

Oracle Utilities Operational Device Management User Guides

Oracle Utilities Operational Device Management User Guides comprise the following:

- Oracle Utilities Operational Device Management Business User Guide
- Oracle Utilities Operational Device Management Administrative User Guide

Release 2.4.0.0.0

F52876-01

**Oracle Utilities
Work and Asset Management and
Operational Device Management**

Business User Guide

Release 2.4.0.0.0

F52875-01

April 2022

Also used with:

- Oracle Utilities Operational Device Cloud Service 22A

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

Overview.....	9
About Oracle Utilities Work and Asset Management.....	9
About Home Page.....	10
Supported Portals on iOS.....	10
User Interface Standard Features.....	12
Page Components.....	12
The Application Toolbar.....	13
Home Icon.....	13
Menu.....	14
Admin Menu.....	15
Back and Forward Arrows.....	16
History Icon.....	16
Search.....	17
Help Menu.....	19
User Menu.....	19
Page Title Area.....	20
Page Actions Toolbar.....	20
Bookmark Button.....	21
Next/Previous Item Buttons.....	21
Duplicate Button.....	22
Delete Button.....	22
Clear Button.....	23
Save Button.....	23
Refresh Button.....	24
Object Display Area.....	24
Multiple Tabs.....	24
Text Box.....	25
Date and Time Fields.....	27
Search Button.....	28
Context Menu Button.....	29
Go To Button.....	29
Broadcast Button.....	29
Embedded Help Icon.....	29
Work Lists.....	29
Scroll Bars with Dynamic Height.....	30
Sorting Functionality.....	31
Get More Functionality.....	31
Adding and Removing Rows Using Arrows.....	31
Reorganizing Items Using the Up and Down Arrows.....	32
Trees.....	32
Required Field Indicator.....	34
Accordion Windows.....	34
Row Summary and Chunk Management.....	35
Dashboard Area.....	35
Script Area.....	36
Portal Pages.....	36
Portals Are Made Up Of Zones.....	36
Users Must Be Granted Security Access To Each Zone.....	37
Zones May Appear Collapsed When A Page Opens.....	37
Some Portals Allow User Customization.....	38
Dashboard Portal.....	38
Bookmarks Zone.....	39
Tags Zone.....	39
Collect Entity Zone.....	39
Current To Do Zone.....	39
Favorite Links Zone.....	41
Favorite Scripts Zone.....	41
Favorite Searches Zone.....	41

Switch Language Zone.....	41
To Do Summary Zone.....	41
Work List Zone.....	42
Zone Types and Characteristics.....	42
Common Characteristics of Zones.....	42
Zone Visibility.....	42
Control Zone Headers.....	42
Explorer Zones.....	42
Search Criteria / Filter Area.....	43
Description Bar.....	43
Results Area.....	43
Title Bar.....	44
Configurable Columns.....	48
Multi-Query Zones.....	48
Map Zones.....	48
Open Map Maintenance Icon.....	49
Tabbed Display Zones.....	49
Timeline Zone.....	49
Common Base Portals and Zones.....	50
All-In-One Portal.....	50
List Zone.....	51
Separate Query and Maintenance Portals.....	51
Query Portal Search Zone.....	51
Maintenance Portal.....	51
Process Flow.....	52
Fixed Pages.....	52
Maintenance Pages.....	53
The Unique Identifier Area.....	53
Confirmation & Error Messages.....	53
Warning If Unsaved Changes Exist.....	53
Query Pages.....	54
The Search Area.....	54
User Preferences.....	54
User - Main.....	55
User - To Do Roles.....	56
User - Access Security.....	56
User - Bookmarks.....	57
User - Portal Preferences.....	57
User - Favorite Links.....	58
User - Favorite Scripts.....	59
User - Characteristics.....	59
User - Miscellaneous.....	59
Initiating Scripts.....	60
Shortcut Key Summary.....	61
WAM Reports.....	65
About WAM Reports.....	65
Running Oracle Utilities Work and Asset Management Reports.....	65
Search Features.....	67
Using Unified Search.....	67
Assets.....	69
About Asset Locations.....	69
Types of Locations.....	69
Service Area.....	70
Location Hierarchy.....	70
Disposition Locations.....	70
Belongs to an Organization.....	70
Contacts.....	70
About Assets.....	71
Components.....	71
Types of Assets.....	71
Asset Hierarchy.....	72
Asset Disposition.....	72
Extending Asset Statuses.....	72

Replicating Assets.....	73
Asset Service History.....	73
Warranty Terms and Warranties.....	73
About Operational Devices.....	73
Measurements.....	74
Device Configuration.....	74
About Specifications.....	75
Firmware.....	76
About Functional Processes.....	76
Maintaining Asset Locations.....	76
Maintaining Contacts.....	77
Maintaining Assets.....	78
Maintaining Asset Replications.....	79
Maintaining Operational Readings.....	79
Maintaining Specifications.....	79
Maintaining Functional Processes.....	80
Maintaining Warranties.....	81
Maintaining Warranty Terms.....	81
Mass Asset Disposition Update.....	81
Operational Reading Alert Review.....	82
360 Degree Search.....	82
GIS Map Viewer.....	82
Operational Device Management.....	85
About Operational Device Management.....	85
About 360 Degree Search.....	86
About 360 Degree View.....	86
About ODM Configurations.....	86
About ODM Configuration Reports.....	87
About ODM Firmware.....	88
About Activity Generators.....	89
Maintaining ODM Configuration Reports.....	90
Maintaining Activity Generators.....	90
Inventory.....	92
About Inventory.....	92
Stock Items.....	92
Stock Item Details.....	93
Physical Inventory.....	94
Inventory Adjustments.....	94
Stock Transfers.....	95
Receiving Inventory.....	95
Returns.....	95
About Material Disposition.....	96
Material Requests.....	96
Material Issue.....	97
Material Returns.....	98
Material Request Templates.....	98
Bill of Materials.....	98
About Shipping Memos.....	99
Maintaining Stock Items.....	99
Maintaining Stock Item Details.....	100
Maintaining Stock Transfers.....	100
Stock Transaction Log Portal.....	101
Maintaining Physical Inventories.....	101
Maintaining Receipts.....	101
Maintaining Shipping Memos.....	102
Maintaining Returns.....	102
Maintaining Material Issue.....	103
Maintaining Material Requests.....	103
Maintaining Material Request Templates.....	103
Maintaining Material Returns.....	103
Maintaining Material Dispositions.....	104
Maintaining Bill of Materials.....	104
Maintaining BOM Parts.....	104
Maintaining Reorder Reviews.....	105

Reorder Review Mass Approval.....	105
Purchasing.....	106
About Purchasing.....	106
Manufacturers.....	106
Vendors.....	106
Vendor Locations.....	106
Purchase Orders.....	107
Purchase Requisitions.....	107
Blanket Contracts.....	108
Invoices.....	108
Expedite Purchase Orders.....	110
Maintaining Purchase Orders.....	110
Maintaining Purchase Order Lines.....	110
Maintaining Purchase Requisitions.....	111
Maintaining Purchase Requisition Lines.....	112
Maintaining Expedite Purchase Orders.....	112
Maintaining Blanket Contracts.....	112
Maintaining Blanket Contract Lines.....	113
Maintaining Invoices.....	113
Maintaining Invoice Groups.....	113
Maintaining Manufacturers.....	114
Maintaining Vendors.....	114
Maintaining Vendor Locations.....	115
Service History.....	116
About Service History.....	116
Downtime.....	116
Failures.....	116
Inspections.....	117
Condition Assessments.....	117
Questionnaires.....	118
Maintaining Service History.....	118
Work Management.....	119
About Work Management.....	119
About Field Activities.....	120
Planning Field Activities.....	120
Approvals for Field Activities.....	120
Scheduling Field Activities.....	120
Field Activity Cost.....	121
Field Activity Workflow.....	121
About Work Orders.....	121
Work Order Includes One or More Activities.....	123
Work Order Workflow.....	123
Template Work Orders.....	124
Projects.....	124
About Work Activities.....	125
Planning Work Activities.....	125
Resource Requirements.....	125
Approving Work Activities.....	126
Scheduling Work Activities.....	126
Work Activity Cost.....	126
Checklists.....	128
Work Activity Workflow.....	128
Work Activity Templates.....	129
About Work Requests.....	130
About Work Planning.....	130
About Permits.....	131
About Communication Logs.....	131
About Service Calls.....	132
Maintaining Field Activities.....	132
Maintaining Work Orders.....	133
Maintaining Template Work Orders.....	133
Maintaining Work Activities.....	134
Maintaining Work Requests.....	135

Maintaining Work Location.....	136
Maintaining Projects.....	136
Maintaining Permits.....	137
Maintaining Service Calls.....	138
Work Planning Dashboard.....	138
Maintaining Activity Generators.....	139
Preventive Maintenance.....	140
About Preventive Maintenance.....	140
Maintenance Schedules.....	141
Maintenance Triggers.....	141
Maintaining Maintenance Events.....	142
Maintaining Maintenance Plans.....	143
Maintaining Maintenance Plan Triggers.....	143
Maintaining Maintenance Specifications.....	143
Maintaining Preventive Maintenance Schedules.....	144
Maintaining Preventive Maintenance Triggers.....	144
Construction Work.....	145
About Construction Work.....	145
Planning Construction Work.....	145
Work Designs.....	146
Construction Work Costs.....	146
Approving Construction Work.....	146
Construction Work Orders.....	146
Construction Work Activities.....	147
Construction Work Locations.....	147
Scheduling Construction Work.....	147
Construction Work Activity Reconciliation.....	147
Communication Logs.....	147
Maintaining Work Designs.....	148
Maintaining Construction Work Orders.....	148
Maintaining Construction Work Activities.....	149
Maintaining Activity Reconciliations.....	150
Scheduling.....	151
About Scheduling.....	151
The Scheduling Process.....	152
Daily Versus Weekly Schedules.....	152
Break In.....	153
About Crew Shifts.....	153
Creating Shifts.....	153
Service Area.....	153
Crew Shift Workflow.....	153
Controlling the Schedule.....	154
Employee Unavailability.....	154
Maintaining Crew Shifts.....	155
Maintaining Employee Unavailability.....	155
Scheduling Portal.....	155
Approvals.....	157
About Approvals.....	157
Approval Profiles.....	157
Approval Routing.....	158
Outstanding Approvals at a Glance.....	159
Approval Dashboard.....	159
Maintaining Approval Requests.....	159
About Change Requests.....	159
Change Request Approval Profiles.....	160
Change Request Approval Routing.....	160
Maintaining Change Requests.....	161
Resource Usage.....	162
About Timesheets.....	162
Timesheet Quick Entry.....	163
About Resource Charges.....	163
Equipment Charges.....	163
Other Direct Charges.....	164

Maintaining Timesheets.....	164
Maintaining Timesheet Quick Entry.....	164
Maintaining Other Direct Charges.....	164
Financials.....	166
About Financial Transactions.....	166
About Cost Adjustments.....	166
Maintaining Financial Transactions.....	167
Maintaining Cost Adjustments.....	167
To Do Processing.....	168
Understanding To Do Processing.....	168
To Do Role.....	168
To Do Priority.....	168
Work Distribution.....	169
A User's Current To Do.....	170
To Do Characteristics.....	170
Related To Do Entries.....	171
Complete A To Do.....	171
Log Entry Events.....	171
Managing To Do Entries.....	172
To Do Management.....	173
To Do Supervisor Functions.....	175
Supervisor To Do Summary.....	175
Supervisor User Summary.....	176
Supervisor To Do Assignment.....	176
To Do List Functions.....	177
To Do Summary.....	177
To Do List.....	178
To Do Entry.....	180
Adding a Manual To Do Entry.....	181
To Do Search.....	181
To Do Dashboard.....	183
Reports.....	185
On-line Report Submission.....	185
Report Submission - Main.....	185
Viewing Historic Reports.....	186
Report History - Main.....	186

Chapter 1

Overview

About Oracle Utilities Work and Asset Management

Oracle Utilities Work and Asset Management provides functionality to handle large volumes of assets and to manage the receipt, installation, maintenance, tracking, and removal of those assets. The system also manages approval processing, tracks purchasing transactions, manages inventory and resources, and tracks costs, accounting, and financial transactions.

The system manages the following main functional areas:

- **Home Page:** The home page provides a comprehensive dashboard where you can access information you need to review and complete your work. For example, the **Main** tab can be configured to display your key work areas; crews or buyers that you monitor, approval summary, your bookmarks or saved searches, and so on. Additional tabs provide easy access for approvers, planners, executives, and others to information and to review organizational functions and complete work.
- **Approvals:** One of the ways that the system simplifies and speeds up work is by routing records when they require approval. Approval requests use approval profiles to determine the properties and requirements for the approval. Approvers can use the **Approval Dashboard** to manage the approvals that require their attention.
- **Assets:** Asset management provides functionality to track the purchase, repair, and maintenance of assets and asset components using general work management processing, preventive maintenance techniques, purchase processing, and inventory tracking. The system also allows organizations to record asset measurements and perform asset inspections in addition to the standard maintenance that is performed on assets.
- **Financial Transactions:** The system tracks transactions that incur costs, including purchasing and procurement costs, inventory and stocking costs, and cost associated with the labor, equipment, and materials that are incurred when completing work.
- **Inventory:** Inventory is tracked, ordered, and received from vendors and allocated in the **Inventory** system. As 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.
- **Preventive Maintenance:** Maintenance type work on assets is a proactive way to care for and maintain proper functioning of an asset. This type of work can include inspections, repairs, replacements and so on. Maintenance schedules and triggers identify the type of asset needing maintenance and triggers tell the system when the asset is due for maintenance.


- **Purchasing:** Purchasing functions can be used to plan, initiate, and process purchases and purchasing contracts from requisition to invoicing. The system also manages vendors, designated buyers, and stock items.
- **Resource Management:** Resources encompass any object that contributes to the running of your organization. This includes crews and crew shifts, employees, equipment, contacts, documents, standard notes and attachments, and so on.
- **Work Management:** Work management is the process by which work orders are requested, approved and activated, work is performed and then finished and closed. While the work is being performed, materials, equipment assets, and labor are tracked, as applicable, to your business processes. The system also allows organizations to set up preventive and predictive maintenance cycles.
- **Construction Work Management:** Construction work management refers to the various functions related to construction type work, such as new facilities, requests for new services or additions, and designing and estimating line extensions. Construction work uses compatible units to identify the resources necessary to install or remove assets associated with a work site.

About Home Page

The **Home Page** portal provides a comprehensive dashboard where you can access much of the information you need to review and complete your work.

The **Home Page** can serve as a personal web page where you configure system elements that you use most frequently.

Refer to the [Home Page Configuration](#) section in the **Administrative User Guide** for information on configuring home page settings.

In general, click on any link, pie chart, or graph to open details about that item. The **Go To**  icon at the front of some items opens the corresponding tab with the related information populated. For example, if you click the symbol in front of a storeroom in the **My Storerooms** section on the **Main** tab, the system opens the **Inventory** tab with that **Storeroom ID** populated. Click **Refresh** to update and display the graphs.

The base package home page can include the following tabs (refer to the embedded help for zone details):

- **Main:** The **Main** tab displays general system information and any custom zones provided by your implementation (such as company news). Click the Add link in any zone to add the related information. The My Buyer, My Storerooms, and My Crews zones do not automatically include the records. For example, if the logged in user is a buyer, they must still add their own buyer to the zone, if they want to include their own buyer in the zone.
- **Approvals:** The **Approvals** tab provides the same information and functionality as the [Approval Dashboard](#).
- **Work:** The **Work** tab can display statistics about work requests and work orders. Some of the work zones might include a link to the **Statistics** portal. Authorized users can edit some of the details displayed in the zones using this portal.
- **Inventory:** The **Inventory** tab provides information about stock items by storeroom. You must enter a storeroom and click **Refresh** to display the zones.
- **Purchasing:** The **Purchasing** tab can display purchasing statistics by buyer. You must select a buyer and click **Refresh** to display the zones.

Supported Portals on iOS

The following portals are supported when accessing the WAM application via a browser on Apple iOS devices in this release:

Crew Work-related Portals:

- Work Requests

- Service Call
- Work Order
- Work Activity
- Permit
- Timesheet
- Other Direct Charges
- Construction Work Order
- Construction work Activity

Inventory-related Portals:

- Material Request
- Material Return
- Receiving Receipt
- Receiving Return
- Physical Inventory
- Reorder Review
- Reorder Mass Approval
- Stock Item
- Stock Item Detail
- Stock Transfer

Chapter 2

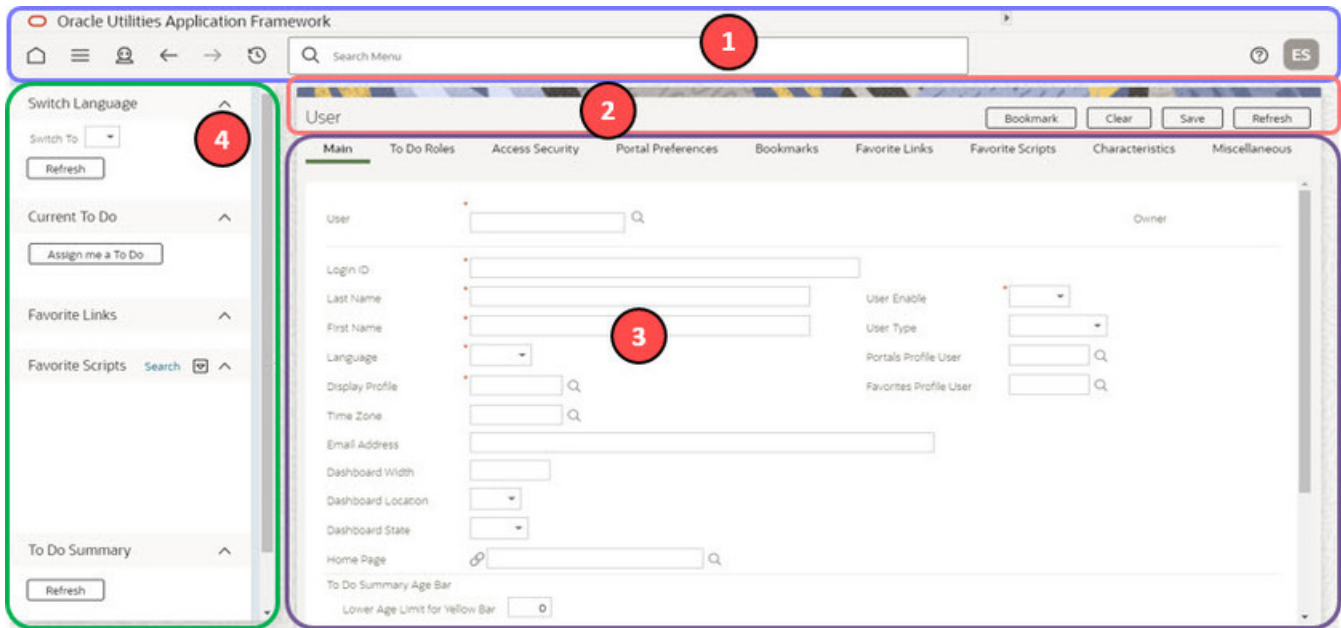
User Interface Standard Features

This section describes basic system concepts, features, and standards of the user interface.

Page Components

Oracle Utilities Application Framework screens are comprised of the following main areas:

1. The Application Toolbar
2. The Page Title Area
3. The Object Display Area
4. The Dashboard Area



Keyboard Shortcuts	Description
Alt+[Moves the cursor focus to the next main page component based on the standard browser keyboard navigation "order".
Shift+Alt+[Moves the cursor focus to the previous main page component based on the standard browser keyboard navigation "order".

NOTE: The look and feel of the application may be modified after the product is installed. See [Custom Look and Feel Options](#) for customization information, including how to change colors, fonts, and other system features. The information provided in this document represents features and functionality available only in the delivered product.


The next topics provide more information about each component and the Script area, which appears in the Object Display area when applicable.

The Application Toolbar

This section describes the features available on the application toolbar.

Home Icon




Click the **Home** icon, , on the Application Toolbar to open your home page. Your home page is defined in [User Preferences](#).

Keyboard Shortcut	Alternate Shortcut
Alt+O	Alt+Shift+O

NOTE: Refer to [Shortcut Key Summary](#) for information about the alternate shortcut.

Menu

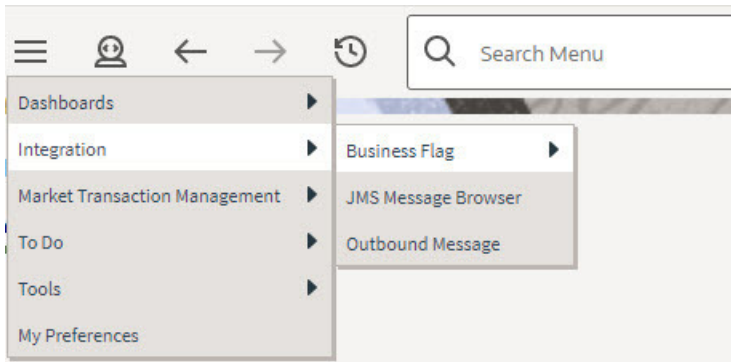


The Menu icon, , is available in the application toolbar to help you navigate to the different pages of the system. You only see the menu items that you have security access for.

Keyboard Shortcut

Ctrl+Alt+M

The menu list is organized by functional area. Clicking the Menu button displays each functional area. Clicking a functional area, in turn, displays a submenu that contains pages within that area.



The pages within each functional area *typically* have two options: Search and Add. If the menu item does not have an Add or Search option, select the menu item itself.

