, project, and expense code, as applicable.

## Allow User Charge

User records that result in costs or charges need to reference a cost center that has this value set to Yes. For example, when users create activities, the referenced cost centers must “allow user charges”. This value is set to No on cost centers that are specifically designated for general ledger or system generated costs. For example, an inventory cost center is referenced on storerooms to balance the debit and credit sides of the general ledger (issue materials for an activity to charge the debit side and reduce the inventory value adjusts the debit side).

## About Expense Codes

Expense codes are codes assigned to a financial expenditure indicating which payment category the cost falls into. They are part of the accounting hierarchy.

Expense codes reference a cost category, resource class, and an account segment. This means that each expense code is only valid for a particular resource class (labor, material, equipment, or other), and must be defined accordingly. General ledger account numbers are a concatenation of the cost center, project, and expense code, as applicable.

## About Factors and Factor Values

Factors are defined by your organization as a tax component needed to calculate tax values related to anything which incurs costs for your organization and must be taxed (such as purchases and resources used to complete activities). For example, you might define different factors for city tax, state tax and federal tax. Each type of factor would define a factor values list providing numerical values used to calculate the tax. The factor is then referenced on a **Tax Rate Schedule**.

## About Distribution Codes

Distribution codes simplify the process of generating accounting entries by defining valid combinations of values for “charts of accounts”.

The system uses the distribution code as a flexible way to construct the final account number to send to the GL system. That algorithm can be coded to construct the final GL account number from data available anywhere in the system.

When creating distribution codes you specify the general ledger account algorithm that will be used by the system when it interfaces financial transactions that reference this distribution code to the general ledger. This account algorithm will construct the actual GL Account Number. For example, the base algorithm constructs the account by concatenating account segments from cost center, project and expense code.

If some accounts are static and do not need dynamic construction, the entire account number can be defined in the **GL Account Detail** and a simple algorithm can retrieve the account number from the same distribution code.

## About Tax Rate Schedules

A **Tax Rate Schedule** defines the type of taxes that are applicable at various instances in time for related resources (materials, labor, equipment, and other). The tax rate can be defined on a resource type, but this can be overridden for specific storerooms if necessary.

Your implementation will need to configure attributes so that tax rates are included on purchase requisitions, purchase orders and blanket contracts.

## Configuring Financials

This section provides configuration tasks for financials administrative support records.

Please refer to the [Financial Transactions Overview](#) section in the Business User Guide for a conceptual overview of financials functionality.

Refer to the [Understanding Financials Configuration](#) section in the Administrative User Guides for configuration guidelines.

## Defining Accounting Periods

Your accounting calendar determines the accounting periods where financial transactions are posted. An open accounting period is the time period during which costs accumulate. You can configure these periods to match the accounting periods used by your general ledger application.

As part of your general configuration your organization will define the accounting calendar in the **Accounting Calendar** portal then designate that calendar as the default in **Financial Master Configuration**. Only one accounting calendar is allowed for the entire system. Accounting calendars include a series of accounting periods with a start and end date. They also specify to and from dates when the calendar is “open” or when transactions can be posted to that calendar.

When defining your accounting calendar, take the following into consideration:

- In general, your organization’s finance department should handle the creation of accounting periods.
- Number of periods cannot be less than 0.
- Defined accounting periods must be less than or equal to the Number of Periods.
- You cannot define a calendar with gaps between the periods. Each period must follow the previous period consecutively. For example, you may forget to include February 29th for leap years.
- You cannot define a calendar with gaps between years. For example, the fiscal year cannot jump from 2013 to 2015 in consecutive periods.
- Dates cannot overlap between periods.
- Period start dates should be on or before the defined end dates.
- The open from date should be on or before the open to date.

### Example:

Fiscal Year	Accounting Period	Description	Start Date	End Date	Open from Date	Open to Date
2015	1	Period 1	06/01/2014	06/30/2015	06/01/2014	07/10/2015
2015	2	GD Period 2 (July) FY2015	07-01-2014	07-31-2014	07-01-2014	08-10-2014
2015	3	GD Period 3 (August) FY2015	08-01-2014	08-31-2014	08-01-2014	09-10-2014
2015	4	GD Period 4 (September) FY2015	09-01-2014	09-30-2014	09-01-2014	10-10-2014
2015	5	GD Period 5 (October) FY2015	10-01-2014	10-31-2014	10-01-2014	11-10-2014
2015	6	GD Period 6 (November) FY2015	11-01-2014	11-30-2014	11-01-2014	12-10-2014
2015	7	GD Period 7 (December) FY2015	12-01-2014	12-31-2014	12-01-2014	01-10-2015
2015	8	GD Period 8 (January) FY2015	01-01-2015	01-31-2015	01-01-2015	02-10-2015

Fiscal Year	Accounting Period	Description	Start Date	End Date	Open from Date	Open to Date
2015	9	GD Period 9 (February) FY2015	02-01-2015	02-28-2015	02-01-2015	03-10-2015
2015	10	GD Period 10 (March) FY2015	03-01-2015	03-31-2015	03-01-2015	04-10-2015
2015	11	GD Period 11 (April) FY2015	04-01-2015	04-30-2015	04-01-2015	05-10-2015
2015	12	GD Period 12 (May) FY2015	05-01-2015	05-31-2015	05-01-2015	06-10-2015

## Defining Account Segments

Account segment fields appear on various records such as on cost centers, expense codes and projects. This is a free form text field which builds the general ledger account which is shown in the financial transaction log when costs are posted.

Your business practices determine how you set up account segments and your interface with your general ledger. Use the general ledger construction algorithms to set up account segments. The Simple GL Construction algorithm (W1-SIMPLEGL) retrieves the current effective GL number defined on the distribution code and use the retrieved value as the general ledger account number. Beyond this, the remaining GL algorithms are used to construct the GL account number by concatenating the account segments from Cost Center, Project and Expense Code. The **Delimiter** parameter on each algorithm is the separator between the segments. The **Project Placeholder** parameter on each algorithm is used as the segment for project if a GL entry is not related to a project.