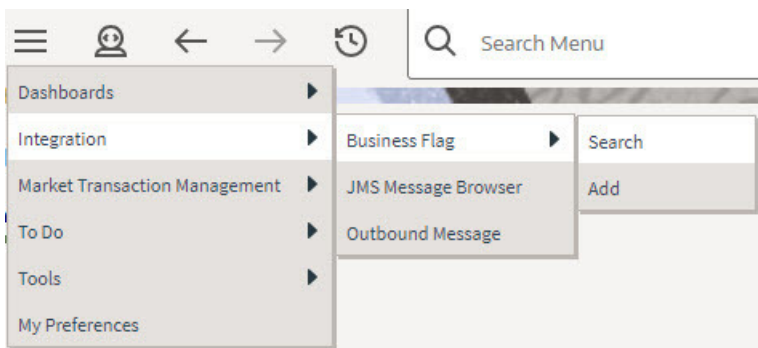
How the Search and Add option behaves depends upon whether the maintenance page is fixed or portal-based.

Fixed:

- **Search:** Displays a pop-up search window for the user to enter the search criteria and select the entity to display. Once the entity is selected, the user is taken to the maintenance map with the data populated in the fields.
- **Add:** Displays the maintenance page with empty fields so that the user can complete the information and save the entity.

Portal-Based:


- **Search:** Displays a search portal where the user can enter the search criteria and select the desired entity. Once the entity is selected, the user is typically taken to a maintenance portal or the information is broadcast to other zones within the same portal.
- **Add:** Either navigates to a page where the user can select the entity type or business object, or directly to the input map where the user can enter and save the entity.



Users may also opt to use [Menu Item Search](#) to find a page.

Admin Menu



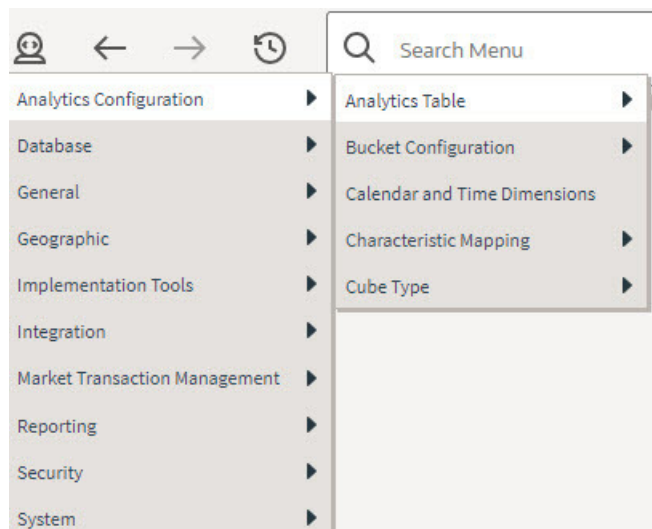
The Admin menu icon, , is available in the application toolbar to help you navigate to the different pages of the system. You only see the menu items that you have security access for. If you don't have security access to any of the entries in this menu, then the whole menu is suppressed.

Keyboard Shortcut

Ctrl+Alt+A

Depending on how your implementation has configured the Admin menu list, it may be organized either by functional area or alphabetically.

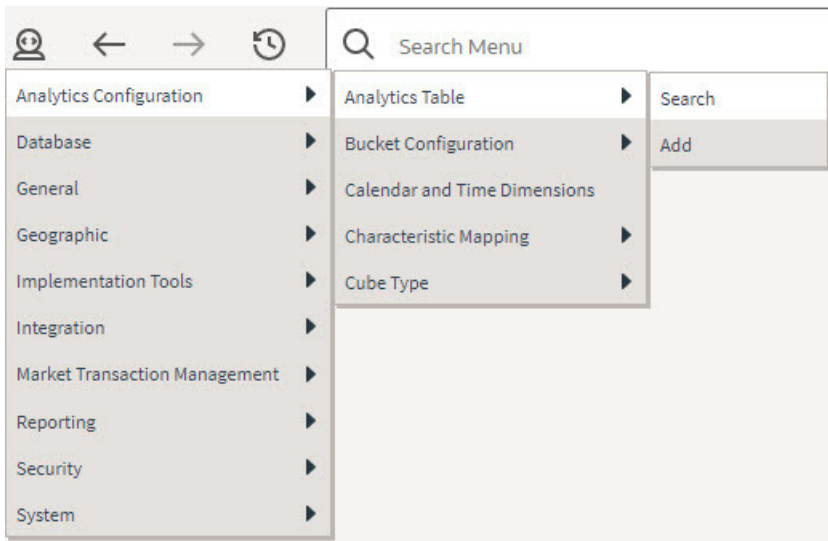
Clicking the Admin icon displays each functional area or alphabetical list. Clicking one of the options, in turn, displays a submenu that contains pages within that area. The following is an example of the Admin menu organized functionally.



NOTE: Menu navigation paths referenced in the administrative user guide provide the functional grouping. If your implementation uses the alphabetic grouping, the page can be found in the alphabetic letter that is the first letter of the page name.

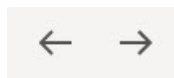
NOTE: Refer to [Installation Options](#) for information about the admin menu configuration options.

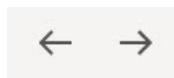
The menu lines displayed for each menu entry *typically* have two options: Search and Add. If the menu item does not have an Add or Search option, select the menu item itself. See [Menu](#) for more information about these options.



Users may also opt to use the [Menu Item Search](#) to find a page.

Back and Forward Arrows



The back and forward arrows, , in the application toolbar allow you to navigate between the most recently viewed pages.

Back

Click the **Back** arrow to open the previously visited page.

Keyboard Shortcut

Alt+B

Forward

Click the **Forward** arrow to return to the page that was most recently dismissed.

Keyboard Shortcut


Alt+G

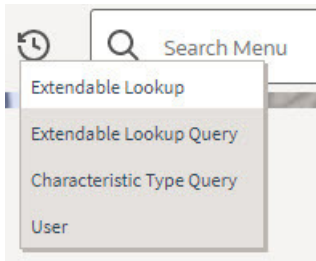
The **back** and **forward** arrows are enabled in the toolbar only when page use warrants their appearance.

NOTE: Under certain circumstances, such as revisiting a page that was left unsaved after being loaded with default data, a page may not appear as it did when you left it.

History Icon

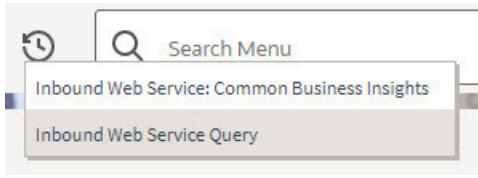


Click the **History** icon, , to display a list of previously visited pages. When you click on an entry in the list to return to an earlier page, all items above the selected page are removed from the list.



The **back** and **forward** arrows, as well as the **History** list, appear in the application toolbar only when page use warrants their appearance.

Note that if the page has been configured to display an information string in the [page title area](#), that information will also be visible in the History dropdown.



Keyboard Shortcut

Ctrl+Alt+H

Search

The toolbar Search field and search icon provide the ability to search for specific menu items or, depending on your product, certain business entities.

Menu Item Search

This search allows a user to enter the description of a menu item entry to navigate directly to the corresponding page or BPA script rather than using the menus to navigate to the desired page or script. Enter your search text starting with a slash "/" to indicate that the search is confined to menu items.

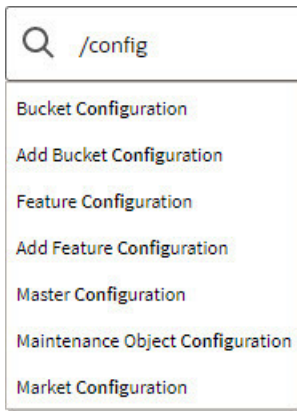
Keyboard shortcut:

Keyboard Shortcut

Alt+F1



Typing text in the field shows menu items whose descriptions include the typed text.



Note that only menu items within the [Menu](#) and [Admin Menu](#) that the user has security access for are included in the results. The text included in the search is taken from the menu item description. As described in [Menu](#), menu lines often have two sub item: Search and Add. In most instances, the description of the "Search" menu item is simply the menu entry text, for example **To Do Entry**. The description of the "Add" menu entry includes the word Add in front of the menu entry text, for example **Add To Do Entry**.

As mentioned in [Menu Item Suppression](#), menu items will not appear if it is associated with a module that is suppressed.

NOTE: When operating in debug mode, the description of the parent menu is displayed in parentheses after the menu item description.

Unified Search

If your product has implemented unified search, you may use the toolbar search box to search for business entities. The unified search feature is a simplified version of your product's main search that allows you to lookup up records using free form text without leaving your current page. For more complex queries, use the link provided as part of the search box to navigate to your product's advanced search portal.



The type of business entities you may query and the filters you may use depend on the specific unified search implementation of your product. For example, if your product supports a search for customers based on their name or address, you may type in part of a name or an address to find matching results. If a unified search option is not configured for your product or you do not have security access to use it then the search box may only be used for searching for menu items.

To search for business entities, you may enter your search criteria in the following ways:


- Type in your search text in **free form**. For example, if your product's unified search option supports a search for customers by name or address, the text you enter may be considered as part of a name or an address.
- Depending on your product's unified search configuration, you may use specific **hints** to explicitly identify your filtering criteria. For example, an address filter may be associated with the hint name "ad:" allowing you to be more explicit and enter "ad: main" to find records by their address.
- You may use free form text or hints but not both.

As you type in your text, the application attempts to find matching records and presents the top most results for your review. You may select a record from the results list to navigate to the corresponding portal.

Refer to your specific product documentation for more information about your product's unified search features.

Refer to [Understanding Unified Search](#) to better understand unified search configuration.

Help Menu

The Help menu is available in the application toolbar to provide access to the following product support options. Open the menu by clicking the Help icon .

Help

Select the **Help** entry in the menu to open context-sensitive help information in a new browser window. Alternatively, use the following shortcut key to launch online help directly.

Keyboard shortcut:

Keyboard Shortcut

Alt+F1

Prepare Issue Details

Select the **Prepare Issue Details** entry in the menu to launch a script to capture details of an error that needs to be reported. Refer to [Prepare Issue Details](#) for more information on this option.


About

Select the **About** entry in the menu to display a window that describes the current release information for your version of your product.

The about window shows a variety of information including:

- The logged in user ID
- The list of products installed in the application along with the release ID. This information comes from the [installation record](#). Note that in a Cloud installation, only the Cloud product name and release ID are displayed. The details of the products that make up the Cloud installation are not displayed (but are still visible on the installation record).

User Menu

The User menu is available in the application toolbar to provide access to user-specific options. Open the menu by clicking the User icon .

NOTE: Configurable Menu. Additional menu entries may be visible in your implementation as this menu configuration allows for additional options to be defined.

Preferences

Select the **Preferences** entry in the menu to view or modify your settings in a form that appears in the **Main** tab in the object display area. Alternatively, use the following shortcut key to access preferences directly.

Keyboard Shortcut

Ctrl+Alt+P

Refer to [User Preferences](#) for more information.

Switch UI View

Select the **Switch UI View** entry in the menu to change the user experience from the current system default to the prior user experience or from the prior user experience to the current version. This option is only available for users that have access to the **FUIVIEW** application service.

Logout

Select the **Logout** entry in the menu to complete your current session of the application. Clicking this item logs the current user out and displays the Welcome page so that another user can login. Alternatively, use the following shortcut key to logout.

Keyboard Shortcut

Ctrl+Alt+L

Page Title Area

The Page Title Area displays the page title and the page actions toolbar.

For certain pages in the system, the page title may be followed by an information string. It is only applicable to portal-based pages that display a single record. This information is only visible if the portal has been configured to display the primary key's information string. Typically only "stand alone" maintenance portals are configured to show the information string, whereas "all-in-one" portals are not configured this way.

FASTPATH: Refer to [Common Base Portals and Zones](#) for more information about standard base product portals.

NOTE: If an information string is long and would interfere with a page actions button, the information string will automatically be truncated and suffixed by an ellipsis, "...", indicating that more information is available. In this situation, the missing text can be viewed if the user hovers their mouse over the string.

The Page Actions Toolbar displays standard control options, such as Save and Duplicate. More detail about the actions that may appear are described in the following section.

Page Actions Toolbar

In the Page Title Area, control functions appear in the Page Actions Toolbar; such as, Clear, Save, Duplicate, Delete and Refresh buttons.

NOTE:

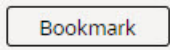
Buttons in the Page Title Area appear as the function is available to the user; for example, the Delete button will not appear if the user has made any changes to the page's content, or does not have the necessary security rights for this action.

In addition, other functions that are specific to a portal-based page may appear in the toolbar, if configured. For example, the ability to navigate to the Search zone for a specific maintenance object may be visible on the maintenance portal.

This section describes the standard features that are available in the Page Actions Toolbar.

Bookmark Button

Click the Bookmark button on the Page Actions Toolbar to save the current page and context to the bookmarks list.



Keyboard Shortcut

Ctrl+Alt+B

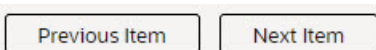
NOTE: Secured. This button is secured using the application service **F1-BOOKMARKS**. Users that do not have security for the application service will not see the Bookmark button (or the corresponding dashboard zone).

After clicking the Bookmark button, you are prompted to provide a name for the bookmark. After saving, it is visible in the [Bookmarks Zone](#), allowing you to return to this page with this context loaded with one click.

NOTE: Unsaved data will not be included in the bookmark. If you view a record and make changes to data without saving the changes and then bookmark the page, the unsaved changes are not included in the bookmark information.

You may change the name of the bookmark or change the sequence of the bookmarks by navigating to the [User - Bookmarks](#) in the user preferences.

Next/Previous Item Buttons



The **Previous Item** and **Next Item** buttons appear in the Page Actions Toolbar under these conditions:

- You access a fixed page, meaning that it is not portal-based.
- You use a search page to display an item.
- There are multiple items in the search results list *below* or *above* the selected item.

If, for example, you enter search criteria of "Smith" on the User Search page, you might see the following results:

- Smith, Albert
- Smith, Peter
- Smith, Zane

If you select **Smith, Peter**, Peter's user information is displayed on the User page. If you then click the **Next Item** button, **Smith, Zane** will be displayed. In addition, because **Smith, Zane** is the last entry in the search results, the **Next Item** button is no longer needed and will not appear in the Page Actions Toolbar.

If you then click the **Previous Item** button, **Smith, Peter** will be displayed; click again and the **Smith, Albert** record will appear, and, because **Smith, Albert** is the first entry in the search results, the **Previous Item** button is no longer needed and will not appear in the Page Actions Toolbar.

Keyboard Shortcut

Alt+P

Alternate Shortcut

Alt+N

Alt+Shift+N

NOTE: Refer to [Shortcut Key Summary](#) for information about the alternate shortcut.

Duplicate Button

Click the **Duplicate** button in the Page Actions Toolbar to create a duplicate of the currently selected object.

A rectangular button with a thin border and the text "Duplicate" centered inside.

Before the new object is added to the database, a dialog appears in which you must enter the unique identifier for the duplicate object. When you click OK, the system creates the record by copying the original record. It may be helpful to think of the duplication function as the equivalent of a "save as" action.

Keyboard Shortcut

Ctrl+Alt+D

NOTE: Copying ancillary information for a record sometimes requires confirmation during a duplication request. If so, you may also be asked if you also want to duplicate ancillary information. If, for example, you duplicate a user group, besides supplying the ID of the new user group, you are asked if all users in the original user group should be linked to the new user group.

This button does not appear in the Page Actions Toolbar when:

- Duplication is not supported for the current object.
- You are on a portal based page. (For these types of pages, if duplication is supported, the actions are defined in a zone on the portal.)
- You have made changes to the object without committing the changes to the database by clicking Save. (After making changes, you must either discard them using the **Refresh** button, or save them.)

Delete Button

Not all transactions have the capability to be deleted. Deletion is related to the business impact of the action. In some transactions, Delete is present for some scenarios and unavailable for others; for example, you may not be able to delete a record at a certain status level.

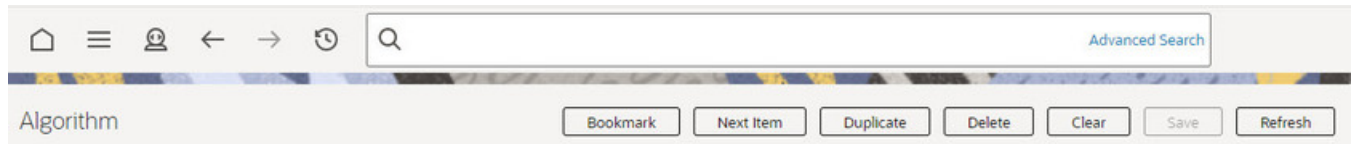
NOTE:

If you make a change to an object and do not save the changes to the database, you cannot delete the object until you either save the changes or refresh the object with the contents on the database.

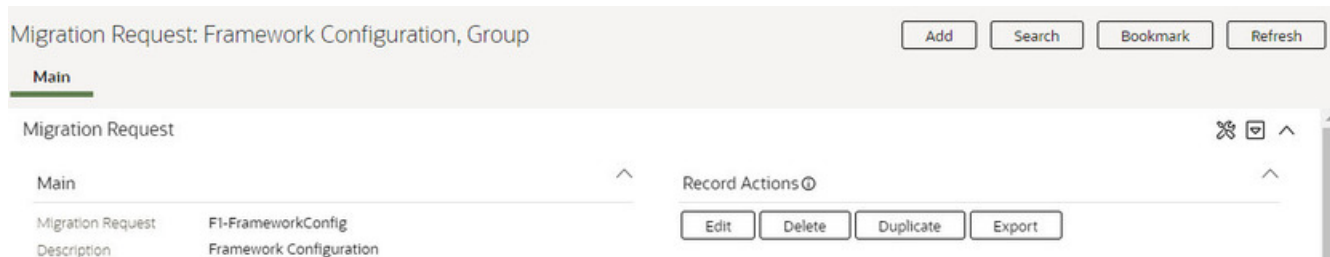
Before you delete an object, the system verifies that the object is not referenced anywhere in the system. If it is, you cannot delete it. If, for example, you attempt to delete a user, the system ensures this user has not been referenced in a user group or on a To Do.

Depending on the type of page that you are using, the option to delete may vary.

- Fixed pages display a Delete button in the page actions toolbar when the option is available.



- Portal-based pages display the Delete button in the object-related map.



- List style fixed pages commonly display the Delete button (garbage can icon) next to the entity in the list.

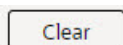


In all cases, before the object is deleted, a confirmation dialog is displayed.

CAUTION: A timeout situation could occur as the system attempts to verify the use of the deleted object if the object is referenced in many places *and* involves large volumes of production data. If this timeout situation occurs, you must work with your database administration group to delete the offending object.

Clear Button

Click the **Clear** button to remove the values from the currently displayed object's fields. This action is typically used to clear input fields when adding a new object.

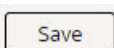


Keyboard Shortcut

Ctrl+Alt+C

Save Button

Clicking the **Save** button saves any changes you have made in the data on the current object.



Keyboard Shortcut

Alt+S

Alternate Shortcut

Alt+Shift+S

NOTE: Refer to [Shortcut Key Summary](#) for information about the alternate shortcut.

The button does not appear unless:

- You enter a page in add mode and default values are loaded.
- You make changes to an object after it is displayed (note that you must tab or click out of the field before the Save button appears).

Refresh Button

Click the Refresh button to reload the values on the page with those from the database.

CAUTION: All unsaved changes are discarded.

Refresh

Keyboard Shortcut

Alt+R

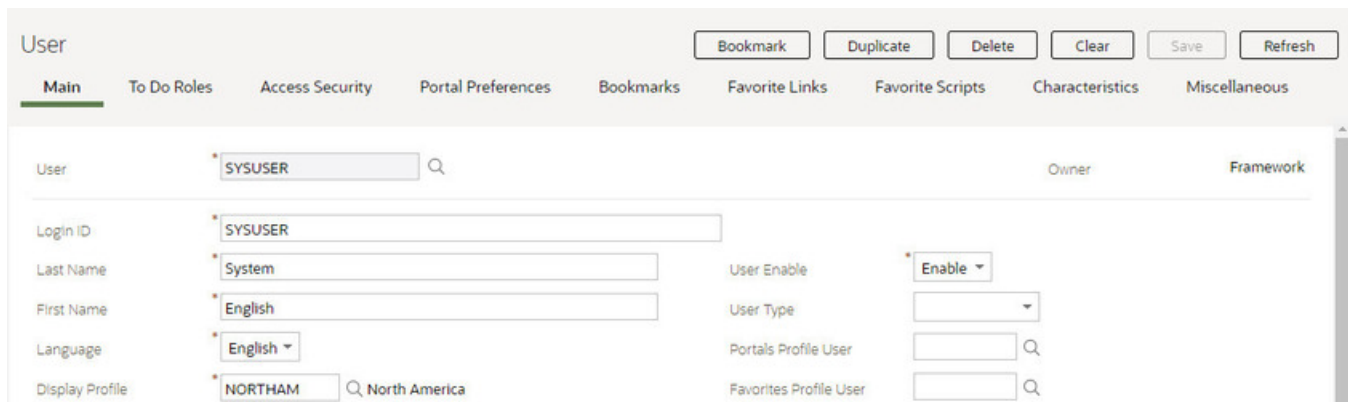
Object Display Area

The Object Display Area contains the details of the object(s) currently being viewed or maintained. For example, if you display a portal, its zones are displayed in this area.

Because many of the page components follow Internet standards of operation, the behavior of these types of components are not explicitly described. This section focuses on special features that are unique to the application framework.

Multiple Tabs

When a page in the application includes a lot of information, the page may be implemented with vertical scrolling to allow the user to view all the information. Or alternatively, the page may be implemented with several tabs, allowing the user to navigate between tabs to view or update information. The following image is a section of the User page showing the multiple tabs.



The screenshot shows a web application interface for a 'User' page. At the top, there are several action buttons: 'Bookmark', 'Duplicate', 'Delete', 'Clear', 'Save', and 'Refresh'. Below these are several tabs: 'Main', 'To Do Roles', 'Access Security', 'Portal Preferences', 'Bookmarks', 'Favorite Links', 'Favorite Scripts', 'Characteristics', and 'Miscellaneous'. The 'Main' tab is currently selected. The main content area contains a search bar for 'User' with the value 'SYSUSER' and a search icon. Below the search bar are several form fields: 'Login ID' (SYSUSER), 'Last Name' (System), 'First Name' (English), 'Language' (English), and 'Display Profile' (NORTHAM). To the right of these fields are 'User Enable' (Enable), 'User Type' (dropdown), 'Portals Profile User' (searchable), and 'Favorites Profile User' (searchable).

Figure 1: Multiple Tabs Example

The names of the tabs describe the type of information that will be displayed in the page when the tab is clicked. For example, if you click on the **Favorite Links** tab, the contents of the page show the favorite links configured for the user.

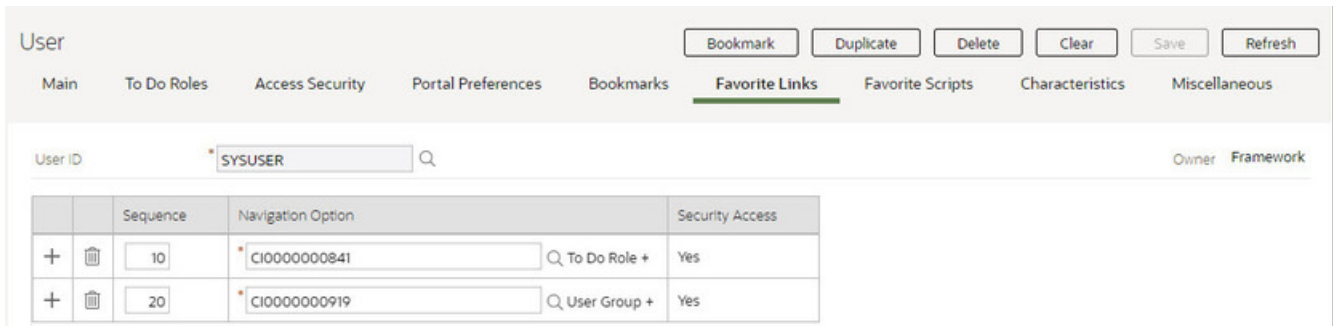


Figure 2: Multiple Tabs Example (After Favorite Links Tab Is Clicked)

NOTE: Hot key options. Rather than clicking on a tab, you can click **F2** or **Shift+F2** to scroll through the tabs. **F2** moves you through the tabs from left to right. **Shift+F2** moves you through the tabs from right to left.

NOTE: Hot key options. Use **Alt+** a number to navigate to the tab page whose relative position corresponds with the number. For example, if you press **Alt+6** on the above page, the **Favorite Links** tab will be displayed. **Alt+0** may be used to navigate to the tenth tab. If there are more than 10 tabs on a page, you must use the mouse or **F2** / **Shift+F2** to display additional tabs. Note that the numeric keypad on your keyboard cannot be used to implement this function.

NOTE: Returning to a page that contains multiple tabs. If you return to a page (by clicking the **Back** arrow or the **History** icon), the page will be positioned on the last tab that was opened.

NOTE: Page versus Tab nomenclature. The documentation frequently refers to the contents of a given tab as a page. However, the page actually comprises the entire collection of tabs.

Text Box

The following is an example of a basic text box:

Description

If the text box is a large field, the text box may be configured to enable scrolling. In this situation, an expanded edit icon, (a pencil), will appear adjacent to it.

Detailed Description

This BO enter algorithm creates a To Do entry.

The To Do Type is set as follows:

- > If To Do Type Element Name is defined, the value is retrieved from this element in the object's related type.
- > If no Element Name is defined or no To Do Type is found on the related 'type', the To Do Type parm is used.
- If no To Do Type is found, a log is added for the record with the error.

The To Do Role is set as follows:

- > If To Do Role Element Name is defined, the value is retrieved from this element in the object's related type.
- > If no Element Name is defined or no To Do role is found on the related 'type', the To Do Role parm is used.

Additional capabilities are provided to display and edit data in these text boxes:

- If the field contains more text than is visible, a pop-up that displays the full text appears when you hover your mouse over the text box. The text in the pop-up can be selected and copied.

Detailed Description

This zone type is used to create an info zone. Info zones contain a list of information about a given object. For example, you could set up an info zone to produce a list of all contacts made by the customer in context.

Info zones offer many features that can be used to change how the list looks "on the fly" (by end users). For example, users can add additional columns to the list, change the sort order, ...

For a more detailed description of how these zones look and behave, open the help and navigate to the index entry titled "info zone configuration".

To set up a new zone, fill in the zone's parameters as per each parameter's Comments. For examples, please refer to the "tips" that appear in

Java Class Name

		Sequence
+	🗑️	* <input type="text" value="10"/>
-	🏠	* <input type="text" value="11"/>

* This zone type is used to create an info zone. Info zones contain a list of information about a given object. For example, you could set up an info zone to produce a list of all contacts made by the customer in context.

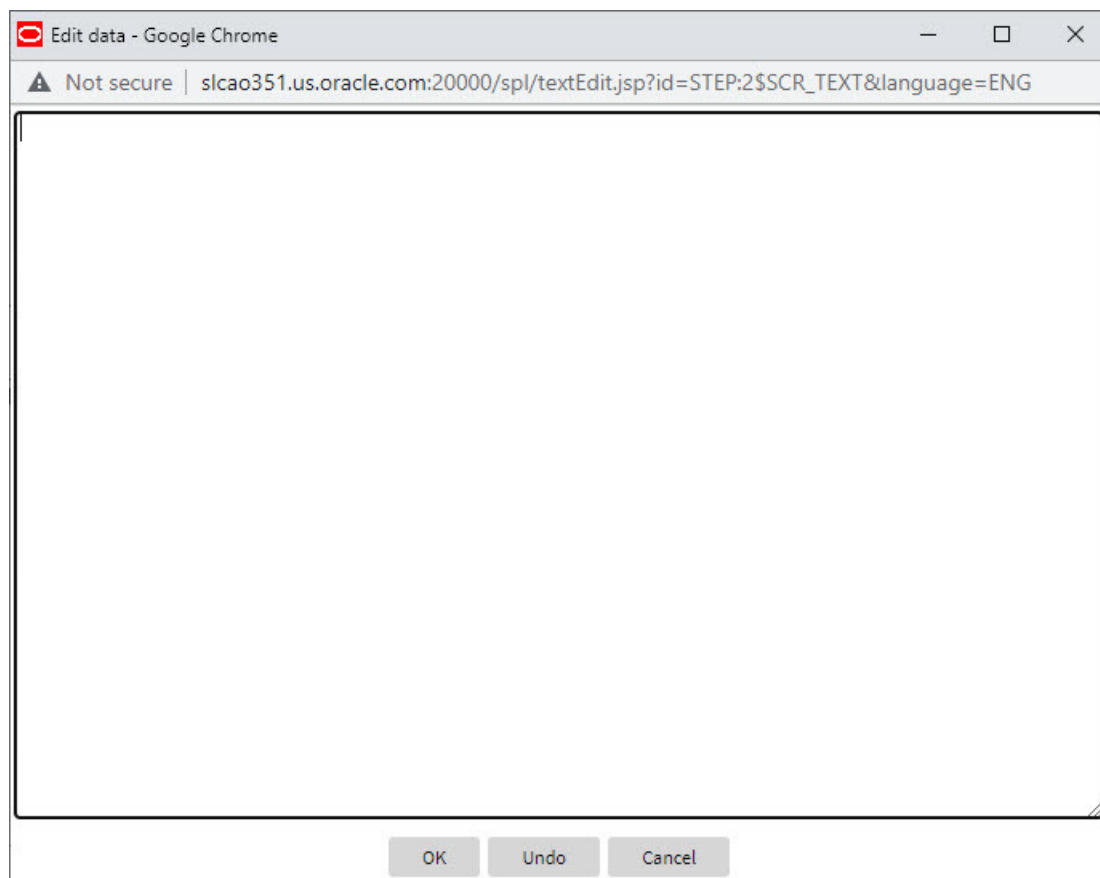
Info zones offer many features that can be used to change how the list looks "on the fly" (by end users). For example, users can add additional columns to the list, change the sort order, ...

For a more detailed description of how these zones look and behave, open the help and navigate to the index entry titled "info zone configuration".

To set up a new zone, fill in the zone's parameters as per each parameter's Comments. For examples, please refer to the "tips" that appear in the dashboard.

This zone's parameters support a single SQL statement. If you require a zone that is built using multiple SQL statements (e.g., if you need to retrieve rows from disparate sources), please use the Multi-SQL version of this zone type.

- If you click the expanded edit icon, an editable pop-up window appears. If the field is not protected as a result of business rules for the page, you may modify the text in this window.



NOTE: The following table highlights special characters that are converted to equivalent standard characters when entered in a text box.

Entered Character Description	Unicode	Converted Character Description
Curly left double quote.	U+201D	Straight double quote.
Curly right double quote.	U+201C	Straight double quote.
Curly left single quote.	U+2019	Straight single quote.

Entered Character Description	Unicode	Converted Character Description
Curly right single quote.	U+2018	Straight single quote.
En dash.	U+2013	Simple hyphen.
Em dash.	U+2014	Simple hyphen.
Ellipse	U+2026	Three dots.
Middle dot (bullet / centered dot)	U+00B7	Asterisk.
Fraction 1/4 (using 1 character).	U+00BC	1/4 (using three separate characters).
Fraction 1/2 (using 1 character).	U+00BD	1/2 (using three separate characters).
Fraction 3/4 (using 1 character).	U+00BE	3/4 (using three separate characters).

Date and Time Fields

When both the date and time are required, the Date and Time fields typically look as follows:

Date / Time

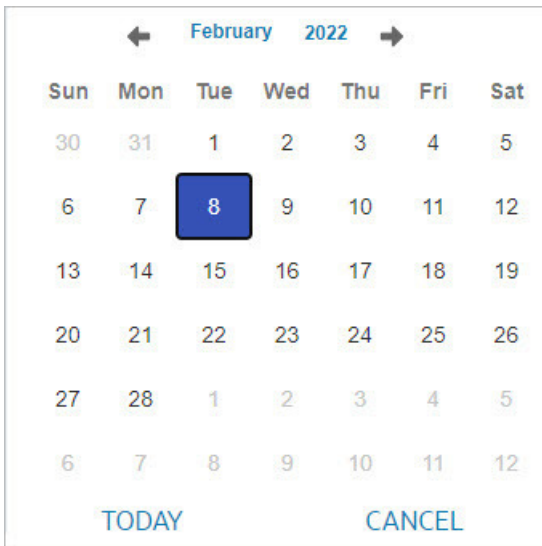
When only the date is required, the Date field looks as follows:

Date

When only the time is required, the Time field looks as follows:

Time

If you click on the calendar icon or press the Enter key while the cursor is positioned to the left of the icon in a Date or Date and Time field, the following window appears:



Click or press enter to select the date and populate the input field. Use the left and right arrows to scroll back or forward a month. Click on the 'Month' text to bring up a range of months to select from. Click on the year to bring up a range of years to select from.

If you click on the clock icon or press the Enter key while the cursor is positioned to the left of the icon in a Time or Date and Time field, the following window appears:

12	01	02	03	04	05	06	07	08	09	10	11	AM
12	01	02	03	04	05	06	07	08	09	10	11	PM
:00	:05	:10	:15	:20	:25	:30	:35	:40	:45	:50	:55	↓

CANCEL

Select the hour then click or press enter on the minutes to select the time and populate the input field.

Search Button

The search button appears primarily as a magnifying glass.

Algorithm Type

When using a pop-up search window, the search button can appear as a button with its function labeled.

You can have the system search for an item that matches the data you entered using any of the following methods:

- Enter a value in the search field and press **Enter** while the cursor is in the search field. The system assumes you want to search for the value entered and pre-populates it in the search window's search criteria. If a unique match can be found, the system automatically populates the field with details corresponding to the record found. If a unique match cannot be found (either no matches or multiple matches), a search page opens containing selectable items.
- Click the Search button without entering a criteria. This will typically cause a pop-up search window to appear so that you can further refine your search.
- Enter a value and tab out of the search field. The system looks for a value that exactly matches what you entered. If it cannot find such a value, the search dialog will not be presented.

When an item is selected in the search page, the search page closes and the input field is populated with the value selected.

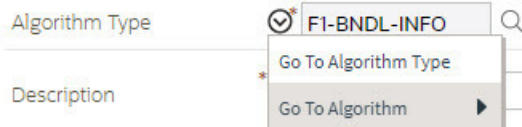
NOTE: Sorting columns within the search grid. You may click on the column heading of columns in a search grid to cause the contents of the grid to be sorted. Refer to [Sorting Functionality](#) for more information.

Context Menu Button

The Context Menu button is a small arrow that appears adjacent to an object's value or information string. An example is shown below:








Context menus exist to help you navigate to other pages and populating the page with the current context (by passing the ID of the related object). In addition, the menu items offered in a Context Menu differ depending on the entity that the menu is related to; for example, when you click the Algorithm Type context menu, a pop-up menu similar to this appears:




Different context menus exist for each of the major objects in the system; for example, in Customer Care and Billing, the context menu for Premise contains premise-oriented menu items, whereas the context menu for Account contains account-oriented menu items.

Go To Button

Clicking a Go To button, , opens the page for the related object - it functions like a hyperlink.

		User Group	Expiration Date	Owner
+		 <input type="text" value="ALL_SERVICES"/>  System User Group	<input type="text" value="12-31-2100"/> 	Framework

Broadcast Button

Some zones have been configured to "broadcast" unique identifiers to other zones within the portal; if so, you'll see a broadcast icon, , in the first column. Clicking the icon causes specific zones to reload - these zones that have been configured to rebuild when the unique identifier is received. For example, you might have configured a zone to display the image of a bill when a bill ID is broadcast from an info zone.


Embedded Help Icon

The icon containing a small 'i' is an Embedded Help icon.



When an embedded help icon is clicked, a tool tip appears containing additional information.

Work Lists


The work list buttons can appear in a zone's column heading, .

Clicking the icon results in populating the dashboard [Work List Zone](#) with the entire column. The user may use the Work List zone to easily process entries one at a time without returning back to the source query zone. The work list stays populated until the user overlays the work list by clicking a different column's icon.

When a user clicks on a work list entry in the dashboard, the system behaves as if the user clicked on the hyperlink in the zone. In addition, the system adds a check mark next to the item showing that the user has "worked on" or visited the entry.


NOTE: The Work List zone is populated only upon user request.

Scroll Bars with Dynamic Height

Sometimes the height of a grid isn't sufficient to display a grid's rows. When this occurs, a vertical scroll bar appears and an "expand button" appears above this bar, . For example, the following grid is not high enough to show all of the available dashboard zones:

Dashboard - 10 of 10 zones selected for display.


Zone	Display	Collapsed	Sequence	Refresh	Security Access
Switch Language	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="20"/>	<input type="text" value="0"/>	Yes
Bookmarks	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes
Collect Entity	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes




If you want to view all of the available zones without using the vertical scroll bar, you can click the expand button. When clicked, the height of the grid expands to show all rows in the grid:

Dashboard - 10 of 10 zones selected for display.

Zone	Display	Collapsed	Sequence	Refresh	Security Access
Switch Language	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="20"/>	<input type="text" value="0"/>	Yes
Bookmarks	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes
Collect Entity	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes
Current To Do	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes
Favorite Links	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes
Favorite Scripts	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes
Favorite Searches	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes
Tags	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes
To Do Summary	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes
Work List	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	Yes





Notice, after the grid is expanded, the grid's vertical scroll bar disappears and the button in the upper corner becomes a Collapse Button, . If you click the collapse button, the grid will return to its original height.

NOTE: Search grids. Expand all functionality is not available in search grids.

Sorting Functionality

Grids have a heading row that contains labels for the columns. By clicking a column heading, you cause the rows to be sorted in the order of the selected column. By clicking on a column more than once, you cause the rows to be sorted in reverse order. Clicking the heading a third time removes the sorting function and returns the grid to its default display organization.

The following shows how information look after organizing the grid, clicking the column heading, by algorithm type (notice the icon in the column name).

	Algorithm	Description 	Algorithm Type 
1	F1-OVRDT-ALG	Validates date in format YYYY-MM-DD	ADHV-DTD
2	F1-ACTIVEWS	Web Service - Verify Active Operations	F1-ACTIVEWS
3	F1-ATCHM-INF	Attachment Information	F1-ATCHM-INF
4	F1-ATSZCLVL	Attachment Size Class Validation	F1-ATSZCLVL
5	F1-BAT-LSDEF	Timed Batch - Service Level Based on 1.5 Timer Interval Factor	F1-BAT-LVSVC
6	F1-BFEMOLOG	Add Business Flag Error MO Log	F1-BFEMOLOG
7	F1-BFSTDNMVA	Business Flag Standard Name Validation	F1-BFSTDNMVA

NOTE: Sorting on dropdown values. If you attempt to sort a column that contains a dropdown value, the information is sorted in the order of the unique identifier associated with the dropdown value, not the description you see on the screen; for example, if you sort on a column containing unit of measure, the information will be sorted in the order of the unit of measure code, not the description that appears in the drop down. This means that the resultant sort will appear incorrect and therefore we discourage you from doing this.

Get More Functionality

Grids in fixed pages that can potentially contain a large number of rows have special processing that we refer to as Get More functionality. Rather than show every row, the system retrieves the first X records (where X varies from grid to grid, and from release to release) and appends a Get More button to the bottom of the grid:


Get More...

When you click the **Get More** button, the system appends the next X rows to the bottom of the grid.


NOTE: The absence of a "get more" button. For pages configured to include Get More, if no button appears at the bottom of the grid, this means all of the rows are displayed.

NOTE: Search grids. The Get More function is not available in search grids.

Adding and Removing Rows Using Arrows










In some editable grids, a new row can be added to another editable grid (the grid on the left) by clicking the Add item button, , of the desired row in the grid on the right:

	Sequence	Description		Sequence	Description
↩	10	Winter kWh charge, first 500	↩	5	Service Charge
↩	20	Winter kWh charge, remaining			

When the Remove item button is clicked, , the row is transferred from the grid on the right to the grid on the left. This change will not be committed to the database until you save the object being maintained

Reorganizing Items Using the Up and Down Arrows

Occasionally, rows can be repositioned in an editable grid by clicking up and down arrows. The following is an example of such a grid:

		Merge Type	Sequence	SQ Rule
	 	Original	10	DA Credits
	 	Original	20	Calc. Number of Days In Bill Period
	 	Original	30	Copy SQs from Master SA to sub SA

Notice the following about such a grid:

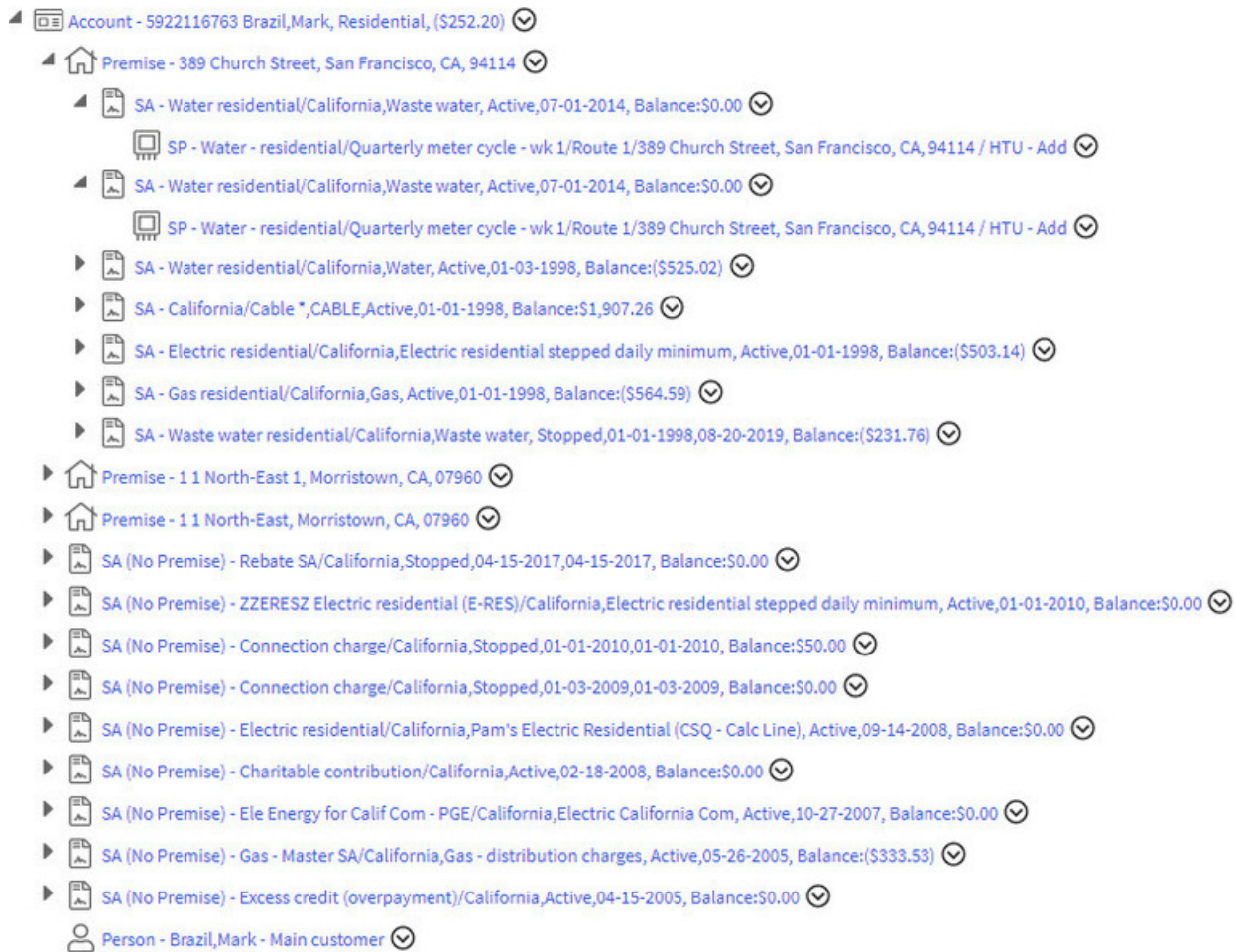
- The up arrow is dim in the first row. This is because it can't be moved further to the top.
- The down arrow is dim in the last row. This is because it can't be moved further down.
- The up and down arrows are usable in all other rows. Each time you click one of these arrows, the respective row is moved up or down one row.