## Creating Accounting Calendars

Navigate to the **Accounting Calendar** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

## Defining Currency

Navigate to the **Currency** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** Once currency must be defined in **Installation Options** as the system's base currency.

## Defining Colors

Navigate to the **Color** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

Cost categories will need to reference a *unique* color.

## Defining Cost Categories

Navigate to the **Cost Category** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the cost category record:

Note: This list does not represent required fields.

- Color — you must have a unique color to represent each cost category.

Expense codes will need to reference a cost category.

## Defining Cost Centers

Navigate to the **Cost Center** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the cost category record:

Note: This list does not represent required fields.

- Expense Code — Add Valid **Expense Codes** to the cost center after it is defined, as required by your business practices. This is determined by the **Financial Master Configuration**.

## Reviewing Cost Center Statistics

Use this procedure to review cost center statistics.

1. Navigate to the appropriate **Cost Center** record in **Edit** mode.
2. Search for and select the cost center with the statistics to review.
3. Click the **Accumulate Monthly Statistics** button.

The system renders the graphs or pie charts to represent the statistics in the **Cost Center Monthly Statistics** section.

## Deactivating Cost Centers

Use this process to deactivate a cost center.

Cost centers must be in active status to be referenced elsewhere in the system.

1. Navigate to the appropriate **Cost Center** record in **Edit** mode.
2. Search for and select the cost center that should no longer be used.
3. Click the **Deactivate** button.

## Defining Expense Codes

Navigate to the **Expense Code** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the expense code record:

Note: This list does not represent required fields.

- Cost Category

Expense codes must be in active status to be used. Cost centers reference a list of valid expense codes, as required by your business practices.

## Deactivating Expense Codes

Use this process to deactivate an expense code.

Expense codes must be in active status to be used.

1. Navigate to the appropriate **Expense Code** record in **Edit** mode.
2. Search for and select the expense code that should no longer be used.
3. Click the **Deactivate** button.

## Defining Factors

Navigate to the **Factor** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** None.

Add factor values in the **Factor Characteristic Values** section after creating the factor.  
Tax rate schedules will need to reference a factor.

## Defining Distribution Codes

Navigate to the **Distribution Code** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

## Defining Tax Rate Schedules

Navigate to the **Tax Rate Schedule** portal under **Admin**, select **Add**, complete the required fields according to your business practices and **Save**.

**Prerequisites:** The following are referenced by the tax rate schedule record:

Note: This list does not represent required fields.

- Factor
- Cost Center
- Expense Code

Your implementation will need to take steps to include attributes which determine the tax rates on purchasing record line items.

Complete the steps, as indicated below, to properly configure tax rate schedules and include tax rates on purchasing documents. Refer to the demonstration environment for an example and to the *Oracle Utilities Application Framework Administration Guide* for more information as needed.

1. Navigate to **Admin #Characteristic Type** and create a tax characteristic type.  
See WD-TAXST in demo.
2. Navigate to **Admin #Factor** and create factors under the tax characteristic type.  
See WD-StateTax in demo.

Add factor values to the factor as needed.



3. Navigate to **Admin #Tax Rate Schedule** and create a tax rate schedule

See WD-Tax1 in demo.

Add tax components for the schedule by using the previously created factors.

4. Navigate to **Admin #Data Area** to introduce a flattened characteristic element for the characteristic type in the Purchase Requisition Line business object.

See %tax in Data Area demo.

Note that this characteristic type should be the same one referenced by the Tax Component's Factor and that it should be configured for Tax Component and Purchase Requisition Line entities. You can simply create a new data area that extends the Requisition Info data area of the base business object if you don't want to create a new business object implementation.

Sample Schema:

```
< schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  < sequence mapField="SEQ_NUM"/>
  < rateSchedule mapField="TAX_RATE_SCHED_CD" uiHint:select="table:W1_TAX_RATE_SCHED"
    fkRef="W1-TXRTS"/>
  < effectiveDate mapField="EFFDT"/>
  < factor mapField="W1_FACTOR_CD" uiHint:select="table:W1_FACTOR"/>
  < factorValue mapField="W1_FACTOR_VALUE"/>
  < charType mapField="CHAR_TYPE_CD" uiHint:select="table:CI_CHAR_TYPE"/>
  < charValue mapField="CHAR_VAL"/>
  < taxAmount mapField="W1_TAX_AMT"/>
</ schema>
```

5. Update the flattened tax element to trigger the tax calculation service.

Once configuration is complete, you can navigate to a purchase requisition line item to see the result.

## Integrations

---

This chapter provides information related to integrating with other applications.

### About Integrations

Oracle Utilities Work and Asset Management can be configured to integrate with external systems through inbound and outbound messaging, or communications. Communications are records of messages sent between the source system and an external system and are most often created as a result of an activity.

Refer to the chapter titled “External Messages” in the *Oracle Utilities Application Framework Administration Guide* for more details.

### Specific Product Integrations

Integrations with other Oracle Utilities applications help to provide clients with a complete application solution. This section identifies the specific product integrations that include Oracle Utilities Work and Asset Management, including:

- **BI Publisher Enterprise:**

BI Publisher Reports allow users to generate and print reports for certain entities, such as operational data, activity reports, purchasing reports, inventory reports, and so on.

Refer to the *Oracle Utilities Work and Asset Management Installation Guide* for information on installing and configuring.

Refer to [#unique\\_1001](#) for more information and configuration details.

- **Esri ArcGIS:** The GIS Integration was deprecated from this release. Please upgrade to v2.1.1 Service Pack 1 to use this functionality.
- **Mobile Workforce Management:**  
Oracle Utilities Work and Asset Management integration with Oracle Utilities Mobile Workforce Management is managed through the Oracle Utilities Field Work Process Integration Pack.  
  