NOTE: Reminder. When you move rows into a grid, the system does not apply the change to the database until you click the Save button (which commits the entire object to the database).

Trees

Trees are used to illustrate complex relationships between objects. The following is an example from Oracle Utilities Customer Care and Billing of a page with a tree that is expanded:

Account Tree



The following points describe how to use trees:

You can expand and contract the nodes in a tree by clicking on the icon that prefixes the node (you can also click the arrows to expand and contract a node).

NOTE: Expand all. If you press `Ctrl` while clicking on an icon that prefixes a node, all child nodes (and their children) expand.

You can click the context menu button to cause a context menu to appear. You can navigate to any of the pages that are presented on this menu. Refer to [Context Menu Button](#) for more information.

You can click on the information that appears in an underlined node to drill into the page in which the object is maintained. For example, if you click on the account information, you will be transferred to the Account - Main page.

NOTE: Important! If you make changes after expanding a tree's nodes (e.g., by drilling down on a Premise and changing the premise's address), the tree will contract to the first node if you return to the page.

Nodes in trees fit into two broad categories:

- **Information nodes.** Information nodes highlight the existence of other data. For example, on the Control Central - Bill Payment Tree, a node exists to highlight the existence of new financial transactions that were created since the last bill was completed.

- **Data nodes.** Data nodes show information about objects in the system. For example, data nodes exist on the Control Central - Bill Payment Tree that show bills and payments.


Required Field Indicator

An asterisk (*) is used to indicate that a field is required. The system determines this based on the field's configuration. When a field becomes required because of an algorithm and not the field's configuration, however, the system will not add an asterisk.

NOTE:

If a page includes a grid, fields within a row may be marked as required.

Operations 





		Operation Name	Description	Schema Type
+		*	*	*

In this situation, the field(s) within the grid are **only** required if the row is being populated.

Accordion Windows

Accordion windows are used when the data held in an object's rows does not comfortably fit into a grid.

Each row in an accordion has a header that summarizes a row's details. When you click a header, the row expands to show the details. The following is an example of an accordion where all headers are collapsed (meaning that only summaries of the rows' details are shown):

<input type="checkbox"/> Steps		All 3 displayed	[Expand All] [Collapse All]	Copy 
<input type="checkbox"/>	 (10)	Edit data - Get the Physical BO for the MO		
<input type="checkbox"/>	 (20)	Edit data - Get the prime keys of the BO		
<input type="checkbox"/>	 (30)	Edit data - Read the BO		

When you expand a header, its details are exposed. For example, the following shows the same accordion after the third row is expanded:

Steps		All 3 displayed	[Expand All] [Collapse All]
<input type="checkbox"/>	<input type="plus"/> <input type="trash"/> (10)	Edit data - Get the Physical BO for the MO	
Step Sequence *	<input type="text" value="10"/>		
Step Type *	<input type="text" value="Edit data"/>		
Text	Get the Physical BO for the MO		
Edit Data Text	<pre>move "parm/mo" to "F1-RetrieveMOOption/input/mo"; move 'F1PB' to "F1-RetrieveMOOption/input/moOptionType"; move 'false' to "F1-RetrieveMOOption/input/isMultiple"; invokeBS 'F1-RetrieveMOOption' using "F1-RetrieveMOOption";</pre>		
<input type="checkbox"/>	<input type="plus"/> <input type="trash"/> (20)	Edit data - Get the prime keys of the BO	
<input type="checkbox"/>	<input type="plus"/> <input type="trash"/> (30)	Edit data - Read the BO	

NOTE: You must expand to update. In order to update a row in an accordion, you must expand it.

Row Summary and Chunk Management

At the top of an accordion you'll find a summary of its rows. The format of this information differs depending on the number of rows and the size of the accordion:

- If the accordion contains every row, a summary of the total number of rows is displayed.
- If the accordion contains a subset of rows, you are presented with a summary of the rows AND you are provided with options to display different "chunks" of rows.
 - Click **First** to display the first "chunk" of rows (the size of a chunk differs depending on the accordion). This option is dimmed if the first chunk is in the accordion.
 - Click **Previous** to display the previous chunk of rows. This option is dimmed if the first chunk is in the accordion.
 - Click **Next** to display the next chunk of rows. This option is dimmed if the last chunk is in the accordion.
 - Click **Last** to display the last chunk of rows. This option is dimmed if the last chunk is in the accordion.

Dashboard Area

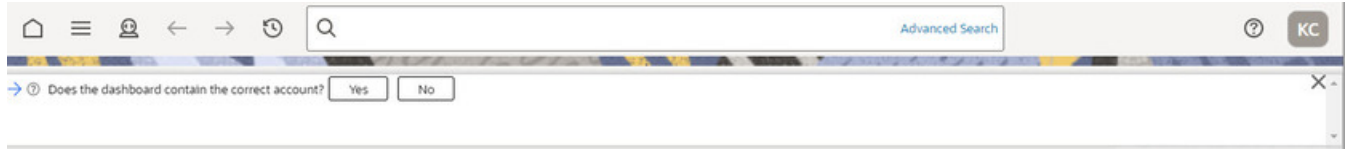
The dashboard is a [portal](#) that contains zones. These zones appear regardless of the object being displayed in the [Object Display Area](#). Refer to [Dashboard Portal](#) for a description of these zones. Every user has control over the zones that appear in their dashboard. Refer to [Portal Preferences](#) for more information about how a user can customize the contents of the dashboard.

The dashboard area is displayed to the left of the Object Display area by default but you have control over its location, width and more as part of your [user preferences](#).

Script Area

When a [script is initiated](#), the [object display area](#) shifts down to make room for the script area.

The information in the script area takes you through the steps in the selected business process (provided by your implementation team).



Portal Pages

This section describes using portals and zones to find and display information. This is the preferred technique used by the base product for building user interfaces.

Additionally, portals and zones may be configured by your implementation to supplement the base package provided functionality.

The contents of this section provide background information about portal page behavior.

Portals Are Made Up Of Zones

The term "zone" is used to describe a section on a portal. Each zone has a heading and content.

Keyboard Shortcuts	Description
Alt+]]	Moves the cursor focus to the next zone on the page based on the standard browser keyboard navigation "order".
Shift+Alt+]]	Moves the cursor focus to the previous zone on the page based on the standard browser keyboard navigation "order".

The following is a portal with seven zones:

- Switch Language
- Bookmarks
- Checked Out
- Current Context
- Current To Do
- Favorite Links
- Favorite Scripts

NOTE: This example may not be available in your application; it is used for illustration purposes only.

The screenshot displays a vertical sidebar with several expandable/collapsible sections. At the top is 'Current Context' with an upward arrow, containing three items: 'Brazil,Mark' with a person icon and a checkmark, '5922116763 4 Brazil,Mark, Residential, (\$252.20)' with a calendar icon and a checkmark, and '11 North-East, Morristown, CA, 07960' with a house icon and a checkmark. Below is 'Current To Do' with an upward arrow and a button labeled 'Assign me a To Do'. The 'Customer Contact' section has an upward arrow and contains a 'Last' timestamp '175 days ago - Dandugula,Krishna', a 'Type' dropdown menu, a 'Comment' text area, and an 'Add Contact' button. Further down are 'Eligible Scripts' with a downward arrow, 'Favorite Links' with an upward arrow, and two expandable items: 'User Group +' with a count of [1] and 'To Do Role +' with a count of [2]. A vertical scrollbar is visible on the right side of the sidebar.

Users Must Be Granted Security Access To Each Zone

A user must be security granted access rights in order to see a zone on a portal.

FASTPATH: Refer to [Granting Access to Zones](#) for information .

Zones May Appear Collapsed When A Page Opens

Zones may be designed to appear collapsed (or minimized) when the portal is initially displayed. To view the information in

a collapsed zone, the user can press the zone's expansion button, , to either expand or collapse the zone:

Favorite Links 

Favorite Links	^
To Do Role +	[1]
User Group +	[2]

A zone's data is only retrieved when it is expanded.

Some Portals Allow User Customization

There are some portals in each base product that are configured to allow user customization. In this case, the portal is visible on the user's portal preferences. For these portals each user has control over the following:

- Which zones appear on their portal
- The order in which the zones appear
- Whether the zones should be "collapsed" (i.e., minimized) when the portal opens.

Users define their preferences by clicking the [User Preferences](#) button. Pressing this button transfers the user to the [User Preferences](#) page on which their [Portal Preferences](#) can be defined.

Portals used for base product query and maintenance pages are typically configured to not allow user customization.

Dashboard Portal

The dashboard is a special portal that always appears on the desktop. Its zones contain tools and data that are useful regardless of the object being displayed.

NOTE:

Minimize the dashboard. You can minimize the dashboard by clicking on the vertical bar between the dashboard and the object display area. This causes the dashboard portal to disappear and displays a single vertical dashboard bar, allowing more of the page area to display. It also suppresses the refresh of the zones within the dashboard based on updates to the context. Clicking the dashboard bar restores the display of the dashboard portal. Maximizing the dashboard then refreshes the zones with the latest appropriate content.

Minimize the dashboard when launching the system. If desired, a URL parameter may be used to minimize the dashboard when launching the system, using the parameter `minDashboard=true`. Refer to [Creating Application Links in External Applications](#) for more information and examples.

Adjusting the dashboard width. You can adjust the width of the dashboard by moving the slider that separates the dashboard from the object display area. This also updates your user preferences with the new width.

Keyboard Shortcut

Alt+J

The contents of this section describe the zones that are provided by the framework product for this portal. Please note the following additional information about the dashboard portal that may impact how it appears.

- **Additional zones may be available in other products.** The zones described below are available in every base product. Please refer to the product-specific documentation for additional dashboard zones available in your product (open the help and navigate to the index entry labeled **Dashboard**).
- **Context Sensitive Zones may appear.** Some pages may define additional zones that should appear in the dashboard only if that page is displayed. These are referred to as Context Sensitive Zones.

- **Dashboard zones are configurable.** The dashboard portal is configured to allow a user to define preferences for it. Refer to [Portal Preferences](#) for more information.

NOTE: Security. As a general rule, dashboard zones each refer to a unique application service so that a security administrator may configure which users are allowed to have access to which zones. Note that the user portal preferences indicates which zones a user has security access to.

Bookmarks Zone

After clicking the [Bookmark](#) button and saving a bookmark, it is visible in the **Bookmarks** zone. Clicking one of the links in the zone navigates to the page where the bookmark was saved and loads the information related to the record that was displayed when the bookmark was saved.

Click the trash can icon to delete the bookmark.

Note that a user may change the name of the bookmark or change the sequence of the bookmarks by navigating to the [User - Bookmarks](#) in the user preferences.

Tags Zone

The **Tags** zone provides a quick way to associate the entity currently being displayed with an entity tag or remove any of the tags it current has. Tagging an entity may be useful to highlight various contexts in which it can be used, for example for testing or reporting purposes. Refer to [Defining Entity Tags](#) for more information.

Note that the zone is visible only when there is an entity in context, i.e. there is an entity to tag, and the user has application security access to this zone.

Collect Entity Zone

The **Collect Entity** zone provides a quick way to add or remove the entity currently being displayed to and from the [migration request](#) in context. This is mainly useful when reviewing data to export and while doing so maintain the export list. The zone automatically appears once an **Entity List** migration request is visited and set in context.

Current To Do Zone

The **Current To Do** zone is designed to streamline the process of working on To Do entries from assignment to completion. It displays key information about the user's current To Do entry and provides functions that help minimize the number of steps it takes to resolve the entry and potentially other entries related to it.

The following sections provide more detail about the zone.

Next To Do Assignment

Users may have the option to click the **Assign me a To Do** button to request the next To Do entry they should work on. This functionality is optional and can be configured to follow your organization's business rules in determining the next task for a user. For example, if your organization practices work distribution "on demand", this feature can be configured to find the highest priority entry a user can work on and assign it to the user. Refer to [work distribution](#) for more information on this business practice.

NOTE: Next To Do Assignment Rules Are In A Plug-in. Refer to [Installation Options](#) for more information on how to configure the system to take advantage of this function.


If the system is successful in determining the next entry the user should work on, this entry becomes the user's current To Do and its information is displayed on the zone.

User's Current To Do

The user's current To Do may be set using the Next Assignment algorithm, or by drilling in on a To Do from the To Do List or To Do Search page as described in [A User's Current To Do](#). Once a current To Do has been set, the zone displays details related to the To Do.

The **Message** describes why the entry exists. The message description allows the user to drill into the page on which the entry's respective object is displayed. For example, if the entry is associated with an asset that needs review, the user is brought to the asset page to view the information and potentially take action. If the **To Do type** has been configured to associate a **script** with this To Do entry's message number, then instead of being taken to the respective page, the associated script is launched.

Info displays the standard To Do information. An icon  appears if there are comments on the To Do and a Log Entry

icon  appears if there is a log entry of type **Forwarded**, **Sent Back** or **User Details** on the To Do. The description drills into the **To Do entry** page allowing the user to view more information about the entry or make changes to it as needed.

The user may use the **Complete** button in the zone to complete the current entry.

Next / Previous Buttons

The **Previous** and **Next** buttons are enabled on this zone under the following conditions:

- The user selected an entry either on the [To Do List](#) page or the [To Do Search](#) page.
- At least one other entry exists *below* (for Next) or *above* the entry selected (for Previous).

Clicking **Next** or **Previous** will bring the user to the transaction indicated on the To Do Entry's message, where its respective object is displayed. It also makes that To Do the Current To Do displayed in the zone.

Keyboard Shortcut	Comments	Alternate Shortcut
Alt+Y	Previous To Do	
Alt+Z	Next To Do	Alt+Shift+Z

NOTE: Refer to [Shortcut Key Summary](#) for information about the alternate shortcut.

NOTE: When a user drills down on a To Do entry from the To Do List page, the system automatically changes the status of the entry to **Being Worked**. If **Next To Do** or **Previous To Do** is subsequently used to scroll through this list, the system does *not* change the To Do entry's status. To change the status of these entries, the user must navigate to the [To Do List](#). On this page, multiple entries may be selected and their status changed with a single user-interface action.

Related To Do Entries

If the To Do entry has non-complete [related entries](#) a summary **Related To Do** description appears. The description states the total number of related entries, how many are still in **Open**, how many are assigned to current user and how many assigned to others. This hyperlink navigates to the [To Do Search](#) query allowing the user to review the related entries. At this point the user may decide to assign entries that can be positively identified as being caused by the same problem as the current To Do also to himself.

If at least one related To Do entry is **Open** or also assigned to the current user, the **Complete All** button is visible. If the user determines that the related To Do entries have been resolved with the appropriate action taken, the button may be used to complete in one step the current entry and all related entries that are **Open** or assigned to the current user.

If the related entries are not resolved with one action, the user may opt to use the **Complete** button to complete the current entry alone and work on related entries one at a time. Using the related To Do information hyperlink the user may navigate to the To Do search page and drill into the first related To Do to work on using the message hyperlink. This entry now becomes the user's current To Do and the **Next** and **Previous** action buttons are set to scroll through the related entries list.

Favorite Links Zone

Each line in the **Favorite Links Zone** corresponds with one of the user's favorite links (e.g., transactions). Clicking on a line invokes the transaction. The first nine entries can be invoked by pressing an accelerator key (Ctrl + the number of the relative position of the link in the list). Note that the numeric keypad on your keyboard cannot be used to implement this function.

Users define their favorite links by clicking the [User Preferences](#) button. Clicking this button transfers the user to the [User Preferences](#) page on which their [Favorite Links](#) can be defined.

Favorite Scripts Zone

The Favorite Scripts zone lists the business process assistant (BPA) scripts that you have defined on your [user preferences](#). You can click on a script to [execute](#) it. You can click the **Search** button in the zone to search for other scripts.

NOTE: If you do not see the Favorite Scripts zone in your dashboard, navigate to User Menu and click the **Preferences** menu item to display the [User - Portal Preferences](#) page. Under the Dashboard Portal row, select the **Display** option for the Favorite Scripts zone.

Favorite Searches Zone

If you marked a saved search as a favorite using [Saved Searches](#) functionality, the search is visible in the **Favorite Searches** zone. Clicking one of the links in the zone navigates to the page where the search was saved and populates the filter values and executes the search.

Switch Language Zone

The **Switch Language** zone is used to switch the application's language. Only languages that are supported by your implementation will appear.

To switch the application's language, choose the desired language and press the **Refresh** button.

NOTE: If your implementation uses a single language, you should suppress this zone by [not granting security rights](#) to the zone's application service.

To Do Summary Zone

The **To Do Summary Zone** has several purposes:

- It presents a summary of the [To Do entries](#) that have been assigned to you and are **Being Worked On**.
- The colored "age bars" highlight the age of the To Do entries. The colors red, yellow and green are used to highlight the relative number of old, middle-aged and new To Do entries. The age associated with each of these colors is [user-configurable](#). Note, you can hover the cursor over an "age bar" to see the number of To Do entries.
- If you click on the To Do Type description or on an "age bar", you'll be transferred to the [To Do List](#) transaction where these To Do entries will be displayed. From this page, you can drill down to the specific entries.

NOTE: Time saver! After you drill down on entry from the [To Do List](#) page, if you have the [Current To Do](#) zone configured in the dashboard, you can use the **Previous To Do** and **Next To Do** buttons in the zone to scroll through additional To Do list entries without returning to the **To Do List** page.

Work List Zone

The **Work List** zone provides quick access to an entire column of an info or query zone. When a user clicks on a work list entry, the system will do whatever it would have done if they'd clicked on the hyperlink in the zone. In addition, the system ticks off that they've worked on the entry (this feature is mostly so users know that they've at least looked at something). Refer to [Work Lists](#) to better understand how to configure query columns to support work lists.

Zone Types and Characteristics

The topics in this section describe common zone types, function, and common characteristics.

FASTPATH: Refer to [The Big Picture of Portals and Zones](#) for a description of general portal and zone functionality.

Common Characteristics of Zones

The following topics describe common characteristics of all zone types supported by the base package.

Zone Visibility

A zone may be configured to dynamically be visible or completely hidden depending on the information being displayed. When hidden, not even the header is displayed.

For configuration information, refer to [Zone Visibility Service Script](#).

Control Zone Headers

A zone may reference a UI Map to define and control its header. This feature provides the means to achieve a uniform look and feel for the application as well as add functionality to the zone header beyond being just a title bar to open and close the zone. Refer to [UI Map](#) and to its context sensitive tips zone for more information on **Fragment** UI Maps and how they are used to construct zone headers.

Explorer Zones

The product provide several zone types that are used to find or display a set of information. These are called explorer zones. There are two types of explorer zones:

- **Info zones.** An info zone displays a list of information about an object. These types of zones often display information without requiring any input from the user.
- **Query zones.** A query zone allows users to find objects matching a given search criteria.

Virtually every aspect of an info or query zone is controlled by its zone [configuration](#). However, most share basic common components described in the subsequent sections.




Search Criteria / Filter Area

For query zones, the Search Criteria Area (also called Filter Area) contains the criteria used to find objects. The design of the zone controls the number and type of filters that appear in this area. After search criteria is entered, clicking the **Search** button causes the system to look for records that match the criteria. The results are shown in the Results Area.


The system automatically collapses the filter area to allow more space for displaying the results. At that time the **Expand Filters** button may be used to open the filter area to enter different criteria. When the filter area is expanded the **Hide Filters** button may be used to collapse the filter area again without performing a new search.

For info zones, the zone's configuration indicates whether or not a Filter area is applicable and if so, whether or not it is open or closed by default. If the filter area is configured, the **Show Filter** icon is visible in the zone header. Click this icon to expose the Filter Area, if it is not visible and click the icon to hide the Filter Area if desired. Use this area to filter the information that appears in the zone by different values. Click the **Search** button to rebuild the Results Area using the specified filters.

For example, the following shows an info zone listing user groups that do not have access to a given Application Service where a filter area has been configured to allow users to show user groups specific to a given user by ID or name.