Refer to the Oracle Process Integration Pack for Oracle Utilities Field Work 12.1 - Implementation Guide for more information and configuration details.

## Defining External Systems

You must create an **External System** record for each external system to which Oracle Utilities Work and Asset Management will communicate. Each external system defines a set of outbound message types that will be sent to that system. Refer to the documentation for your specific integration for more information.

## Data Synchronization for Integrations

In most integrations, synchronization of data between the two systems is supported through a set of business objects, master configurations, batch controls, and pre-configured XAI Inbound Services.

### Types of Synchronization Requests

Data synchronization is performed via synchronization requests sent from the external system via a middleware integration component. The application supports three types of synchronization requests:

#### Initial Synchronization Requests

Initial synchronization requests are used during the initial configuration of the integration. They facilitate import of data that synchronizes service points, locations, and contact information.

#### Ongoing Synchronization Requests

Ongoing synchronization requests are used when updating existing data in Oracle Utilities Work and Asset Management based on changes in corresponding data in the external system. Ongoing synchronization requests can be used to update locations, assets or components and to send activity completion information.

#### Composite Synchronization Requests

Composite synchronization requests are requests that contain synchronization requests for multiple types of data within a single request. For example, a composite request could contain requests to update a location, and complete an activity. This supports situations where multiple types of data must be updated based on a single change in the external system.

Navigate to **Admin > Business Object** and search for “W1%sync%” to locate business objects used for data synchronization.

#### Base Package Synchronization Request Business Objects

Navigate to **Admin > Business Object** and search for “W1%sync%” to locate these objects.

#### Master Configurations for Data Synchronization

Master configurations are used to define aspects of the synchronization process, including resolution of foreign keys and the type of synchronization business objects to use for each type of data being synchronized.

Navigate to **Admin > Master Configuration** to review or modify master configuration settings.

## Batch Controls for Data Synchronization

Batch controls perform processing for initial synchronization requests such as allocating keys to data, resolving foreign keys, and loading data (instantiating business objects representing entities such as contacts, assets, etc).

“Initial Sync Request - Resolve Keys [maintenance object]” batch controls invoke a generic maintenance object transition process to invoke the “Resolve Keys - Initial Sync” algorithm for synchronization requests of the appropriate type.

Parameters used by “resolve keys” batch controls include:

- **Maintenance Object:** (Required) the maintenance object (asset, contact, etc.) to be processed. This must be set to the Sync Request maintenance object for the batch control.
- **Restrict By Batch Code:** Restricts processing to synchronization requests whose current state is linked to this batch code.
- **Restrict By Business Object:** Restricts processing to synchronization requests linked to this business object.
- **Restrict By Status Code:** Restricts processing to synchronization requests of this status (default: KEY\_ALLOCATD).
- **Max Errors:** Specifies the maximum number of errors allowed before the process exits.

Refer to the [Oracle Utilities Work and Asset Management Batch Controls](#) section for a list of these synchronization batch controls.

## Batch Scheduling for Data Synchronization

The following sequence indicates the order in which the batch controls on the Initial Sync Request BO life cycle should be executed.

1. Transformation / Schema Validation Job
2. Key Allocation Job
3. Foreign Key Resolution / BO Validation Job (dependent on ALL Key Allocation Jobs finishing)
4. Load Job

## XAI Inbound Services for Data Synchronization

XAI inbound services are used to facilitate invoking the Sync Request Seeder business object by the middleware components upon receipt of a synchronization request.

Navigate to **Admin > XAI Inbound Service** to review or modify master configuration settings.

## Understanding Synchronization Request Processing for Integrations

This section provides an overview of the processing that takes place when a synchronization request is sent. For each step in the process, the table below provides a brief description of the processing that takes place, and lists the specific objects involved.

1. The external system sends a synchronization request to the middleware integration layer.  
For example, consider a request to update information about an asset installation.
2. The middleware components transform the request from the external system to the format, to the format used by Oracle Utilities Operational Device Management (this format is based on the business object schemas of the synchronization request business objects).
3. The middleware component invoked the appropriate XAI Inbound Service, and sends the transformed request.  
XAI Inbound Service: W1-SyncRequestInbound (mapped to the W1-SyncRequestSeeder business object)

4. The XAI Inbound Service invokes the Sync Request Seeder business object, which in turn, determines which synchronization request business object to create (based on the type of data in the synchronization request and the Seeder Sync Master Configuration).  
Synchronization Request BO: W1-SyncRequestSeeder
5. For initial synchronization requests, background processing creates master data for each synchronization request, including the following steps:
  - Data Transformation / Schema Validation
  - Allocate Keys
  - Resolve Foreign Keys / Validate Business Object
  - Load Data

## BI Publisher Integration

BI Publisher Reports allow users to generate and print reports for certain entities, such as operational data, activity reports, purchasing reports, inventory reports, and so on.

## BI Publisher Integration Setup Steps

Complete the following to configure your BI Publisher integration.

### Master Configuration and Batch Configuration

Set master configuration parameters and batch job submission parameters according to your implementation requirements. Refer to the embedded help for details.

### Report Definition

Configure Report Definition settings to allow your implementation to import and export definition report records from one server to another. Navigate to the **Report Definition** portal under **Admin** and select **Search**.

Refer to [Defining Report Definitions](#) for information on the parameters and settings.

## Control Table Setup Sequence

---

This section provides a checklist of the primary data that must be configured as part of your implementation. This can be used as a useful map of the relationship between objects in the system.

It includes:

- Listing of the objects that must be defined as part of the setup process
- Identification of the order in which objects should be defined and any prerequisites for setup
- Identification of any custom lookups that must be defined in relationship to each setup object
- Identification of objects associated with or referenced by each setup object

### Setup Sequence

Use the following tables as your roadmap for creating system data.