User Groups not Linked   

Filters: Application Service CILZRBPP , User ID EUBALDO

	User Group	Description 	Grant Access
1	*PORTAL*	Portal System Default	<input type="button" value="Grant Access"/>

Use to show user groups with specific user

User ID

First Name

Last Name

The info zone includes configuration to indicate whether the filter area, when exposed, should be shown at the top of the zone (above the results) or at the bottom of the zone (below the results). For query zones, the search criteria is always displayed at the top.

NOTE: Press Enter. Rather than clicking the **Search** button, you can press Enter to trigger the search / info zone refresh.

Description Bar

The Description Bar describes the search criteria that were used to build the Search Results Area.

Results Area

The Results Area displays the appropriate list of objects. For an info zone, this is the list of appropriate objects with any input filter criteria considered. The following is an example of the results for an info zone. For a query zone, this is the list that matches the search criteria defined in the Filter Area. The following is an example of the results set of an Info Zone.

Business Objects with Status Reason




	Business Object	Description	Statuses
📶 1	FI-BusinessFlagError	Business Flag Error	Discarded
📶 2	FI-CubeView	Cube View	Active, Inactive
📶 3	FI-GenericBISyncRequest	Generic BI Sync Request	Discarded, Cancel
📶 4	FI-MessageFromDevice	Message From Device	Canceled
📶 5	FI-MessageToDevice	Message To Device	Queued, Canceled, Processed
📶 6	FI-MigrDataSetExport	Migration Data Set Export	Canceled
📶 7	FI-MigrDataSetImport	Migration Data Set Import	Canceled, Completed



For a query zone, the columns that appear may differ depending on the search criteria. For example, if you search for a customer by their driver's license number, the resulting columns may be different than the ones displayed when you search by customer name.

The following points highlight details related to viewing and working with the result set:

- **Broadcast.** If the results set is configured to display a broadcast icon, , it means that at least one other zone on the portal is configured to display data related to the record selected in the results list. Clicking this icon will cause the other zone or zones to be built or refreshed to display related data.
- **Hypertext.** One or more columns in the zone may be configured as hypertext. Clicking the hypertext will result in navigating to another page to display data related to that column based on the configuration for that column.
- **Maximum height.** The maximum height of the report area is controlled by the zone's configuration. A vertical scroll will appear when more rows exist than fit into the height.
- **Maximum rows.** The maximum number of rows that can appear is controlled by the zone's configuration. Excess rows are truncated before display. Please note, you can configure the zone to [download more rows to Excel](#) than appear in the Results Area.
- **Control column sorting.** By default zones are designed to allow a user to sort the results list by clicking any column heading. Clicking once sorts ascending. Clicking again sorts descending. However, a zone may be configured to not allow this type of sorting.
- **Rearranging columns.** A zone may be configured to allow a user to rearrange columns and add or remove columns visible in the Result Area. Refer to [Column Configuration](#) for the details.

The type and variety of columns that appear in the Results Area is controlled by the zone's configuration.

Title Bar

The Title Bar contains the zone's description along with other icons and functions described in this section.

In the corner of the Title Bar, you'll see an indicator that tells you whether the zone is collapsed (e.g., minimized) or expanded. Each time you click the indicator, the zone toggles between expanded and collapsed. The following image shows a collapsed zone.




NOTE: Recommendation: For zones that are configured to show on portal preferences, set up [Portal Preferences](#) to collapse zones that aren't needed every time a portal is displayed. Rationale: The system doesn't build collapsed zones when a portal is displayed. Instead, collapsed zones are only built when a user expands them. Thus, indicating a zone is collapsed improves response times.

If the zone is configured with help text, a help icon appears to the right of the zone title. Click this icon to view the help text.

If a [saved search](#) is open for a zone, the name of the saved search is displayed after the zone title.

The following sections describe additional icons and tools visible on the title bar. Note that in addition to the information described in this section, additional actions that are specific to a zone may appear based on configuration. The actions may appear in the form of hypertext, an icon or a button.

Explorer Zone Menu

Click the Explorer Zone Menu icon  to list available tools. The following topics describe each tool.

Clear Filters

Click the **Clear Filters** link in the Explorer Zone menu to clear the filter values and zone data.

Export to Excel

Click the **Export to Excel** link in the Explorer Zone menu to download the zone's data to Excel. This link is hidden if the zone has been configured to not download rows to Excel.

Please note the following with respect to this feature:

- Clicking export performs the search because it is possible to configure a number of records for excel that differs from the number of records configured to display.
- Because the search is performed, you can request an export to Excel even without first searching on the zone. The filters populated (or not populated) at time of exporting are the ones used for the export. If the user did not search at all or did a search and then changed the filters and exported before searching again, the results in the export may not match what is visible on the zone.
- It is possible through zone configuration to mark columns that are visible to the user interface as suppressed in the export. Conversely, it's also possible to mark columns as suppressed on the user interface but included in Excel. As such, the columns visible in the export may not always match what is visible in the user interface. This applies when for zones that support [configurable columns](#). Columns removed from view through this technique are still included in the export if not marked to suppress.
- If the column results include any HTML, the HTML is stripped out before exporting to Excel.

Print

Click the **Print Zone** link in the Explorer Zone menu to print the zone.

Save Search Options

Query zones and info zones that include filter areas, allow a user to capture and save filter criteria so that the same search can be more quickly executed in the future. Users may save many different search options for the same zone. Users also have the option to nominate one search as the default so that this search is automatically loaded upon entering the zone. In addition, users may choose to set one or more searches as a Favorite, which causes the search to appear in the [Favorite Searches](#) zone in the dashboard. This allows a user to navigate to the portal for that zone and load the saved search in one click.

The Explorer Menu includes various options for supporting Saved Searches that are described below.

Note the following additional points about this functionality:

- **Click Search before Saving.** When a user wants to save a search, the search criteria should be entered and the **Search** button should be clicked prior to saving the search.

- **Criteria is saved, not results.** Although a user must click **Search** prior to saving the search, the system saves the criteria, but not the results. When returning to the query or info zone in the future and opening a given saved search, the results displayed will be the current records in the system that match the saved search criteria.
- **Multi-Query Zones.** When a user is on a multi-query zone, the searches are saved at the multi-query zone level so that all saved searches for the various search dropdowns for a given multi-query zone are visible in the Open Saved Searches menu option.
- **Functionality is Secured.** If for any reason an implementation wishes to restrict which users are allowed to save searches, the functionality is securable. Only users with **Inquire** access to the **F1_USRFAVSCH** application service will see these options in the explorer menu. This is the same application service that is used to provide access to the Favorite Searches dashboard zone.

Save As

The **Save As...** option in the Explorer Zone menu allows a user to save the current entered search criteria. This should be clicked after executing the search.

The user is prompted to provide a name for the search. This search is now available from the Open Saved Searches option. If desired, a user can also choose to make this search the Default or may choose to Add to Favorites.

This option is visible whether or not an existing saved search is loaded. When a saved search is loaded, this option allows a user to make changes to the search criteria and save a new search with that criteria.

Open Saved Search

The **Open Saved Search...** option in the Explorer Zone menu is visible if there are any saved searches for the zone that is currently visible. Select this menu entry to view the current list of saved searches for the zone. For multi-query zones, any saved search for any of the query zones that are part of the multi-query zone are visible. Choose the desired saved search. The criteria is updated with this information and the search is performed.

Save

The **Save** option in the Explorer Zone menu is visible if a Saved Search is already loaded. This option allows a user to save changes to the criteria of an existing saved search (overwriting the previously saved criteria).

Set As Default

The **Set As Default** option in the Explorer Zone menu is only visible if a saved search is loaded and it is not currently the default. Click this option to nominate this search to be the one that is loaded when navigating to this zone from the menu.

Remove As Default

The **Remove As Default** option in the Explorer Zone menu is only visible if a saved search is loaded and it is currently the default. Click this option to no longer load this search automatically when navigating to this zone from the menu.

Add To Favorites

The **Add To Favorites** option in the Explorer Zone menu is only visible if a saved search is loaded and it is not currently included in the favorite searches zone. Click this option to include this search in the [Favorite Searches](#) zone. Once a search is in the Favorite Searches zone, you can use the link to the search in the dashboard to navigate to the portal for this zone and load the saved search in one click.

NOTE: Not available for dependent zones There are some portals that include a zone that is not the main query or info zone on the portal, but is rather a zone providing additional information for a record that is displayed. If a zone like this

has filter options or configurable columns, a user is able to save searches and mark one as a default, but will not be able to add the saved search to favorites because the zone is dependent on other data on the page that is not captured for the individual zone.

NOTE: Dashboard Zone Preferences. Be sure that your [portal preferences](#) are configured to include the Favorite Searches dashboard zone.

Remove From Favorites


The **Remove From Favorites** option in the Explorer Zone menu is only visible if a saved search is loaded and it is currently included in the [Favorite Searches](#) zone. Click this option when you want to keep the saved search but you no longer want it visible in your Favorite Searches zone. (Deleting a saved search that is also marked as a favorite will automatically remove the search from the favorite searches zone.)

Delete

The **Delete** option in the Explorer Zone menu is only visible if a saved search is loaded. Click this option to remove the currently opened saved search from the list of saved searches.


Note that if the saved search had been configured as favorite, deleting the saved search remove it from the favorite searches zone.

Show Filters

Click the **Show Filters** icon  to show the [Filter Area](#). This button only appears if the zone has been configured to have filters.


NOTE: If the zone is configured to have filters and the drag and drop (column configuration) area, only one or the other is shown at any given time. It means that a user can either change the filter criteria and refresh the search results or change the column configuration and refresh the results.

Show Column Configuration Area

Click the **Column Configuration** icon  to show the [Column Configuration Area](#). This button only appears if the zone has been configured to allow columns to be dragged and dropped into the Results Area.

NOTE: If the zone is configured to have filters and the drag and drop (column configuration) area, only one or the other is shown at any given time. It means that a user can either change the filter criteria and refresh the search results or change the column configuration and refresh the results.

The Open Zone Maintenance Icon

Click the **Go To Zone** icon  to switch to the zone's [zone maintenance](#) page. On this page, you can change the parameters that control the zone's behavior.

NOTE: Debug mode only. This icon only appears when the system operates in [Debug](#) mode.

Configurable Columns

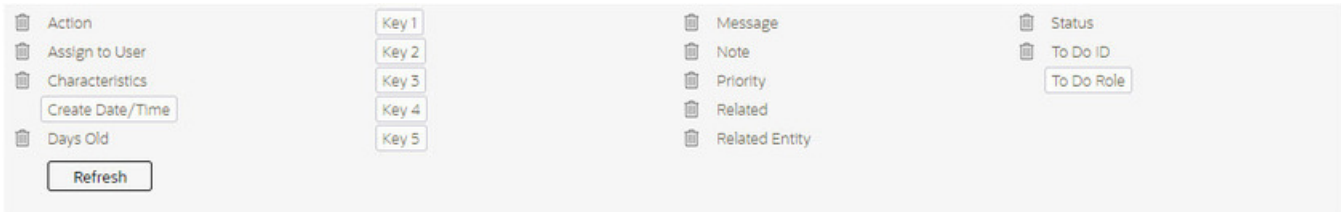
Info zones and query zones may be configured to allow users to rearrange columns and to remove or add columns to the output. This is also referred to as "drag and drop" of columns.

When a zone is configured to support this functionality, the [Show Column Configuration Area](#) icon is visible in the zone title area. Click this icon to expose the Column Configuration Area, if it is not visible and click the icon to hide the Column Configuration Area if desired.

If configurable columns are allowed, a user may rearrange the order of columns currently displayed in the Results Area by clicking the column heading and "dragging" it to the desired location.

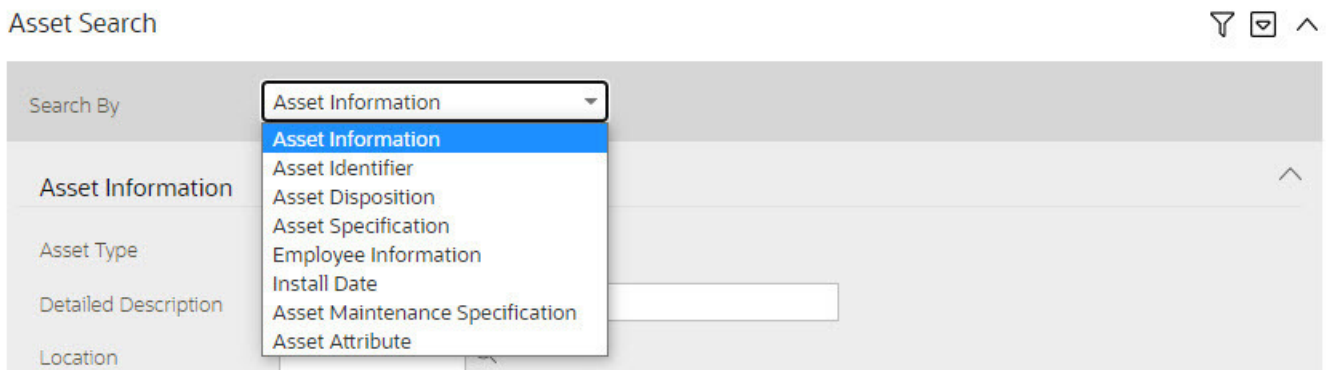
In addition, the column configuration area may display columns that are not currently in the Results Area. These are indicated by a cream background. A user can drag one of these columns to the results area and then click **Refresh** to rebuild the zone. To remove a column that is currently in the results area, click the trash can icon next to the column name in the column configuration area and then click **Refresh** to rebuild the zone.

The following is an example of how the Column Configuration Area looks:



Multi-Query Zones

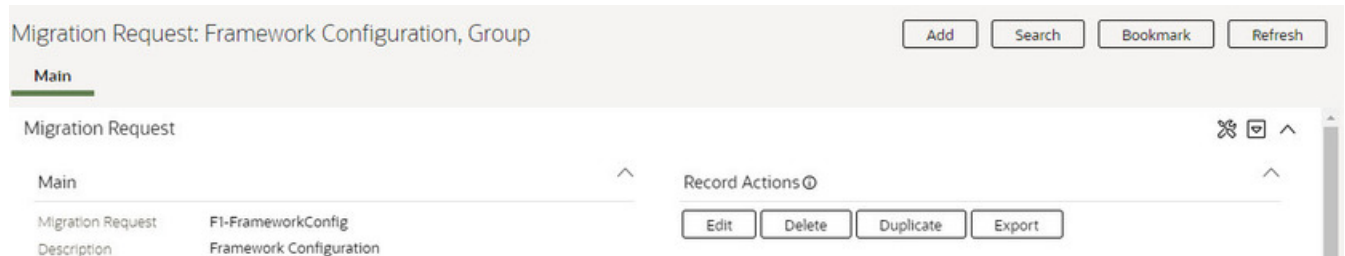
This type of zone combines individual Query Zones into a single query zone. A **Search By** dropdown, built from the zone descriptions of the individual Query Zones, allows you to select the desired query to execute. Selecting a specific Query Zone displays the filters of that query in the filter area. At this point the zone simply executes the selected Query Zone, as if it was executed on a standalone zone. By default, the first query zone option is selected.



Map Zones

A map zone displays information related to one or more objects. For objects that are maintained using portal-based pages, the standard display of the object is using a map zone. Map zones may be configured to display an explicit map or may dynamically display the map based on the configuration of the record being displayed.


Because the information displayed in a map zone is configured using HTML, virtually any format supported by HTML is possible. The following is an example of a typical map zone used to display information about an object.



A [tool tip](#) can be used to display additional help information for the details shown in a map zone. The tool tip may appear adjacent to the section header or adjacent to individual elements on a map.

NOTE: You can customize the help icon or disable map help using the [Custom Look and Feel](#) feature configuration.

Open Map Maintenance Icon

Click the **Go To Map** icon  to switch to the [UI map](#) maintenance page. On this page, you can view the HTML configuration for the zone. Note that for map zones that display a business object configured without a map, the elements displayed are automatically rendered. In this case, no icon is visible.

NOTE: **Debug mode only.** This icon only appears when the system operates in [Debug](#) mode.

Tabbed Display Zones

This type of zone combines individual [map zones](#) into a single zone, allowing for complex of information to be organized and displayed in different tabs.

Tab labels are built from the zone descriptions of the individual zones and become the zone's title. Selecting a specific tab displays the information managed by the underlying zone. By default, the first tab is selected.

Timeline Zone

A timeline zone is used to show events over time for multiple types of data. This type of zone is helpful for providing a consolidated view of separate events occurring in time that may in fact impact each other. For example, in the billing system, a timeline can show bills, payments and collection processes for the same time period.

The topics in this section describe the functionality available in timeline zones.

Timelines Zones Are Configured By Your Implementation Team

The product provides support for creating timeline zones. However, whether timeline zones are configured for your implementation and on which portals they may be visible will depend on your specific configuration. Refer to [Configuring Timeline Zones](#) for general information about configuring timeline zones. Refer to your specific edge application's documentation for details about what type of timeline information is provided out of the box.

You Can Move Through Time

You can click the controls at the top of a timeline zone to change the date-range of the zone's information. To reposition the timeline to a specific date, selected the desired month and year at the top of the zone and click the search arrow.

Controls at the top right of the zone allow you to navigate through time.

- To go back one year, click the double-left arrow.
- To go back one month, click the single-left arrow.
- To go to today, click the middle dot.
- To go forward one month, click the single-right arrow.
- To go forward one year, click the double-right arrow.

Timelines Can Have Many Lines

Depending on how a timeline zone is configured, it will typically have one or more "lines" that show when significant events have occurred. Each line represents a specific type of data. The data may be related to a specific maintenance object, for example, payments may be displayed in one line and bills in another. It may also be possible to create multiple lines for different types of records for the same maintenance object. For example, if your product provides an algorithm to display cases, it may support defining a different entry for each case type.

Each Line Shows Events

Each line on a timeline may contain zero or more events where each event shows the date when the event occurs. For example, the payment line in a timeline has a separate event for every payment received from the customer. Each line's description contains the number of events on the line.

CAUTION:

If a line has more events than can fit onto a timeline, the line will show the first "chunk" of events and a message will appear in the "more info area" explaining that some events have been truncated. If this happens and the truncated events are in a later period, you can reposition the timeline's base period to show the truncated events.

Hovering Over An Event Shows the Event Information

When you hover over an event's date icon, a pop-up widow appears showing details of the event. The following information may appear:

- The event date.
- The event "hover text". Typically the standard information string is shown for each event.
- The event's common "information string" appears. This information is hypertext to allow for easy access to the transaction on which the object is maintained.
- Additional information may appear. This is dependent on the particular timeline algorithm. Refer to each timeline algorithm's description to find out if additional information appears in the pop-up area under any conditions.
- If the algorithm has configured BPA scripts that can be executed to perform business processes on the object, the BPA script description appears prefixed with a "wizard's hat" icon.

Common Base Portals and Zones

The following sections describe common standards followed by the product for base query and maintenance dialogue.

All-In-One Portal

The all-in-one portal is used for low volume objects, where a small number of records are expected.

For example, many control tables use this style of portal. In this portal, rather than separate query and maintenance portals, a single portal allows a user to find a record and maintain the record. A zone listing all existing records is displayed when

first entering the portal. A user can perform actions on existing records or add new records from there. More details about the list zone are provided below.

Refer to [Maintenance Portal](#) for information about zones commonly found on maintenance portals.

The all-in-one portal will typically provide the ability to add a new record. This may be a button in the Page Action area or a hyperlink defined directly on the list zone's header area.

List Zone

This info zone appears when first entering the portal and displays a row for each record.

The following functions are commonly available in a list zone:

- A broadcast icon is included in each row. Clicking this icon opens other zones that contain more information about the adjacent record.
- An **Edit** icon may be included in each row. Clicking this icon launches the maintenance BPA script to modify the adjacent record.
- A **Duplicate** icon may be included in each row allowing the user to duplicate the adjacent record without having to first display the record.
- A **Delete** icon may be included in each row allowing the user to delete the adjacent record without having to first display the record.
- If the record has a simple BO lifecycle of **Active** and **Inactive**, the list zone may include a button to **Activate** or **Deactivate** the adjacent record.

Separate Query and Maintenance Portals

For high volume objects where more sophisticated search criteria is needed to find a particular record, separate portals are provided to search for the record and to maintain it.

The query portal contains a zone to search of the object that you wish to maintain. Additional information about the query portal's search zone is provided below. Clicking an appropriate record in the search results navigates to the object's stand-alone maintenance portal.

The query portal will typically provide the ability to add a new record. This may be a button in the Page Action area or a hyperlink defined directly on the list zone's header area.

Query Portal Search Zone

The query portal contains one zone used to search for the object.

The zone may be a standard query zone or may be a multi-query zone providing one or more ways to search for your record.

Once the search results are displayed, click on the hypertext for the desired record's information string to navigate to the stand-alone maintenance portal.

Maintenance Portal

A maintenance portal refers to a page that is used to view and maintain a specific record. These types of records are governed by a business object, which defines the user interface layout for both the display of the record and the add or update of the record.

An all-in-one portal is a type of maintenance portal where the maintenance zone is displayed once a record has been broadcast from the list zone. For the separate query and maintenance portal dialogue, the maintenance portal includes only zones relevant for the specific record selected.

Maintenance portals may have multiple tabs to display information relevant for the object. Often, if the object has a Log, the log is on a separate tab. (However, there may be some maintenance portals that include the log zone on the Main tab.)

The stand-alone maintenance portal is typically configured with the ability to navigate back to the query portal using either a **Search** button in the Page Actions area or using a **Go To Search** hyperlink in the main zone's header area. The stand-alone maintenance portal may also include the ability to add a new record using an Add button in the Page Actions area. (If not, a new record can be added by navigating through the menu or by navigating to the query portal where an Add hyperlink may be found.)

The following sections describe zones that are common to portals used to maintain a record. Note that additional zones may exist for specific all-in-one portals or stand-alone maintenance portals based on the particular functionality. Refer to the documentation for each portal to understand the specific zones that are provided.

Object Display Zone

This is a zone that displays the object being maintained.

An object being displayed may or may not allow for actions or updates to be made. Typically there is a Record Actions section in the display area that presents the valid actions. Depending on the functionality, the actions may prompt the users to input values required for that action.

Log Zone

If the maintenance object for the record includes a status, it typically has a log that records important events in its lifecycle.

This log is shown in an info zone on the portal, either on the Main tab or on a separate Log tab. A link appears in the zone's title bar to allow the user to add a new manual log entry.

Process Flow

This portal is used to guide a user through the steps needed to complete a process flow. If the process flow type is configured to keep a record of a completed process flow then this portal may also be used to display the details of such record.

Refer to [Understanding Process Flows](#) for more information.

Each type of process flow is assumed to be configured with designated custom menu options that allows you to launch a new process flow of that type as well as review saved off and completed records. When reviewing existing process flow records, you are brought to a query portal with options for searching for a specific record. You may **Resume** a saved off but not yet finalized process flow or review a complete one.

The **Process Flow** zone on the portal's **Main** tab page assists the user in completing the process flow as well as displays the information captured on a completed process flow.

Fixed Pages

There are some pages in the system that are built using a fixed page metaphor. All user interfaces implemented in this way function in a similar manner. This section describes functionality to these types of pages.

NOTE: Not all pages in the application use this metaphor. The preferred metaphor for pages in the system is to use portal-based pages. For more information see [Common Base Portals and Zones](#).

Maintenance Pages

Maintenance pages are used to maintain (add, change, etc.) the information in the system using a fixed page metaphor. This section focuses on functionality common to all maintenance pages using this metaphor.

The Unique Identifier Area

Every object in the system must have a unique identifier. For some objects, typically administrative or control table data, the unique identifier is defined by the user creating the record. Other objects, typically master or transaction entities, the unique identifier is assigned by the system after the record is saved.

Where the identifier is displayed depends upon whether the ID is user-defined or system generated.

Pages displaying objects with a user defined identifier (such as To Do Type) typically display the identifier on the left along with a search button. For master and transaction tables, (such as To Do Entry) the unique identifier is typically found in the upper right corner of the screen along with a search button and is labeled as the entity's ID. In this case, if there is an information string associated with the object, it is displayed on the left.

If the object being maintained has an associated context menu, a [Context Menu Button](#) is displayed appear.

Confirmation & Error Messages

Whenever a database modification action is requested, the system first validates the data to be put on the database.

First, the system validates the information. If there are errors, the first error encountered is displayed in a window. In addition to the error message, the error window also contains the error number and the identities of the programs that detected the error. If you don't understand an error and you have to call product support, please supply them with this information.

If there are no errors, the system is modified.

Warning If Unsaved Changes Exist

If you make any changes to an object and you don't commit these changes to the database, the following warning appears: "You have unsaved changes that will be lost if you continue. Press OK to continue anyway."

If you want to commit the changes, click **Cancel**. You will then return to the page and can make further changes or press Save. If you want to discard your changes and transfer to the new page, click **OK**.

CAUTION: This warning is not displayed if you are working on a *new* object on a fixed page and you navigate to a different page. Information about new objects is preserved in memory in anticipation of your return to the page. For example, if you are in the middle of adding a new user and you want to quickly reference information on a different page, you will not receive the above warning because your information will be saved. If you use the back button to return to the user page, all of the information you were adding will be present and you can enter the rest of the new user's data.

Query Pages

Query pages are used to display record sets in the system. Most query pages function in a similar manner. This section focuses on functionality common to all of the system's query pages.

The Search Area

The upper right corner of the screen is almost always the identifier of the object associated with the search. In Oracle Utilities Customer Care and Billing for example, the unique identifier on the Account Financial History page is the Account ID.

To the right of the unique identifier is a [Search button](#). This button is used to search for a different object than the one whose data is being displayed in the page.

If the object being displayed has an associated context menu, a [Context Menu Button](#) is displayed.

User Preferences

The Preferences (or My Preferences) transaction and the User Maintenance transactions are the same except that with My Preferences, only the following items can be modified:

- Main Tab
 - Language
 - Display Profile
 - Time Zone
 - Email Address
 - Dashboard Width
 - Dashboard Location
 - Dashboard State
 - Home Page
 - To Do Summary Age Bar Parameters
- Bookmarks
 - All information can be modified
- Portal Preferences
 - All information can be modified
- Favorite Links
 - All information can be modified
- Favorite Scripts
 - All information can be modified

User - Main

This page lets you view or define user information and access settings.

To open the page, use one of the following navigation options:

- The **Admin > Security > User > Search** menu item.
- The **Menu > My Preferences** menu item.
- The **User Menu > Preferences** menu item in the application toolbar.

Note that some fields are only modifiable when the page is opened from the from the **Admin** menu.

Description of Page

Basic user identification information is viewed or set in the **First Name**, **Last Name**, **User ID**, and **Login ID** fields, all of which are modifiable only when the page is opened from the **Admin** menu. User and logon IDs must be unique. The Login ID is the ID that the user uses to log in to the system. The User ID is the ID that is used throughout the application when users are assigned to tasks or when the user ID is stamped on records when logging add or update actions.

Language defines the language in which the user's screen prompts and messages appear.

Set **User Enable** to allow the user to access the system.

NOTE: This field, available only when the page is accessed through the **Admin** menu, provides special field-level security. A user can change the field only if the **Enable/Disable** access mode is set.

Display Profile defines the [display options](#) for the user.

The next group of fields is used to view or set user preferences. For example, you can set up [portal preferences](#) on a "template user" and indicate that specified users will inherit their portal preferences from the template. Doing so provides consistency in the application of account information and dashboard zones. This fields are modifiable only when the page is accessed through the **Admin** menu.

- Set **User Type** to **Template User** if the user's preferences are to match those of other users.
- If a user's [portal preferences](#) are to be inherited from a "template user", use the **Portals Profile User** to define the user ID of the template user. If applied, the specified user will inherit portal preferences from the **Portals Profile User** and will be unable to customize those preferences.
- If a user's [favorite links](#) are to be inherited from a "template user", use the **Favorites Profile User** to define the user ID of the template user. If applied, the specified user will inherit favorites preferences from the **Favorites Profile User** and will be unable to customize those preferences.

The **Time Zone** associated with the user may be defined. Implementations may use this information for features and functions where the user's time zone impacts display of data. Refer to [Designing Time Zones](#) for more information about the use of time zones.

Email Address captures the user's email address.

Dashboard Width defines the number of pixels to use for the user's [dashboard](#) area. A setting of 250 is recommended as a minimum. Depending on the width of a user's monitor, a higher setting (such as 250) will display more information without compromising the space on the main area of the application. Setting the value to 0 will suppress the dashboard. The width setting is updated in background to retain the value based on dynamic sizing.

Dashboard Location governs whether the dashboard appears to the left or right of the screen. If no value is specified, the default position of the dashboard is left.

Dashboard State indicates whether the dashboard is open or closed. The state is updated in background to retain the value based on dynamic opening or closing of the dashboard.

Home Page defines the page that appears when the user starts a session or clicks the [Home](#) link.

The To Do entry fields control how To Do entries are aged in respect of the "age bars" that appear on [To Do Summary](#), [To Do Supervisor Summary](#) and the [To Do Summary Dashboard Zone](#).

- Use **Lower Age Limit for Yellow Bar** to define the number of days old a To Do entry must be to be considered "yellow" in the age bar. To Do entries below this limit are considered "green" in the age bar.
- Use **Upper Age Limit for Yellow Bar** is the high limit for the "yellow" bar. To Do entries older than this are considered "red" in the age bar.

The **User Group** collection defines the group(s) with which the user is associated and the date (if any) when the user's association expires. This feature is modifiable only when the page is accessed through the **Admin** menu. For more information, see [The Big Picture Of Application Security](#). Assignments of users in a user group can also be established or maintained through the [User Group - Users](#) page.

NOTE: The ability to maintain the user groups on this page requires special field-level security. A user can change the field only if the **System Administration** access mode is set.

If your implementation has configured [Object Erasure](#) for the User maintenance object, the **Object Erasure Schedule** dashboard zone is visible displaying the erasure status for this user record, if applicable. Refer to [Erasing User Information by Obfuscation](#) for more information on object erasure for User records.

User - To Do Roles

Open the User page and then navigate to the **To Do Roles** tab to define the To Do roles assigned to the user.

The information on this page cannot be modified if accessed through **My Preferences**.

FASTPATH: A user's To Do roles control the types of **To Do** entries the user can access. For additional information on To Do roles, see [To Do Entries Reference A Role](#).

Description of Page

The grid contains the user's To Do roles. To modify a To Do role linked to the user, simply move to a field and change its value. To remove a To Do role, click the - (minus) button. To add a new To Do role, click the + (plus) button and enter the role.

Where Used

Every **To Do** entry references a role. A To Do role has one or more users (and a user may belong to many To Do roles). Users who are part of the To Do role assigned to a To Do entry may work on the To Do entry. For more information on this feature, see [The Big Picture Of To Do Lists](#).

User - Access Security

Open the User page and then navigate to the **Access Security** tab to define a user's security rights. You cannot modify this page if you accessed it through **My Preferences**.

FASTPATH: Refer to [The Big Picture of Application Security](#) for more information about data access roles and access groups.

Description of Page

Use the **Default Access Group** to define the access group that is defaulted on new records added by this user that are subject to [row security](#).

The scroll area contains the **Data Access Roles** to which this user belongs. A user's data access roles play a part in determining the accounts whose data they can access.

To add additional data access roles to this user, click the + button and specify the following:

Enter the **Data Access Role**. Keep in mind that when you add a **Data Access Role** to a **User**, you are granting this user access to all of the accounts 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: You can also use [Data Access Role - Main](#) to maintain a data access role's users.

The tree highlights the **Access Groups** to which the above **Data Access Roles** provide access.

User - Bookmarks

This page is used to manage a user's bookmarks that are created when a user clicks the [Bookmark](#) button and saves a bookmark. From this page the user can delete a bookmark, change the name of the bookmark or change the sort sequence.

Open the User page and then navigate to the **Bookmarks** tab to modify or delete a bookmark.

Description of Page

Sequence is the sort sequence associated with the bookmark. Bookmarks with the same sequence are then sorted alphabetically.

Name is the user-defined name assigned to the bookmark. This is the text that is visible in the [Bookmarks](#) dashboard zone.

NOTE: Favorite Links vs. Bookmarks. Favorite links are configured through the user page and define pages that a user can navigate to. They do not define context (although some pages may use information currently in global context when navigating). Some users may not be allowed to configure their own favorite links and would see the favorite links associated with a profile user. Bookmarks are created using a bookmark button available on each page. The bookmark saves navigation information, context information and also captures some information about the state of a page. Bookmarks are not governed by any user profile configuration.

User - Portal Preferences

The base product contains several [portals](#) that allow users to customize them via portal preferences. Portal preferences allow users to control:

- Which zones appear on the portal
- The order in which the zones appear
- Whether the zones should be "collapsed" (i.e., minimized) when the portal opens.

NOTE: You may not be able to change your portal preferences. If a note appears immediately before the list of portals, a system administrator has configured your user ID to reference a **Portals Profile User** (this is defined on the **Main** tab). Preferences set in this way cannot be modified. System administrators do this in order to enforce a common look-and-feel throughout the user community.

Open the User page and then navigate to the **Portal Preferences** tab to modify a user's preferences.

Description of Page

The [accordion](#) contains a row for every portal configured to show on user preferences and to which you have access. To change how a portal's zones appear, expand the respective row and change the elements accordingly. The remainder of this section describes how you can configure how a portal's zones appear.

Zone lists all of the [zones](#) in the portal.

Place a check under **Display** if the zone should be displayed on the portal.

Place a check under **Initially Collapsed** if the zone should be minimized when the portal is initially opened. Setting zones as initially collapsed is a good idea to save space and to help pages to load more quickly. Refer to [Zones May Appear Collapsed When A Page Opens](#) for more information.

CAUTION: Recommendation. It is recommended that you set your preferences to collapse zones that you don't use often. This is because the system doesn't perform the processing necessary to build collapsed zones until you ask to expand a zone. Opening a page with zones collapsed accelerates the response times of portal.

Use **Sequence** to control the position of the zone in respect of the other zones. This field is disabled if **Display** is not checked. Duplicate **Sequence** numbers are allowed.

For zones displaying data that is changing often, use **Refresh Seconds** to define in seconds how often the zone is refreshed. The minimum valid value is 15. The maximum valid value is 3600 (1 hour). A value of 0 indicates no automatic refresh.

Security Access indicates whether the user has access rights to a zone. Refer to [Granting Access To Zones](#) for more information.

User - Favorite Links

This page is used to define a user's favorite transactions and URLs. Each such link appears as an entry in the [Favorite Links Zone](#) in the [Dashboard](#) area.

NOTE: A hot key can be used to execute a favorite link. Rather than clicking a button in the favorite links zone, you can invoke your first nine favorite links by pressing `Ctrl +` a number (where the number corresponds with the relative position of the transaction in your list of favorites). Note that the numeric keypad on your keyboard cannot be used to implement this function.

Open the User page and then navigate to the **Favorite Links** tab to modify a user's favorite links.

Description of Page

NOTE: You may not be able to change your favorite links. If a **Note** appears beneath your **User ID**, your user id has been setup to reference a specific **Favorites Profile User** as defined on the **Main** tab. If a system administrator has setup your user ID in this way, your preferences may not be changed. System administrators may apply this feature in order to enforce a common look-and-feel throughout the user community.

Each row references a [Navigation Option](#). The navigation options supplied with the base package correspond with the system's menu items. This means you can setup a favorite link to open any transaction in the system (in either **add** or **update** mode).

NOTE: Favorite links can point to other places. The navigation options supplied with the base package always open a transaction's **Main** tab. If you find yourself continually accessing a different tab page, you can have your implementation team create a new navigation option that references the desired tab page. Once this navigation option is set up, you can reference it as one of your favorite links. In addition, it's also possible for your implementation team to setup navigation options that open a URL outside of the system. For example, your implementation team could setup a navigation option to a credit rating company's website. Once this navigation option is set up, you can reference it as one of your favorite links.

Use a **Sequence** to control the order in which the navigation option appears in the [Favorite Links Zone](#).

Security Access indicates if you have been given security rights to the respective transaction. Refer to [The Big Picture Of Application Security](#) for more information.

NOTE: Favorite Links vs. Bookmarks. Favorite links are configured through the user page and define pages that a user can navigate to. They do not define context (although some pages may use information currently in global context when navigating). Some users may not be allowed to configure their own favorite links and would see the favorite links associated with a profile user. Bookmarks are created using a bookmark button available on each page. The bookmark saves navigation information, context information and also captures some information about the state of a page. Bookmarks are not governed by any user profile configuration.

User - Favorite Scripts

This page is used to define a user's favorite scripts. Each script is displayed in the **Favorite Scripts** zone on the **Dashboard** portal where it can be used to [initiate a script](#).

NOTE: A hot key can be used to execute a favorite script. Rather than selecting a script from the business process assistant, you can invoke your first nine favorite scripts by pressing `Ctrl + Shift + a number` (where the number corresponds with the relative position of the script in your list of favorites). Note that the numeric keypad on your keyboard cannot be used to implement this function.

Open the User page and then navigate to the **Favorite Scripts** tab to modify a user's Favorite Scripts.

Description of Page

NOTE: You may not be able to change your favorite scripts. If a **Note** is displayed on top of the grid, your user id has been setup to reference a specific **Favorites Profile User** as defined on the **Main** tab. If a system administrator has setup your user ID in this way, your preferences may not be changed. System administrators may apply this feature in order to enforce a common look-and-feel throughout the user community.

Each row in the grid contains one of your favorite **Scripts**.

Use the **Sort Sequence** to control the order in which the scripts appear in the [business process assistant menu](#).

Security Access indicates whether you have access to the script's application service. Refer to [The Big Picture Of Application Security](#) for more information.

User - Characteristics

Select **Admin > Security > User > Search** and navigate to the **Characteristics** tab to define characteristic values for the user. You cannot modify this page if you accessed it through My Preferences.

Description of Page

Use the characteristics grid to capture additional information about the user.

You can only choose **Characteristic Types** defined as permissible for the user record. Refer to [Setting Up Characteristic Types & Their Values](#) for more information.

Enter a valid **Characteristic Value** for the selected characteristic 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.

User - Miscellaneous

Select **Admin > Security > User > Search** and navigate to the **Miscellaneous** tab to view additional information associated with a user.

User Id, First Name and Last Name are displayed. Depending on your implementation, additional zones may appear on the User Portal

If additional zones appear, use the embedded help for information about the fields managed on the zone.

Initiating Scripts

Your implementation team can set up business process assistant (BPA) [scripts](#) to walk you through business processes. Once scripts have been set up, you can initiate a script using any of the following methods:

- You can define a list of your "favorite scripts" on your [user preferences](#). You can then invoke any of the scripts from the Favorite Scripts zone on the dashboard portal. You can also invoke any of your first 9 favorites by pressing `Ctrl + Shift + a number` (where the number corresponds with the relative position of the script in your list of favorites). For example, if you press `Ctrl+Shift+2` and you've setup your "favorite scripts" to be **Start Autopay** and **Stop sending marketing information**; the **Stop receiving marketing information** script is invoked. Note that the numeric keypad on your keyboard cannot be used to implement this function.
- You can configure a menu item to launch a script by referencing a [navigation option](#) that is a **script** navigation option type.
- The system creates [To Do entries](#) to highlight tasks that require attention (e.g., bills in errors, accounts without bill cycles, etc.). 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. This script might prompt the user to first correct the customer's default mailing address and then re-complete the bill. Refer to [Executing A Script When A To Do Is Launched](#) for more information.
- You can indicate that a script should be [launched upon entering the system](#). This functionality enables an external system to launch the system and immediately launch a script.
- The following [hot keys](#) cause an interactive menu is displayed:
 - `Ctrl+Alt+Q`
 - `Ctrl+Shift+S`

Description	Eligible
Account Sync on CS Req Account page	Eligible
Accounting Calendar Maintenance	Eligible
Activity Asset Completion Action	Eligible
Activity Resource Requirement - Add User Log Entry	Eligible
Add Crew Type Resources	Eligible

The menu allows you to search for a script using any word in its description. Note, if you've turned on **Search only Eligible Scripts**, your search is limited to [eligible scripts](#). Note, the system displays an indication if each script in the grid is **Eligible** or **Ineligible** in this section.

You can initiate a script from this area by clicking on it.

When a script is initiated, this menu closes and the [object display area](#) shifts down to make room for the script area.

NOTE: Securing scripts. Only scripts the user is allowed to execute are displayed. Refer to [Securing Script Execution](#) for more information.

NOTE: Customize the script area. Using the [Height](#) step type, an individual script may be configured to use more or less of the script area than the standard script area. It is also possible to configure a script to hide the script area altogether.

The information in the script area takes you through the steps in the selected business process (note, your implementation team controls the information that appears in the script area).

The following points highlight useful information common to all scripts:

- **Return focus to script area.** Often during a script the focus moves from the script area to another section of the page. For example, you might be required to fill in data for a new record being added. When you are ready to return to the script area, rather than using the mouse to return to the area, you can press `Ctrl+Shift+F`. The focus returns you to either the first available input field or to a "default" button if no input field is available.
- **The Continue button.** Many steps in a script ask you to click the **Continue** button when you're ready to restart a script. For example, a step may ask you to confirm a customer's automatic payment information and then click **Continue** when you're ready for the script to restart. Rather than using the mouse to click this button, you can press `Ctrl+Shift+C`.
- **Selected button changes color.** When you click a button in the script area, the button selected changes color so that you can easily see which option you selected.

FASTPATH: Refer to [The Big Picture Of Scripts](#) for more information about scripts.

NOTE: You can abort a script at any time by clicking the red X button in the upper right corner of the script area (or by pressing `Ctrl+Shift+X`).

Shortcut Key Summary

The following table summarizes the keyboard shortcuts (hot keys) available in the system:

NOTE: It is possible that your specific product supports additional shortcut keys. Refer to your product documentation for more information.

NOTE: It is possible that some of the following system accelerator keys may duplicate shortcuts, or hot keys, that are predefined within your browser. If this is the case, the browser's functionality commonly takes precedence.

NOTE: It is possible that your implementation uses one of the accelerator keys listed below for a different purpose. For example, in some languages, a shortcut key may be used for a special character in the alphabet not included on all keyboards. It is possible to turn off any of the `Alt+letter` shortcuts and `Ctrl+Alt+letter` shortcuts listed below using a properties file entry. The property is `ouaf.shortcut.ignore.altKeys=` where one or more letters may be entered separated by a comma. When setting this property, the system automatically enables a 'potential alternate key' for certain functions (see the table below). Refer to the *Server Administration Guide* for more information about the property.

Accelerator Key	Potential Alternate Key (see Note above)	Function
<code>Ctrl+Alt+A</code>		Opens the Admin Menu .
<code>Alt+B</code>		Returns to the previous page on which you were working. This is the equivalent of clicking the Back arrow.