Here we list the order in which you perform tasks and the pages you'll use to set up your system. The order is important because some information must exist before other information can be defined (i.e., many dependencies exist).

**NOTE:** The sequence in which you configure system objects is important. By following this sequence carefully, you can streamline the configuration process and reduce the amount of time required for setup.

In the setup tables, the **Sequence** column displays the following codes:

- L1 = No setup prerequisites. These should be defined before the remaining objects.
- L2 = Some L1 prerequisites. These should be defined after all L1 objects and before L3 objects.
- L3 = Defined after all L1 and L2 objects.
- L4 = Defined after all L1, L2, and L3 objects.
- L5 = Defined after all L1, L2, L3, and L4 objects.

## Control Data Setup

To implement the system, you must set up your organization's business rules in "control tables". Setting up these tables is time-consuming because we allow you to tailor many aspects of the system to meet your organization's requirements. We strongly recommend that you take the time to document how you plan to set up all of these tables before you use the following roadmap to enter the control data. Time spent understanding the interrelationships between this data will reap the rewards of a clean system that meets your current and long term needs.

### Oracle Utilities Application Framework Administrative Data Setup

These setup tasks are typically performed by an administrative user and are accessed from the **Admin** menu. Most of these objects establish global configuration. They are defined at implementation and usually do not change again.

Sequence	Object	Description	Prerequisites	Framework Administration Guide Reference
L1	Country	Your organization's country.	None	<a href="#">Defining Countries</a>
L1	Currency	Your organization's native currency.	None	<a href="#">Defining Currency Codes</a> The overall system uses the base currency defined in <b>Installation Options</b> , and all financial transactions use the base currency. However, the purchasing system supports other currencies which can be defined in the <b>Currency</b> portal.
L1	Display Profile	Controls how dates, times, and numbers displayed.	None	<a href="#">Defining Display Profiles</a>
L1	Language	The language to use for this implementation.	None	<a href="#">Defining Languages</a>
L1	Time Zone	Your organization's base time zone.	None	<a href="#">Defining Time Zones</a>
L1	To Do Role	Used to associate users with To Do entries.	None	<a href="#">Defining To Do Roles</a>

Sequence	Object	Description	Prerequisites	Framework Administration Guide Reference
L1	Work Calendar	The work calendar for your organization, which identifies your public holidays.	None	<a href="#">Defining Work Calendar</a>
L1	Geographic Type	If your company uses geographic coordinates for dispatching or geographic information system integration, you need to setup a geographic (coordinate) type for each type of geographic coordinate you capture for assets.	None	<a href="#">Defining Geographic Types</a>
L1	Outbound Message Type	Establishes a categorization for outbound messages.	Business Object and Schema	<a href="#">Defining Outbound Message Types</a>
L1	Extendable Lookups	The "lookup" table is a single table that holds valid values for many columns on many tables. Extendable lookups are custom fields where valid values are limited to a predefined list.	None	<a href="#">Defining Extendable Lookups</a>
L1	Lookups	The "lookup" table is a single table that holds valid values for many columns on many tables. Lookups are custom fields where values can be added. Only lookups where the "Custom" flag is checked can be modified.	None	<a href="#">Defining Lookup Options</a>
L2	Feature Configuration	Add a feature configuration of feature type 'Schema Constants' and set the value for 'Default Country'.	Country	<a href="#">Defining Feature Configurations</a>
L2	Installation Options	Control various aspects of the system. You must set up 'Address Information' and 'Global Context' algorithms.	Time Zone, Language, Currency	<a href="#">Defining Installation Options</a>
L2	Master Configuration	Enables an implementation to capture various types of information in the system.	None	<a href="#">Defining Master Configurations</a>

Sequence	Object	Description	Prerequisites	Framework Administration Guide Reference
L2	To Do Type	Used to define types of To Do Entries.	To Do Role	<a href="#">Defining To Do Types</a>
L2	Global Context Algorithm	You will need to set up an algorithm that populates global context values. The global context is used by various zones in the system to display relevant data. This algorithm is plugged-in on the installation record.		<a href="#">Global Context Overview</a>
L2	Characteristic Types & Values	If you need to introduce additional fields to objects that were delivered with minimal fields, you can add a characteristic type for each field you want to capture.	None	<a href="#">Setting Up Characteristic Types &amp; Values</a>
L2	External System	If you are using an integration you must configure the external system in this portal.	Outbound Message Type	<a href="#">External Systems</a>
L3	Phone Type	Defines the phone number format to use throughout the system.	Phone Number Format Algorithm	<a href="#">Defining Phone Types</a>

## Security Setup

Refer to [Defining Security & User Options](#) for more information.

Object	Description/Notes	Framework Administration Guide Reference
Application Service	All base package transactions are automatically populated	<a href="#">Defining Application Services</a>
Security Type	Only needed if field-level security is being implemented	<a href="#">Defining Security Types</a>
User Group	<p>A group of users who have the same degree of security access.</p> <p>One user group, ALL-SERVICES, is automatically setup. It references all other application services and a single user called SYSUSER .</p> <p>WAM also provides two additional user groups, WAM_SERVICES and ODM_SERVICES.</p> <p>Note, you may be able to import sample user groups from the demonstration database.</p>	<a href="#">Defining User Groups</a>

Object	Description/Notes	Framework Administration Guide Reference
	<p>Also, you may be able to import user groups if your organization has already defined them using LDAP.</p> <p>Note, you won't be able to set up users at this point</p>	
Data Access Role	Used to provide rights to Access Groups (Owning Organization)	<a href="#">Defining Data Access Roles</a>
Access Group	Used to provide data segregation by 'Owning Organization' which is found on most objects in WAM.	<a href="#">Defining Access Groups</a>
User	<p>Defines a user's user groups, data access roles, portal preferences, default values, and To Do roles.</p> <p>SYSUSER is automatically set up, along with WAM_USER and ODM_USER. Note, you may be able to import your users if your organization has already defined them using LDAP.</p> <p>You must return to your user groups and define all of their users.</p>	<a href="#">Defining Users</a>

## Oracle Utilities Work and Asset Management Administrative Data Setup

These setup tasks are typically performed by an administrative user and are accessed from the **Admin** menu. These objects are defined at implementation and are only updated as needed when system configuration or foundation data changes. In general these objects establish the "Type" upon which the master data objects are based.

For more detailed information on the functionality related to the objects described, refer to the related section in this guide.

### General System Administrative Data

Sequence	Object Type	Prerequisites	Associated with / Referenced by
L1	Communication Type	None	Inbound Communication
L1	Statistic	None	Home Page
L1	Color	None	Cost Category Related <b>Master Configuration</b> must be configured.
L2	Inbound Communication	Communication Type	Completion Event Type Completion Event  e Note: this option is on the <b>Work Management</b> menu under <b>Menu</b> .
L5	Attachment	Attachment BO	Completion Event Assets Configuration Specifications Locations



Sequence	Object Type	Prerequisites	Associated with / Referenced by
			Service History Warranty Note: it is recommended that you use Document Types and Documents instead of attachments in Oracle Utilities Work and Asset Management.

### Approval Processing Administrative Data

One of the ways that the system simplifies and speeds up work is by routing records when they require approval. For example, purchase requests, work requests, employee leave, and other functions can be configured to require approvals.

Refer to [Understanding Approval Processing Configuration](#) for more specific configuration guidelines related to these objects.

Sequence	Object Type	Prerequisites	Associated with / Referenced by
L1	Approval To Do Type	OUIAF Setup	Approval Profile
L1	Approval To Do Role	OUIAF Setup	Approval Profile
L1	Approval User Group	OUIAF Setup	Approval Profile
L2	Approval Profile	To Do Type To Do Role User Group	System-Wide Refer to the <a href="#">Defining the Approval Hierarchy</a> section for examples of profiles to create. Related <b>Master Configuration</b> must also be configured.

### Financials Administrative Data

Financials management involves creating and managing entities such as expense codes, distribution codes, cost centers, cost categories, tax rate schedules, accounting calendars, , and so on.

Refer to [Understanding Financials Configuration](#) for more specific configuration guidelines related to these objects.

Sequence	Object Type	Prerequisites	Associated with / Referenced by
L1	Accounting Calendar	None	Related <b>Master Configuration</b> must also be configured.
L1	Cost Center	None	Expense Code Business Unit Tax Rate Schedule Employee Unavailability Type Related <b>Master Configuration</b> must be configured.
L1	Factor and Factor Value	None	Tax Rate Schedule

L1	Distribution Code	GL Account Construction Algorithm	Financial Transactions Related <b>Master Configuration</b> must also be configured.
L2	Cost Category	Color	Asset Type Expense Code
L2	Expense Code	Cost Category	Tax Rate Schedule Employee Unavailability Type Related <b>Master Configuration</b> must also be configured.
L3	Tax Rate Schedule	Factor Cost Center Expense Code	Purchase Requisition Purchase Order Blanket Contract Stock Item Related <b>Master Configuration</b> must also be configured.

### Resource Management Administrative Data

Resources encompass any object that contributes to the running of your organization. This includes hierarchies, employees, contacts, documents, standard notes and attachments, managing wage earning types and timesheets, and so on.

Refer to [Understanding Resource Management Configuration](#) for more specific configuration guidelines related to these objects.

Sequence	Object Type	Prerequisites	Associated with / Referenced by
L1	Contact Type	None	Locations
L1	Unit of Measurement — Resource	Resource Class (custom lookup)	Stock Item Equipment Miscellaneous Resource Craft
L1	Organization Type	None	Organization Asset Location Type Storeroom Type
L1	Service Class	None	Crew Shift Activity Type Crew
L1	Service Area	None	Asset Location Asset Employee Crew Shift
L1	Document Type	None	Document

L1	Employee Type	None	Employee
L1	Crew Shift Type	None	Crew Shift
L1	Crew Type	None	Crew
L2	Business Unit	Payroll Cost Center	Employee
L3	Employee Unavailability Type	Cost Center Expense Code	Employee Unavailability

### Inventory Management Administrative Data

Inventory is tracked, ordered, and received from vendors and allocated in the Inventory system. Materials are purchased and received, then as that inventory is allocated to work, the system passes this information back to the Work Management system so that maintenance managers and crews know that their parts are available.

Refer to [Understanding Inventory Configuration](#) for more specific configuration guidelines related to these objects.

Sequence	Object Type	Prerequisites	Associated with / Referenced by
L1	Storeroom Type	None	Storeroom
L2	Commodity Code	None	Stock Item

### Work Management Administrative Data

Work management is the process by which work orders are requested, approved and activated, work is performed and then work is finished and closed. While the work is being performed, materials, equipment assets and labor are tracked.

Refer to [Understanding Work Management Configuration](#) for more specific configuration guidelines related to these objects.

Sequence	Object Type	Prerequisites	Associated with / Referenced by
L1	Project Category	None	Project
L1	Time Periods	None	System-Wide for timekeeping
L1	Labor Earning Type	None	Timesheet
L1	Overtime Type	None	Timesheet
L2	Completion Event Type	Activity Type	Activity Type
L2	Activity Type	Completion Event Type Service Class Category (custom lookup)	Activity
L2	Maintenance Manager	User ID	Work Order
L2	Planner	User ID	Work Order Template Work Order Statistics Related <b>Master Configuration</b> can be configured for home page metrics.