Accelerator Key	Potential Alternate Key (see Note above)	Function
Ctrl+Alt+B		Launches the Bookmark dialog. This is the equivalent of clicking the Bookmark button.
Alt+C, Ctrl+Alt+C, Alt+Shift+C		Clears the currently displayed object from the page. This is the equivalent of clicking the Clear button.
Ctrl+Shift+C		If the current step in the Script area has a Continue button, this shortcut is equivalent to clicking Continue .
Ctrl+Alt+D, Alt+D		Creates a duplicate of the object being displayed. Equivalent to clicking the Duplicate button.
Ctrl+Alt+F		Positions the user's cursor to the Search input field.
Ctrl+Shift+F		If the Script area is open and you are working on a different area of the page, this key returns focus to the script area. Focus returns to either the first available input field or to a "default" button if no input field is available.
Alt+G		Returns to the most recent page that was displayed when the Back arrow was clicked. Equivalent to clicking the Forward arrow.
Ctrl+Alt+H		Opens a dropdown list of recently-visited pages, the equivalent of clicking the History icon.
Alt+J		Minimizes the dashboard . It also expands the dashboard, if it was minimized.
Alt+K		Transfers you to To Do Entry - Main for the current item in the To Do list. Pressing Alt+K while on this page takes you to the first To Do entry on the list.
Ctrl+Alt+L		Logs the user out of the system. This is equivalent to clicking the Logout menu item.
Ctrl+Alt+M		Opens the Menu .
Alt+N	Alt+Shift+N	Displays the next entry in the search results. Equivalent to clicking Next Item .
Alt+O	Alt+Shift+O	Navigates to the user's home page. Equivalent to clicking the Home icon.
Ctrl+Alt+P		Opens your User Preferences page. Equivalent to clicking the Preferences menu item.
Alt+P		Displays the previous entry in the search results. Equivalent to clicking Previous Item .
Ctrl+Alt+Q		Opens the business process assistant , where you can select a script that walks you through a business process.
Alt+R Ctrl+Alt+R		Refreshes the page with the last saved version of the object being displayed in the page. Equivalent to clicking the Refresh button.
Alt+S	Alt+Shift+S	Saves any changes that you've made on the database. Equivalent to clicking the Save button.
Ctrl+Shift+S		Opens the business process assistant , where you can select a script that walks you through a business process.
Alt+X	Alt+Shift+X	Opens the To Do Summary .
Ctrl+Shift+X, Ctrl+Shift+Z		If the Script area is open, equivalent to clicking the Close button in the script area. If a pop-up window is open, this closes the window.

Accelerator Key	Potential Alternate Key (see Note above)	Function
Alt+Y		Displays the previous item in the To Do list. Equivalent to clicking the Previous button in the Current To Do zone.
Alt+Z	Alt+Shift+Z	Displays the next item in the To Do list. Equivalent to clicking the Next button in the Current To Do zone.
Alt+Insert		Inserts a new row into an editable grid if the insertion point is in the grid. Equivalent to clicking the Insert button.
Alt+Delete		Removes a row from an editable grid if the insertion point is in the grid. Equivalent to clicking the Delete button.
Alt+[Navigates to the next main page component area of the application. Refer to Page Components for the list of the main page components. Using Shift+Alt+[reverses the sequence of navigation.
Alt+]		Navigates to the next zone . When you are on a portal with zones, this key brings you to the next zone in the portal. If you are on the last zone in the portal, the key brings you to the first zone on the dashboard, if visible. Keying through all the zones on the dashboard brings you to the first zone on the portal displayed in the main area. If the page displayed is a fixed page rather than a portal, this key will only navigate within the dashboard. Using Shift+Alt+] brings you to the previous zone.
Alt+F1		Invokes online Help for the application. Equivalent to clicking the Help entry in the Help menu.
Alt+1 ...Alt+9, Alt+0		These keys are only applicable when a page / portal with multiple tabs is open. The shortcuts open the tab page whose relative position corresponds with the shortcut number. For example, if you press Alt+3 on a page with multiple tabs, the third tab is displayed. Alt+0 navigates to the 10th tab. If there are more than ten tabs on a page, you must use the mouse or F2/Shift+F2 to display additional tabs. Note that you cannot use the numeric keypad for these shortcuts.
Ctrl+1 ...Ctrl+9		Invokes the favorite link whose relative position corresponds with the shortcut number. For example, if you press Ctrl+3, the transaction for your third "favorite link" is invoked. Note that you cannot use the numeric keypad for these shortcuts.
Ctrl+Shift+1 ... Ctrl+Shift+9		Invokes the favorite script whose relative position corresponds with the shortcut number. For example, pressing Ctrl+Shift+2 selects the second of your "favorite scripts". The numeric keypad cannot be used for these shortcuts.
Alt+left arrow		Populates a scroll area with the previous entry. Equivalent to clicking the scroll control  .
Alt+right arrow		Populates a scroll area with the next entry. Equivalent to clicking the scroll control  .
Enter when in a field with an adjacent Search button		Invokes the search. Equivalent to clicking the Search Button .

Accelerator Key	Potential Alternate Key (see Note above) Function
Enter when a row in a search result grid is highlighted	Selects the item and closes the search. Note that after selecting an item from the search results, you can press Alt+N or Alt+P to scroll up or down through other items in the result grid without returning to the search results.
Enter when a menu item is highlighted	Selects the menu item.
Enter when the insertion point is within a tree node	Equivalent to clicking on the item in the tree: <ul style="list-style-type: none"> • If the insertion point is on a node image, the node is either expanded or collapsed, dependant on its current state. • If the insertion point is on a context menu, the context menu is opened and normal menu processing applies. • If the insertion point is on a "link" element, normal navigation processing occurs.
Tab when a row in a search result is highlighted	Highlights the next row.
Shift + Tab when a row in a search result is highlighted	Highlights the previous row.
Space when the insertion point is on a button	Equivalent to pressing the button.
Space when the insertion point is in a date/time field	Opens the date/time selection window.
Space when the insertion point is in a check box	Toggles the check box between checked and unchecked.
Page Up while in a list grid	Scrolls the contents of the list grid up approximately one page.
Page Down while in a list grid	Scrolls the contents of the list grid down approximately one page.
Up or Down Arrow when in a dropdown list	Moves to the next or previous value in the list. You can also press a letter key to locate the next value starting with that letter.
Up or Down Arrow keys when in search results	Highlights the next/previous row in search results.
Up, Down, Left or Right Arrow when using a menu	Highlights other menu items.
Esc when using a menu	Closes the menu.
F2	Opens the next tab page when viewing a page / portal with multiple tabs.
Shift+F2	Opens the previous tab page when viewing a page / portal with multiple tabs.

Chapter 3

WAM Reports

About WAM Reports

WAM reports include maintenance reports related to work orders, activities, and permits as well as purchasing reports related to requisitions, orders, inventory, and materials.

Oracle Utilities Work and Asset Management reports are implemented using Oracle Business Intelligence Publisher (BI Publisher) and are administered via Oracle Business Intelligence Publisher and the Report functionality of the Oracle Utilities Application Framework.

Refer to the [Oracle Business Intelligence Publisher documentation](#) for more information about installing Oracle Business Intelligence Publisher and [creating and defining reports](#). For additional details on how reports are defined and administered see [Reporting and Monitoring Tools](#) in the Oracle Utilities Application Framework *Administrative User Guide*.

Running Oracle Utilities Work and Asset Management Reports

Users can run a report directly from a record. Refer to [About WAM Reports](#) for additional information.

Navigate to the record and click **Print** from the **Record Actions** zone. The reports provided out-of-the-box are:

- **Activity Report:** This report generates an activity report for a selected activity type. The report details characteristics of the activity, such as the associated asset, work priority, and location information as well as activity dependencies and resources.
- **Crew Shift Daily Report:** This report generates a daily schedule report for each shift associated with a particular crew. Report displays basic details about the shift, such as the schedule date, time range, and shift status, as well as a detailed schedule of daily activities for the shift.
- **Crew Shift Weekly Report:** This report generates a weekly schedule plan for each shift associated with a particular crew. Report displays basic details about the shift, such as the scheduled start and end date and shift status, as well as a detailed schedule of weekly activities for the shift.

- **External Permit Report:** This report generates an external permit report that includes such criteria as the permit's effective and expiration dates, requestor, current permit holder, work location, and work window. This report also includes information about fees and checklist details.
- **Internal Permit Report:** This report generates an internal permit report that includes such criteria as the permit's effective and expiration dates, requestor, current permit holder, work location, and work window. This report also provides details for asset and location information, labor, equipment, materials, and checklist details.
- **Material Request Report:** This report generates a material request report depicting details for the selected material request header by status and specific storeroom. The report also includes additional information involved with the request, such as required by date, deliver to location, stock item and quantity details, as well as primary bin information and other selected criteria.
- **Physical Inventory Report:** This report generates a physical inventory report listing the stock items for a particular storeroom. Report details include the bin, stock item code, and unit price of each item in the storeroom.
- **Purchase Order Report:** This report generates a purchase order report by buyer for a particular vendor. The report includes the PO number, blanket contract number, and vendor details as well as the purchase order line and purchase order line manufacturer details.
- **Receiving Report:** This report generates a receiving report by transaction number. The report details include associated PO and BC numbers, status of the PO, vendor and manufacturer details, and stock item and receipt line details.
- **Requisition Report:** This report generates a requisition report by requestor for a particular vendor. The report includes requesting line details, line details for accounting, work orders and activities, manufacturers, and notes.
- **Shipping Memo Report:** This report generates a shipping memo report by memo number or shipping date. The report includes ship to details, tracking information, and the purpose for the items being shipped. Also included is a table describing details about the items being shipped.
- **Work Order Report:** This report generates a work order report by work order number or activity. The report details characteristics of the work order, such as the associated assets and locations, work priority, as well as activity dependencies, resources and labor requirements.

NOTE: Account Security. The sample reports provided with the product do NOT incorporate account security. If a user has been given security to view the report, then all the data in the report is available for viewing.

NOTE: Also note that a limited version of **Business Intelligence** is provided with the application so users can access the sample reports. Customized reports will require purchase of the more robust **Business Intelligence** application.

Chapter 4

Search Features

Using Unified Search

The Unified Search feature allows you use the toolbar search field to search for business entities as well as search for menu items. See [Unified Search](#) for more information about the general use of the unified search field.

In Oracle Utilities Work and Asset Management and Oracle Utilities Work and Asset Cloud Service, the Unified Search field allows searching for Assets or Asset Locations but only one or the other, depending on configuration.

The following filters can be used to indicate your specific search criteria and are available to use when searching for Assets. When using the Asset Location Unified Search, only the filters indicated by an asterisk (*) can be used.

Filter	Filter Rule
Badge Number	Minimum of 5 characters excluding whitespaces.
Serial Number	Minimum of 5 characters excluding whitespaces.
Tag Number*	Minimum of 5 characters excluding whitespaces.
Building*	Minimum of 2 characters excluding whitespaces.
Address*	One or more numeric characters followed by space and an alphanumeric character.
Description*	Minimum of 5 characters excluding whitespaces.

For example, to search for serial number 678912, the system would look at the filter rule (all except address) and return all “likeable” search results until it reaches 20 results or finds an exact match.

To search explicitly using a specific filter, precede the filter value with one of the following predefined hints:

Filter	Hint	Example
Badge Number	bn:	bn:12345
Serial Number	sn:	sn:678912
Tag Number*	tn:	tn:56567
Building*	bl:	bl:B45
Address*	ad:	ad:1401 Flower

Filter	Hint	Example
Description*	ds:	ds:Smith Dr

NOTE: Hints can either be all lowercase or all uppercase. For example, bn:12345 and BN:12345.

Examples:

- To search for an asset whose badge number is 12345, you could enter "bn:12345" (hint) or "12345" (no hint)
- To search for an asset whose serial number is 678912, you could enter "sn:678912" (hint) or "678912" (no hint)
- To search for an asset whose tag number is 56567, you could enter "tn:56567" (hint) or "56567" (no hint)
- To search for a building whose number is B45, you could enter "bl:B45" (hint) or "B45" (no hint)
- To search for an address located at 1401 Flower Street, you could enter "ad:1401 Flower" (hint) or "1401 Flower" (no hint)
- To search for a description whose description is smith at vine, you could enter "ds:Smith Dr" (hint) or "Smith Dr" (no hint)

Clicking the **Advanced Search** link opens the Query Search page for your unified search, either Asset or Asset location.

For information about configuring the Unified Search for your implementation, refer to [Understanding Unified Search](#) in the *Administrative User Guide*.

Search Results

Each result in the search result list may include any of the following:

- Asset Header Info, Asset ID, Status, and Badge number (highlighted when using the Badge filter) or "No match found"
- Asset Header Info, Badge Number, Asset ID, Status, and Serial Number (highlighted when using the Serial Number filter) or "No match found"
- Asset Header Info, Asset ID, Status, and Tag Number (highlighted when using the Tag Number filter) or "No match found"
- Asset Header Info, Asset ID, Status, and Badge Number (highlighted when using Building filter) or "No match found"
- Asset Header Info, Asset ID, Status, and Address (highlighted when using Address filter) or "No match found"
- Asset Header Info, Asset ID, Status, and Description (highlighted when using Description filter) or "No match found"
- Filter Name and Search Criteria (always displayed)

To select an item from the search results, click the result.

Chapter 5

Assets

About Asset Locations

An asset location describes the physical location where the asset is *installed* after it leaves the storage location. Some examples of asset locations are service points, underground connections, poles, and so on.

Asset locations have their own history and lifecycle and can be organized within a reporting hierarchy. For example, a campus might be defined as the parent location with the buildings on the campus defined as child locations to that parent. The parent can define attributes or identifiers common to all child locations, however specific location information must be defined on each location individually.

Refer to [Understanding Inventory](#) for information on storage locations.

Types of Locations

Locations identify the physical location of an asset or component. An asset location defines the physical location of where an asset is *installed* while an out of service location defines the physical location where assets or components are *stored*.

Refer to [About Asset Locations](#) section for information about physical locations where assets are *installed*.

Assets that are no longer “installed” at a location are considered to be “out of service”. Valid out of service storage locations might be storerooms, warehouses, receiving docks, retirement locations, real estate, vehicles, or any other location as defined by your organization.

Some examples for storage locations include:

- A storeroom or repair shop is a location where assets are physically stored. This type of location can be composed of a simple description (‘Central Storage’ or ‘Repair’) but might also have address information.
- A real estate location is typically an address or a building name.
- A vehicle or "fleet asset" can be configured as a storeroom and as such, can have trackable stock items and their associated tracked assets stored in the vehicle. Vehicle storerooms can be configured using “Truck” as the valid Storeroom Type.

Service Area

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.

Location Hierarchy

A parent organization can include asset locations in its hierarchy. The locations can be defined in hierarchies with parent and child locations as part of the organization. The parent can define attributes or identifiers common to all child locations, however specific location information must be defined on each location individually.

A child location cannot be in active status if the parent location is inactive. It is also not possible to make a location inactive if it has associated assets or child locations.

Disposition Locations

Disposition location refers to either a storage location or an asset location.

Storage locations are considered to be storerooms, meter shops, vehicles, or warehouses while an asset location is considered to be the physical location of an installed asset whether or not the asset is in service, out of service, or retired.

Examples depicting the differences between a storage location and an asset location:

- Assets received from the manufacturer and awaiting inspection can be held at a receiving warehouse (Storage Location).
- An asset, once it passes inspection, can be moved to a storeroom (Storage Location).
- A service person can load the asset onto their service vehicle (Storage Location).
- If damaged, the asset can be removed from the service point and sent to the meter shop for repairs (Storage Location).
- Service person installs the asset at a service point (Asset Location).
- Assets, such as pipes and poles, that remain at the install location but are no longer in use (Asset Location).
- Once repaired, the asset can be installed at another service point (Asset Location) or returned to inventory (Storage Location).

Belongs to an Organization

When a new asset or storeroom is created, a parent organization can be entered which links the new asset location or storeroom to the organization.

Contacts

Contacts are usually individuals or business entities who should be contacted in reference to a service point, work, or any other issues.

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 person contact might be a customer calling about their asset or a business contact might be a foreman at a job site or supervisor of a storeroom. Once on the **Contact** portal, a user can click **Create Service Call** to report a service issue for the individual contact.

About Assets

An asset describes such objects as meters, poles, pipes, transformers, components, or any other material item owned or managed by an organization.

Key aspects of an asset are as follows:

- Have monetary or functional value
- Able to be inspected
- Able to be serviced and maintained
- Includes a disposition history
- Could include one or more components

Assets are always associated with a location. Refer to [About Asset Locations](#) for more information.

Assets typically exist within a structured hierarchy based on the relationship between the asset and its locations and organizations.

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 a “Connected to” field which allows 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.

Components

Components are devices or other objects that are attached to an asset.

One of the most common examples of components are the electronic reading devices (ERTs) that are often attached to analog meters. However, a component can be any device attached to another asset.

Since components are always attached to another device, components cannot be installed at a location. However, they may be received, tracked, maintained, and inspected like any other device. Given this, activities and work orders can be created to perform these tasks on components.

Detaching components involves either moving the component to a new asset or transferring the component to a storeroom, if tracked, or an out of service location, if not tracked. The process within the system is to search for and select the component and then "detach" it from the asset where it currently resides.

Types of Assets

There are really just two types of assets, an asset or a component. Objects such as meters, poles, pipes, lines, and transformers, are all considered assets, but components are also considered assets. What differentiates these two types of assets is the specific business object. The *asset* business object requires the creation of an asset location record, but the *component* business object requires no asset location record be created.

Refer to [Components](#) for information about components.

Refer to [About Asset Locations](#) for information about asset locations.

The Asset business object can be used to group multiple types of assets into categories:

- **General Assets** category involves *Tracked* assets, such as transformers and motors, and non-tracked assets, such as non-linear connected assets with the exception of fleet assets.
- **IT Assets** involve both tracked and non-tracked IT or facility assets associated to a person or employee record. An example might be meters and laptops.
- **Infrastructure Assets** involve connected assets that do not need to be tracked. An example might be pipes and poles.
- **Fleet Assets** are used for vehicles, such as trucks and loaders.

Asset Hierarchy

Assets typically exist within a structured hierarchy based on the relationship between the asset and its locations and organizations. The **Tree View** tab on the asset record displays the asset's location within the organization's structure.

Asset Disposition

An asset's disposition is the combination of the asset's location (storage location or asset location) and status. An asset's disposition changes throughout its lifecycle as the asset is moved between locations and service statuses.

The number of assets able to be installed at a location must be valid for the asset relationship (Multiple, Single) defined for the "Number of Assets Allowed" parameter on the location type.

Implementers can create additional asset dispositions; however, each asset disposition must indicate whether the disposition will represent an *Installed* or *Not Installed* asset and whether the asset is to be maintained by Inventory.

Some examples of Asset Disposition combinations are as follows:

- When an asset is *received from the manufacturer*, its disposition is *In Receipt* at the *receiving warehouse*.
- When an asset is *being inspected*, its disposition is *In Stores* at the *receiving warehouse*.
- When an asset is *moved into a storeroom*, its disposition is *In Stores* at the *storeroom*.
- When an asset is *moved into a service vehicle*, its disposition is *In Stores* at the *service truck*.
- When an asset is *installed at a service point*, its disposition is *Installed* at the *service point*.
- When an asset is *being repaired*, its disposition is *In Repair* at the *meter shop*.
- When an asset is *retired from service*, its disposition is *Retired* at the *storeroom*.
- When an asset is transitioning from *In Stores* to *Installed* or *Installed* to *InStores*, *Repair*, or *Retired*, its disposition is *Pending Disposition* in the system.

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 Character Large Object (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.

Replicating Assets

Asset Replication process enables a company to replicate large batches of assets or devices that they receive, inspect, and store in order to start tracking the assets.

Functionality enables such processes as:

- Replicating assets in the same lot or pallet
- Selecting assets for testing or inspection
- Mass updating of asset disposition, such as rejecting or accepting a lot or pallet.

Processing creates new asset IDs, badge numbers, and serial numbers for assets based on general information that is entered related to each batch. When a new replication event is created it is tied to a replication activity type to determine default settings and attributes for the replication. The actual asset information can be entered manually or uploaded using a CSV file.

Assets can then be moved between locations or dispositioned using the **Mass Asset Disposition Update** feature. See [Mass Asset Disposition Update](#) for more information.

Asset Service History

Information regarding service or maintenance on an asset is captured as service history and stored in the Asset Service History zone on the **Maintenance** tab of the asset record. An Asset Service History zone logs the date and time of the service as well as any attachments, if applicable, for every service history performed for the life of the asset.

Please refer to [About Service History](#) for more information.

Warranty Terms and Warranties

Warranty terms hold the duration, term, value, or usage limit of assets or items that are under warranty. The warranty term also specifies the specification and vendor that a warranty claim can be issued to when needed.

Warranty terms can be defined as calendar warranties (such as number of years) or runtime warranty terms (such as number of miles). When a warranty term is created and set to *Active* status, it can be associated with assets.

A warranty is a written guarantee issued to your organization promising to repair or replace a damaged item within a specified period of time or usage limit. The **Warranty** portal manages the warranties held for assets or other purchased items for your organization.

Once warranties have been created referencing the term, the **Warranty** section on the **Warranty Term** record displays a list of the applicable warranties.

About Operational Devices

Operational devices are any asset that is used in a metering system. These devices can be physical objects, such as smart meters, communication components, or communication relays, or can be virtual objects, such as firmware.

Please see other topics in this section for concepts related to operational devices.

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.