L3	Failure Profile (Failure Cause, Modes, Repair, Component)	None	Preventive Maintenance Asset Type/Component Type
----	---	------	--

## Asset Management Administrative Data

The Oracle Utilities Work and Asset Management system provides functionality to track the purchase, repair and maintenance of assets and asset components using general work management processing, preventive maintenance techniques, purchasing processing and inventory tracking.

Refer to [Understanding Asset Management Configuration](#) for more specific configuration guidelines related to these objects.

Sequence	Object Type	Prerequisites	Associated with / Referenced by
L1	Unit of Measurement — Measurements (UOM)	None	Measurement Type Measurement Type Identifier (custom lookup)
L1	Question ( Condition Assessment )	None	Service History Type
L2	Configuration Type (ODM)	Configuration Type BOs and Lookups <ul style="list-style-type: none"> <li>Configuration Type BO</li> <li>Configuration Class</li> <li>Configuration Category</li> </ul>	Configurations Asset Configuration Class (custom lookup)
L2	Measurement Type	Measurement Unit of Measurement (UOM)	Measurement
L2	Service History Type	Condition Assessment Question	Service History Asset Type Condition Assessment Activity Type
L2	Asset Location Type	Parent Asset Location Organization Type (optional)	Asset Type
L2	Out of Service Location Type	Organization (optional) Parent Storeroom Type	Asset Type
L3	Condition Assessment Group	Service History Type	Condition Assessment Class
L4	Condition Assessment Class	Condition Assessment	Vehicle Assessment Building Assessment
L5	Asset Type/Component Type	Location Type Service History Type Assessment Class Failure Profile Asset Location Type	Asset

## Purchasing Administrative Data

Oracle Utilities Work and Asset Management uses Purchasing functions to plan, initiate, and process purchases and purchasing contracts. It also manages vendors, designated buyers, and stock items.

Refer to [Understanding Purchasing Configuration](#) for more specific configuration guidelines related to these objects.

Sequence	Object Type	Prerequisites	Associated with / Referenced by
L1	Capability Type	None	Vendor Capability (on the <b>Vendor</b> record)
L1	Payment Terms	None	Vendor
L1	Free on Board	None	Vendor
L2	Buyer	User ID	Purchase Commodities Related <b>Master Configuration</b> can be configured for home page metrics.
L3	Purchase Commodity	Buyer	Stock Item Related <b>Master Configuration</b> can be configured for home page metrics.

## Master Data Setup

These objects are typically defined during initial system setup and are then updated regularly as new updates are needed.

For more detailed information on the functionality related to the objects described, refer to the “User Guide” section in the online help or printed user’s guide.

**Note:** All basic Framework setup, including system and database setup and any modifications or extensions to base business objects, must have been completed before beginning these setup tasks.

To access the maintenance portals for the following objects, select **Menu** >[*submenu*]>[*object name*].

The chapters related to the functionality related to these objects in the remainder of this guide are organized in order the same order as indicated by this sequence.

## Resource Management Master Data

Resources encompass any object that contributes to the running of your organization. This includes hierarchies, employees, contacts, documents, standard notes and attachments, managing wage earning types and timesheets, and so on.

Refer to [Understanding Resource Management Configuration](#) for more specific configuration guidelines related to these objects.

Sequence	Resource Object	Prerequisites	Example/Notes
L2	Document	Document Type Category (custom lookup)	Document Attachments Standard Notes URLs
L2	Organization	Organization Type	

Sequence	Resource Object	Prerequisites	Example/Notes
		Parent Organization (optional)	
L2	Contacts	Contact Type	Business Contact Person Contact <ul style="list-style-type: none"><li>• Employees</li><li>• Vendors</li></ul>
L3	Crafts	Resource UOM — Non material class Expense Code	
L3	Equipment	Resource UOM — Non material class Cost Center Expense Code	
L3	Miscellaneous Resources	Resource UOM — Non material class Cost Center — Typically the type that does not allow user charges Expense Code	
L3	Employee	Employee Type User ID Craft Business Unit Service Area Course (custom lookup)	Employee Information Employee Billable Rate
L4	Crews	Crew Type Employee	
L5	Crew Shifts	Crew Crew Shift Type Schedule Plan Template Employee Service Class Service Area	Crew Shift Template Daily Schedule Schedule Plan Crew Shift with Activity Schedules Related <b>Master Configuration</b> must also be configured.
L5	Shift Plan Templates	Crew Schedule Plan Type Work Calendar	Shift Plan Templates Common Shift Plan Template Subscription Shift Plan Template

## Purchasing Master Data

Oracle Utilities Work and Asset Management uses Purchasing functions to plan, initiate, and process purchases and purchasing contracts. It also manages vendors, designated buyers, and stock items.

Refer to [Understanding Purchasing Configuration](#) for more specific configuration guidelines related to these objects.

Purchasing Object	Prerequisites	Examples/Notes
L1	Manufacturer	None
L1	Vendor	None
L2	Vendor Locations	Vendor
L3	Blanket Contract Access List	User ID
L4	Blanket Contracts/w Line Items	Stock Items Blanket Contract Access List Vendor Payment Terms Cost Center

### Inventory Management Master Data

Inventory is tracked, ordered, and received from vendors and allocated in the Inventory system. Materials are purchased and received, then as that inventory is allocated to work, the system passes this information back to the Work Management system so that maintenance managers and crews know that their parts are available.

Refer to [Understanding Inventory Configuration](#) for more specific configuration guidelines related to these objects.

	Inventory Object	Prerequisites	Examples/Notes
L2	Storeroom	Storeroom Type Contact (optional) Organization (optional)	Storeroom Storeroom with Inventory Management
L3	Stock Items	Resource UOM Usage Expense Code Inventory Expense Code Purchase Commodity Commodity Code Vendor Location Manufacturer Stock Item Category (custom lookup)	
L3	Stock Item Details	Stock Item Storeroom Cost Center Buyer Vendor Bin Information	Stock Item Detail - Inventory Stock Item Detail - Direct Purchase Stock Item Detail - Specific Direct Purchase Stock Item Detail - Specific Lot Stock Item Detail - Expensed At Receipt
L4	Material Request Template	Storeroom Location Stock Item Detail Expense Code	
L4	Reorder Review	Stock Item Detail	Related <b>Master Configuration</b> must also be configured.

	Inventory Object	Prerequisites	Examples/Notes
L4	Inventory Adjustment	Reason Code (custom lookup)	

## Asset Management Master Data

The Oracle Utilities Work and Asset Management system provides functionality to track the purchase, repair and maintenance of assets and asset components using general work management processing, preventive maintenance techniques, purchasing processing and inventory tracking.

Refer to [Understanding Asset Management Configuration](#) for more specific configuration guidelines related to these objects.

	Asset Object	Prerequisites	Examples/Notes
L2	Assets	Asset Type	Tracked General Assets Tracked IT Asset / In Out Service IT Asset In Out of Service General Asset In Out of Service Infrastructure Asset Communication Component Communication Relay
L2	Components (Asset Components)	Asset Type	In/Out of Service General Component In Out Service IT Component Operational Device Tracked Component Tracked General Component Tracked IT Component
L2	Vehicles (Fleet Assets)	Asset Type	In Out Service Fleet Asset
L2	Operational Devices (Assets)	Asset Type	Operational Device Tracked Asset Operational Device In/Out of Service Asset Smart Meter Manual Meter
L2	Asset Location	Asset Location Type Site Location (custom lookup)	General Asset Location Linear Asset Location Plant Asset Location Service Point Operational Device Asset Location
L2	Out of Service Location	Out of Service Location Type	Warehouse Repair Shop
L2	Configuration	Configuration Type	Single Asset Configuration Asset/Component Configuration Asset to Asset Configuration



	Asset Object	Prerequisites	Examples/Notes
			Asset/Component to Asset Configuration Asset to Asset/Component Configuration Asset/Component to Asset/Component Configuration
L2	Operational Reading	Measurement Type Measurement Numeric Gauge Reason (custom lookup)	
L3	Bills of Materials	Stock Item	General Stock Item Independent Part BOM
L3	Warranty Term	Vendor	Calendar Warranty Term Runtime Warranty Term
L3	Specification	Specification BO Asset Type Manufacturer BOM ID Maintenance Schedule (as applicable)	
L3	Fleet Specification	Vehicle Detail (custom lookups)	
L3	Service History	Activity Failure Profile Deficiency (custom lookup) General Weighted (custom lookup) Percent of New (custom lookup) Asset Disposition (custom lookup) Downtime Reason (custom lookup)	General Preventive Maintenance Inspection Operational Device
L4	Warranty	Warranty Term	Calendar Warranty Runtime Warranty
L4	Configuration Reports	Configuration Report BOs Specification Reporting Mode (custom lookup)	Single Asset Asset/Component Asset to Asset Asset/Component to Asset Asset to Asset/Component Asset/Component to Asset/Component

### Work Management Master Data

Work management is the process by which work orders are requested, approved and activated, work is performed and then work is finished and closed. While the work is being performed, materials, equipment assets and labor are tracked.

Refer to [Understanding Work Management Configuration](#) for more specific configuration guidelines related to these objects.

Sequence	Work Management Object	Prerequisites	Examples/Notes
L1	Preventive Maintenance Calendar	None	
L2	Preventive Maintenance Schedule	Maintenance Business Object Asset Type	Maintenance Schedule Reliability Centered Maintenance Schedule
L3	Preventive Maintenance Trigger	Maintenance Trigger Business Object Maintenance Schedule Template Work Order Activity Type Service Class Approval Profile Outage Type (custom lookup) Backlog Group (custom lookup)	Calendar Anniversary Calendar Interval Runtime Anniversary Runtime Interval RCM Calendar Anniversary RCM Calendar Interval RCM Runtime Anniversary RCM Runtime Interval Asset Location Maintenance Trigger
L3	Work Order Template	Planner Approval Profile Work Class (custom lookup) Work Category (custom lookup) Work Priority (custom lookup)	General Template Work Order Asset Location Specific Template Work Order Note: Refer to special instructions for the work priority field.
L4	Template Activity	Resources	General Template Activity Asset Location Specific Template Activity
L5	Activity	Activity Type Completion Event Type (as applicable) Storeroom Asset Project Work Order Service Class Cost Center Phase (custom lookup) Crew (as applicable) Backlog Group (custom lookup)	
L5	Work Order	Project Cost Center Crew Planner	Note: Refer to special instructions for the work priority and criticality fields.

Sequence	Work Management Object	Prerequisites	Examples/Notes
		Criticality (custom lookup)	Related <b>Master Configuration</b> must also be configured.
		Work Class (custom lookup)	
		Work Category (custom lookup)	
		Work Priority (custom lookup)	

## Batch Control Configuration

Batch controls are used to process and organize information throughout the system. For example, reorder review relies upon batch processing to complete the reorder processing and generate the appropriate purchasing documents. Likewise, shift plan templates use batch patch processing to generate shifts and make updates to shift plans. This section describes the base package batch controls provided with Oracle Utilities Work and Asset Management.

Use the **Batch Control** portal and **Application Viewer** for more details about these batch controls.

To verify the status of a batch process you can check the Batch Job Submission portal and search for the related batch control. For example, if you have scheduled activities to be created by batch processing from a work order, you would search for **W1-WO**.

- Navigate to **Main Menu > Batch > Batch Job Submission**.
- Search for the appropriate batch control to find out the status of a job.

Refer to the [Background Processes](#) section for more information about tracking batch processes, the Batch Run Tree, and other functionality related to batch processing.

## About Monitors

The main batch controls that might require configuration are monitors. Monitors determine when an event has taken place within the system and trigger another event in response. These may or may not require modification to the interval, or frequency of doing a status check on events and launching triggers.

As a general rule, all monitors are executed the first time an entity enters a given state. After that event, the system relies upon the batch process configuration to determine when the process is called again.

### Non-Timed Monitors

Monitor type batch controls can be configured with no defined timer interval so that they can be run whenever required. Typically, organizations that use a third-party software to manage monitoring and batch triggers would configure monitors in this way.

### Timed Monitors

Monitor type batch controls that do not have third-party software to trigger the processes can use the simple processing in base package batch controls to launch the batches at whatever time interval is set as the timer interval.

# Oracle Utilities Work and Asset Management Batch Controls

It is important to properly manage the order and frequency with which batch processes are run. Below is a list of batch jobs which, if scheduled, must be in the order listed. These are recommended settings (not requirements) based on which aspects of the system you are using and your business practices.

To find the details for batch controls, reference the **Batch Control** record and then look at the maintenanceObject Parameter.

Refer to [Defining Batch Controls](#) for information on controlling the order in which the processes run, scheduling processes, and other details.

## Scheduled Monitor Processes

Batch controls are listed here in the general sequence that they should be run:

Batch Control	Description	Suggested Freq.
W1-FTM Financial Transaction Periodic Monitor Process	<p>Invokes monitoring rules associated with the current state of financial transactions. All monitoring rules throughout the financial transaction's business object's inheritance chain are considered.</p> <p>By default, the process periodically monitors financial transactions whose current state is not associated with a batch code.</p> <p>Batch parameters govern whether the processing is further restricted by batch code, business object and status.</p> <p>This processing also transitions objects from Created status to Populate GL Details to Frozen, as applicable. If the system encounters errors, the object is transitioned to Error. For example the GL number might not be able to be generated because of an incorrect cost center/expense code combination. There is a monitor on the error state to try again.</p> <p>The error state creates a To Do. Users can always manually try to freeze the FT. However if the issue that caused the error has been fixed, the monitor will take care of the FT and To Do before users take actions.</p> <p>The W1-GLDL process, GL Download Extract, extracts and downloads the general ledger details.</p>	Run hourly.
W1-APVRQ Approval Request Periodic Monitor Process	<p>Manages transitioning of approval requests and routing them to the appropriate approvers. For example, if the current approver times out due to not responding in the allotted amount of time, this processing</p>	Run several times an hour.

Batch Control	Description	Suggested Freq.
	<p>moves the approval request to the next approver.</p> <p>The W1-APVRQ monitor process, Approval Request Periodic Monitor Process,</p>	
W1-ACTRR Activity Resource Requirement	Users can indicate a future 'Reservation' date on Material Requirements and the system will try to reserve the material when that date becomes current.	Run several times a day.
W1-ACT Activity MO Periodic Monitor Process	<p>Updates the activity Held For Parts flag based on availability in inventory. The monitor can be scheduled to re-evaluate the Held for Parts flag on all active activities against inventory quantity changes causes by stock re-order, material issue or return".</p> <p>The <b>Activation Date/Time</b> on an activity defines a future activation date. The system will activate the activity once it is both approved and the 'Activation Date' becomes current. This monitor uses this date to auto-activate the activity.</p>	Run hourly.
W1-WO Work Order MO Periodic Monitor Process	<p>Causes the system to close Completed work orders after the indicated number of days.</p> <p>This is relevant for auto close and service schedules.</p>	Run hourly.
W1-TMST Timesheet MO Periodic Monitor Process	Posts approved timesheets to create financial transactions and to the general ledger.	Run daily.
W1-ODCTR ODC MO Periodic Monitor Process	Posts approved direct charges to create financial transactions and to the general ledger.	Run daily.
W1-COSTM Cost Center Periodic Monitor Process	<p>Accumulates monthly and year to date statistics per cost center.</p> <p>This batch process invokes monitoring rules associated with the current state of cost centers. All monitoring rules throughout the cost center's business object's inheritance chain are considered. By default, the process periodically monitors cost centers whose current state is not associated with a batch code. Batch parameters govern whether the processing is further restricted by batch code, business object and status.</p>	Run daily.
W1-ASTYP Asset Type Monitor Process	Accumulates asset type average cost statistics.	Run daily.
W1-CRSHF Crew Shift Periodic Monitor Process	Generates shifts based on templates.	Run several times an hour.
W1-ASSET Asset Monitor	Monitors preventive maintenance triggers to create the related work orders.	Run daily.

Batch Control	Description	Suggested Freq.
	This monitor also creates PM Forecast records for the Asset. The planning horizon (number of days into the future) is controlled by a parameter on the W1-ASTMNTMON algorithm.	
W1-BCHDR Blanket Contract Header Monitor Process	Transitions blanket contracts that are awaiting activation to Active when future blanket contracts become current.	Run daily.
W1-BSRR Batch Stock Reorder Monitor Process	Creates the purchasing documents based on reorder review records.	Run daily.
W1-GLDL GL Download Extract	Extracts and downloads the general ledger details by creating a flat file.  The GL batch control code is defined in the <b>Financial Master Configuration</b> .	
W1-PRJ Project MO Periodic Monitor Process		
W1-STAT Statistic Monitor Process	Triggers gathering of various statistics for graphs.	Run daily.
W1-SFWKT Shft Wkly Templt Periodic Monitor Process	Creates new shifts based on templates.	Run several times an hour.
W1-ASCDU Asset Current Disposition Update		
W1-SID SID MO Periodic Monitor Process	Causes the stock item detail to monitor itself to check it's reorder levels and see if it is time to create a reorder review.	
W1-CVNR Create Vendor from Characteristic		

## General Monitor Processes

Monitor controls are available in the base package so that if your implementation does add monitor algorithms to a business object for the related maintenance object, the batch control already exists to run. As delivered, the maintenance object instances current state does not reference a monitor process.