Since they almost entirely depend on the usage of the related asset, readings cannot be calculated or predicted accurately by the system. Instead, readings must be collected and entered into the system, either manually by a user or imported as the result of activity completion. Data readings that fall outside of expected limits or thresholds can be set to trigger alerts. The alerts create ToDo entries and can trigger the creation of related work records, as appropriate.

Operational Reading Activities

The taking of measurements can be effectively managed by setting up the appropriate work orders or activities to track and schedule the work. This type of scheduling can be even more stringently managed using maintenance triggers and schedules.

Asset records display any related measurement activities in the **Measurements** zone.

Operational Reading Quick Entry

While most readings typically come from an external system via an interface, users can also enter read information manually using the Reading Quick Entry tool. This tool can also be used to upload multiple readings from a spreadsheet.

Operational Reading Alert Review

Reading alert review provides users with an easy dashboard to review alerts related to asset measurements. Generally an alert is triggered when an asset reading falls outside of an established threshold. Users can select the appropriate measurement type to see a list of measurements of that type that are in error. This list is sorted by the most recent errors first. The **Asset** portal displays a similar zone on relevant assets that have had measurements taken. This zone also includes an **Add** link where users can enter a measurement directly on the **Asset** record.

Select appropriate assets (usually assets that have readings in error) and click the **Make Valid** button which automatically moves the status of the reading from an error to a valid state.

Measurement Activity zone

This zone displays a listing of activities related to a measurement. This provides visibility into maintenance planning activities that are triggered by measurement alerts. For example, your system configuration might trigger an alert if an out of range measurement is entered.

This zone also includes an **Evaluate** button which, when clicked, moves the measurement to the next state where the measurement ranges will be evaluated, such as valid or error.

Device Configuration

Device configurations represent specific configurations of a device as of a certain time. Over time, a device can have many configurations.

Device configurations are effective-dated allowing a device to retain its identifiers (such as serial number, badge number, and so on.) even while the quantities it measures change over time.

Each device configuration can have one or more measuring components associated with it. Attributes used to define device configurations include the following:

- The device configuration type
- The parent device

- The effective date and time of the device configuration
- The time zone in which the device operates

About Specifications

Specifications describe design details and functional performance criteria for a set of assets. This can include manufacturer, model, and other information specified for the assets.

Specifications may also contain written procedures, careful handling instructions for hazardous or delicate materials, or any other set of attributes needed to provide information about a set of assets.

The base package specifications provide a general starting point for your specification set-up including the following:

- General Maintenance Specification
- Operational Maintenance Specification
- Operational Device Specification
- Fleet Specification
- General Specification
- IT Specification
- Infrastructure Specification

In Oracle Utilities Operational Device Management

- Analog Meter Specification
- Communication Component Specification
- Communication Relay Specification
- Firmware Specification
- Smart Meter Specification

Implementers may want to include additional specification details. An example of how this can be accomplished using the fleet or vehicle specification would be to extend the data area *WI-VehicleDetail*. 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

Some of the main uses for specifications are as follows:

- **Configuration Specifications** enable definition of different attributes on each device type.
- **Engineering Specifications** enable creation of Manufacturer, Bill of Material (BOM), and Model details for specific types of assets.
- **Preventive Maintenance Specifications** provide for the building and sharing of maintenance strategies between an asset or group of assets.

Specifications can be linked to stock items. This linking associates the stock items to the tracked assets that are also associated with the specification which 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.

Please note the following with regard to specifications:

- Specifications often include an attachment which supports the specification definition or provides the actual specification from the manufacturer.
- A specification may also include default *in service* and *out of service* maintenance schedules for assets or components linked to the specification. When the specification is associated with the asset or component, the related maintenance schedule becomes in effect for the asset or component.

Firmware

Firmware is software that is embedded in a piece of hardware and enables the hardware to work as the manufacturer intended. For example, commands to enable relay devices to automatically open and close.

Refer to [About ODM Firmware](#) for more information.

About Functional Processes

A functional process is a grouping of asset locations that contribute to a larger function together. For example, multiple asset locations throughout the city that contribute to waste water functions can be grouped together under that process and examined for how well they are contributing.

The **Functional Process** portal gives users the ability to associate asset locations to multiple functional processes. Using water hydrants as an example, the hydrant can be part of the overall **Water Distribution** process and can also be part of the overall hydrants within the city. Multiple asset locations can be added at one time to any functional process using the **Add Asset Location** zone, which is only available when the functional process is in a *Created* status, just like the **Edit** hyperlink in the **Functional Process Location** zone.

The portal enables maintenance and renewal costs for the assets at those asset locations to be tracked across the functional process. Cost data is segregated based on cost category and broken down by **Month to Date**, **Year to Date**, and **Lifetime**. This information is display-only. User can view the cost summary for a specified period or narrow the period by specifying a different “from” and “to” date.

The portal also provides multiple ways to create work orders against a functional process:

- Users can select the **Asset Location** context menu from the **Functional Process Location** zone on the **Main** tab.
- **Work Creation** zone on the **Maintenance** tab enables users to create work orders against selected asset locations and assets.
- Select the **Tree View** tab and drill down to the **Asset** context menu to create the work order.

Also, the **Functional Process** zone on the **Maintenance** tab of the **Asset Location** record displays all of the functional processes associated with a particular asset location.

Functional Process Hierarchy

A functional process has its own hierarchy. The selected asset locations for the functional process can include or not include their child locations. An asset location can be associated with many functional processes irrespective to where it sits within the hierarchy.

Refer to [Maintaining Functional Processes](#) in the **Business User Guide** for more information.

Refer to [Understanding Functional Process Types](#) in the **Administrative User Guide** for more information.

Maintaining Asset Locations

This portal is used to display and maintain asset location records.

Refer to [About Asset Locations](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, then select **Asset Location**. You are brought to a query portal with options for searching for a specific asset location. Once an asset location has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Asset Location:** This zone provides details about the physical location where the assets are installed.
- **Currently Installed Assets:** This zone provides a list of all the assets currently installed at this location.
- **Service Point Installed Assets:** This zone provides a list of all the assets installed at this service point.
- **Asset History:** This zone provides log details for the assets installed at this location.
- **Asset Summary:** This zone lists the number of assets, by asset type, that are currently associated with the asset location.
- **Child Locations/Organizations:** This zone lists the child locations and organizations associated with the asset location or organization.
- **Attachments:** This zone enables users to manage all types of attachments for the asset location record.

The following zones may appear as part of the **Maintenance** tab page:

- **Service History:** This zone displays a log of the service histories recorded for the asset location. Also, new service history can be created from this zone.
- **Activity History:** This zone provides a log of the activities that have been performed at this asset location.
- **Permit:** This zone displays permit information for this asset location.
- **Permit Service History:** This zone displays a list of the permits that have been associated with assets at this location.
- **Asset Location Business Flag:** This zone displays the most recent business flags for the Asset Location which were created in last 30 days.

The following zones may appear as part of the **Tree View** tab page:

- **Location Hierarchy:** This zone broadcasts a hierarchy view of the asset location.
- **Currently Installed Assets:** This zone provides a list of all the assets currently installed at the location.

The following zones may appear as part of the portal's **Statistics** tab page:

- **Asset Location Cost Summary:** This zone displays a table of maintenance, renewal, and operational cost per cost category over the life of the asset. Costs buckets for each cost category are **Month To Date**, **Year To Date**, and **Lifetime** costs.
- **Cost by Service Class:** This zone displays the total cost breakdown by service class for this asset.

Maintaining Contacts

The **Contact** portal is used to display and maintain contacts.

Refer to [About Asset Locations](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, then select **Contact**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Contact:** This zone displays information about the contact.
- **Location Referencing Contact:** This zone displays location information for the contact.
- **Contact Relationship:** This zone displays information on all the records associated with the contact.

- **Communication Log:** This zone displays a log of the contact's communications.

Maintaining Assets

This portal is used to display and maintain asset records.

Refer to [About Assets](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, and then select **Asset**. You are brought to a query portal with options for searching for a specific asset. Once an asset has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Asset:** This zone provides details about the asset, such as the badge number, disposition, and status as well as any contextual insights if configured.
- **Attached Components:** This zone provides a list of components that are installed on the asset.
- **Asset Attributes:** This zone displays a list of attributes for the asset.
- **Attachments:** This zone enables users to manage all types of attachments for the asset record.
- **Asset Warranty:** This zone provides warranty information for the asset.
- **Asset — Notes:** This zone enables users to enter additional notes for the specific asset.

The following zones may appear as part of the **Maintenance** tab page:

- **Asset Service History:** This zone displays a log of the service histories recorded for the life of the asset. Also, new service history can be created from this zone.
- **Measurements:** This zone provides a log of measurements created for the life of this asset.
- **Maintenance Trigger:** This zone displays a log of preventive maintenance triggers that are associated to the maintenance schedule identified on the maintenance specification on the asset header.
- **Asset List Maintenance Trigger:** This zone displays a log of location specific maintenance triggers where the asset's location is present on a template activity's location list of the template work order identified on the PM trigger.
- **Maintenance Schedule Forecast:** This zone displays a log of maintenance schedules forecasted for the asset.
- **Activity History:** This zone provides a log of the activities that have been performed on the asset over the life of the asset.
- **Hazardous Materials:** This zone displays a list of hazardous materials associated with the asset.
- **Permit:** This zone displays permit information for this asset.
- **Permit Service History:** This zone displays a list of the permits that have been associated with the asset.
- **Asset Business Flag:** This zone displays the most recent business flags for the asset.
- **Functional Process:** This zone displays the functional processes associated with the current asset record.

The following zones may appear as part of the **Tree View** tab page:

- **Asset Hierarchy:** This zone displays a hierarchy view for the asset.
- **Currently Installed Assets:** This zone displays a list of currently installed assets at this location.

The following zones may appear as part of the portal's **Statistics** tab page:

- **Asset Cost Summary:** This zone displays a table of maintenance, renewal, and operational cost per cost category over the life of the asset. Costs buckets for each cost category are **Month To Date**, **Year To Date**, and **Lifetime** costs.
- **Cost by Service Class:** This zone displays the total cost breakdown by service class for this asset.

- **Asset Condition Score Breakdown:** This zone displays a list of assessment scores for each assessment type for the asset.

The following zones may appear as part of the **Asset Disposition** tab page:

- **Disposition History:** This zone displays all of the status and location changes that have occurred during the life of the asset.
- **Components History:** This zone displays all the components that were and are currently attached to the selected asset during its life.

The following zones may appear as part of the portal's **Asset Performance** tab page:

- **Asset Performance:** This zone provides asset and performance information about the asset that can be used for planning purposes.

The following zones may appear as part of the portal's **Log** tab page:

- **Asset Log:** This zone displays log information for the asset.
- **Asset Sync Requests:** This zone displays sync requests associated with the asset.
- **Asset Shipping Memos:** This zone lists the shipping memos associated with the asset.

Maintaining Asset Replications

This portal is used to display and maintain large quantities of assets or other devices by batch.

Refer to [About Assets](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, then select **Asset Replication**. You are brought to a query portal with options for searching for asset replication information. Once an asset replication has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Asset Replication:** This zone provides details about the replicated assets, asset controls, and new asset IDs.
- **Completion Event:** This zone displays the completion events associated with the replicated assets.

Maintaining Operational Readings

The **Operational Reading** portal is used to display and maintain an operational read record.

Refer to [About Operational Devices](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, and then select **Operational Reading**. You are brought to a query portal with options for searching for a specific operational read. Once the read has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Measurement:** This zone displays information about the operational reading.
- **Measurement Activity:** This zone lists the measurement activities associated with the operational reading.

Maintaining Specifications

This portal is used to display and maintain specification records.

Refer to [About Assets](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, then select **Specification**. You are brought to a query portal with options for searching for a specific specification. Once the specification has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Specification:** This zone provides details about the specification.
- **Warranty Term:** This zone lists the warranty terms associated with the specification.
- **Specification Assets:** This zone provides information about the asset associated with the specification.
- **Attachments:** This zone enables users to manage all types of attachments for the specification record.

Maintaining Functional Processes

The **Functional Process** portal is used to display and maintain functional processes.

Refer to [About Functional Processes](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, and then select **Functional Process**. You are brought to a query portal with options for searching for a specific functional process. Once the functional process has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Functional Process:** This zone displays general information about the functional process.
- **Functional Process Location:** This zone lists the asset locations, with an indicator if child locations are included, associated with the functional process.
- **Attachments:** This zone enables users to manage all types of attachments for the functional process record.

The following zones may appear as part of the **Maintenance** tab page:

- **Maintenance Trigger:** This zone displays a list of maintenance triggers for the asset locations associated with the functional process. Broadcasting a maintenance trigger populates the **Activity History** zone search fields to display the PM work orders for that trigger at that asset location and for that asset.
- **Asset List Maintenance Trigger:** This zone displays a list of location specific maintenance triggers where the asset's location is present on a template work order identified on the PM trigger. Also, broadcasting an asset list maintenance trigger populates the **Activity History** zone search fields to display the PM work orders for that trigger at that asset location and for that asset.
- **Work Creation:** This zone enables work to be created for the asset location associated with the functional process.
- **Activity History:** This zone provides a list of all the activities that have been performed at the asset location over the life of the asset for this functional process. This zone also displays a filtered list of activities once the user broadcasts a line from the **Maintenance Trigger** zone or from the **Asset List Maintenance Trigger** zone.

The following zones may appear as part of the portal's **Tree View** tab page:

- **Functional Process Hierarchy:** This zone displays a hierarchy view of asset locations, assets, and components associated with the functional process.

The following zones may appear as part of the portal's **Statistics** tab page:

- **Functional Process Cost Summary:** This zone displays a summary of the actual maintenance, renewal, and operational costs per expense category for the functional process. Cost data is segregated based on cost category and broken down by **Month To Date**, **Year To Date**, and **Lifetime** costs.
- **Functional Process Period Cost:** This zone displays the actual maintenance and renewal costs which have been charged to the associated asset locations for the functional process for a specified period by *From* and *To* date.

Maintaining Warranties

The **Warranty** portal is used to display and maintain a warranty record.

Refer to [About Assets](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, and then select **Warranty**. You are brought to a query portal with options for searching for a specific warranty. Once a warranty has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Warranty:** This zone lists information about the warranty and associated asset.
- **Attachments:** This zone enables users to manage all types of attachments for the warranty record.
- **Warranty - Notes:** This zone displays notes associated with the warranty.

Maintaining Warranty Terms

The **Warranty Term** portal is used to display and maintain a warranty term record.

Refer to [About Assets](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, and then select **Warranty Term**. You are brought to a query portal with options for searching for a specific warranty term record. Once a warranty term has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Warranty Term:** This zone displays information about the terms of the warranty.
- **Warranty:** This zone lists information about the warranty and associated asset.
- **Attachments:** This zone enables users to manage all types of attachments for the warranty term record.

Mass Asset Disposition Update

The Mass Asset Disposition Update portal is used to move many assets of the same type from one asset location to another asset location. The target location is always a single asset location. For example, if a pallet of meters has been received in the receiving asset location, mass disposition can be used to move this batch of assets to a storage asset location or an out of service location. You can also use Mass Asset Disposition Update to retire tracked assets or to put a tracked asset into a storeroom location such as a repair shop.

Refer to [About Assets](#) for more information.

NOTE: Assets that are inventory tracked (managed as stock items via inventory control) cannot be moved in or out of inventory storerooms using mass asset disposition update. In this case, the asset must be moved using stock transactions.

You can access the portal from **Menu**, select **Asset Management**, and then select **Mass Asset Disposition Update**. You are brought to a query portal with options to change the status and location of multiple assets at one time. Select the appropriate assets and click the **Change Disposition** button to execute the change.

Operational Reading Alert Review

The **Operational Reading Alert Review** portal is used to review operational reading alert records.

Refer to [About Operational Devices](#) for more information.

You can access the portal from **Menu > Asset Management > Operational Reading Alert Review**. You are brought to a query portal with options for searching for operational reading alerts by measurement type. Select the appropriate reading alerts and click the **Remove Alert** button to remove the alert.

360 Degree Search

The **360 Degree Search** portal is used to search for assets, components, specifications, location, and contacts.

Refer to [About Operational Devices](#) for more information.

You can access the portal from **Menu**, and then select **360 Degree Search**. You are brought to a query portal with options for searching for a specific object record.

From the search results list, you can navigate to the selected portal to view data for the object.

This feature is only used in **Oracle Utilities Operational Device Management**.

GIS Map Viewer

The GIS Map Viewer provides a method to view WAM assets and work information based on geospatial mapping. Maps generally consist of a base map plus feature layers which can overlay the base to show various map features such as work locations, infrastructure, and assets.

You can access the portal from **Menu**, then click **GIS Map Viewer**. You are brought to a map with options for searching and viewing work, assets, specific locations and so on.

General Tools

- The **Home** button returns the user to the original map position and extent.
- A pop out **Map Overview** is available in the upper right hand corner that provides a larger context of the current position.
- Additional tools include:
 - Zoom In
 - Zoom Out
 - Set map location as user default
 - Zoom to previous
 - Zoom to next
- Selection tools such as rectangle, freehand, polygon or by location.
- Click a highlighted object (such as an asset or work location) on the map to display a popup with more details.
- Select assets then click **Create Work** to open the WAM dialog.
- Drop a pin to select an exact point on the map. This can also then become a work location which you can **Attach to Asset** to indicate the linear asset that the work is associated to.

Dashboard Controls

- **Global Search:** A **Global search box** is provided that will find any address or place by typing them into this search.
The address search is especially useful in cases where users need to find a location in based on a customer's description. For example, if a customer calls in about a trouble situation (for instance, a water leak, live electricity, and so on.), the WAM user can type in the address details provided by the customer to pinpoint the location of the issue. The work that is created can then be attached to the closest asset in the proximity of the issue.
- **Asset tab:** The **Asset** tab is used to select a specific list of assets and view the list in the dashboard panel. There are actions available for individual records (for example, Quick View, Create Work Request, and Remove From List) as well as actions that can be performed on the full list of records (for example, Create Work Order or Create Activity). Links are also provided to navigate to the transactional screen for an asset.
 - **Drop pin:** the drop pin button allows the user to click on a specific point anywhere on the webmap, and either create a work request at that location, or to link that point to the related linear asset and create a work order. These 'pin drops' are represented in WAM as Work Locations.
- **Work tab:** The **Work** tab is used to display details of work related items, focused on Work Requests and Activities. Filtering options are provided to view only the work that applies to the current need, based on status, date range, and other criteria.
 - For **Work Requests**, options are provided to perform a Quick View of the record or navigate to the transactional screen for the Work Request.
 - For **Activities**, options are provided to perform a Quick View of the record, view all assets on the map related to the Activity, view all assets on the map related to the Work Order for the Activity, or navigate to the transactional screen for the Work Request.
 - By default, the **Work Locations** layer is enabled when the **Work** tab is accessed, since work records can be associated with these locations.
- **Layers tab:** The **Layers** tab provides the ability to turn map layers on or off. A drop-down is provided to access the configured web maps.
- **Share tab:** The **Share** tab allows for two key redlining capabilities:
 - Tools are provided so a user can write and/or draw on the map to call attention to something, provide instructions on the map, or add any other notes that would be useful if displayed on the map.
 - The map can be exported to a shareable format such as email or export to PDF.

Highlighting Assets

- **General Highlighting**
 - Your organization configures custom colors for **Highlighted Assets**, **Assets with No Work**, **Selected Assets**, and **Related Assets**. These are set in **Master Configuration**.
 - Hovering over a result in the **Asset** tab displays the result in the color configured for **Highlighted Asset**.
 - Assets that are associated with an activity record in the **Work** tab are highlighted in the **Related Assets** color. The related assets for an activity can be highlighted using the **Highlight** icon, which appears for each of the activity records in **Work** tab results.
- From the **Asset** tab:
 - Selecting results from the **Asset** tab displays map features that have a corresponding **Asset Location** in WAM. The selected assets are displayed in the color configured for **Selected Asset**. Features are not selected if the asset location is not found in WAM.
 - If you make a selection on the **Work** tab, then navigate to the **Assets** tab, the system retains the selection if the selected assets have work associated.

For example, If you are on the **Work** tab and use the **Work** selection wizard to select 20 map features (out of which work is created and matched with the filter criteria for only 8 features in WAM), the 8 features are highlighted in the **Selected Asset** color and there is no feature selection for the rest of the GIS assets. When you click the **Asset** tab, the 8 highlighted features are retained. If all 8 features are mapped assets in WAM, the **Selected Asset** color will be applied to the 8 features updating the asset information in **Asset** tab results. If you click back on the **Work** tab, all 8 features will remain highlighted in the **Selected Asset** color.

- From the **Work** tab:
 - Selecting results from the **Work** tab displays map features that have work created (the underlying map feature has a corresponding asset location in WAM and has a work record that matches the default filter criteria. For the rest of the features, no selection color is displayed.
 - If no assets are selected when you navigate to the **Work** tab, the system looks for linked work on all assets within the boundary extent.
 - If you make a selection on the **Asset** tab then navigate to the **Work** tab, the selection from the **Asset** tab is maintained. The assets that have related work created are displayed in the color configured for **Selected Asset**. Any other assets are displayed in the color configured for **Assets with No Work**.

For example, if you are on the **Asset** tab and use the selection wizard to select 20 map features, but only 12 are mapped in WAM, the 12 features will be highlighted in the **Selected Asset** color and the remaining 8 will be highlighted in the **Assets with No Work** color. If you then click on the **Work** tab, the 12 features will be retained. If you click back to the **Asset** tab, all 12 features will remain highlighted in the **Selected Asset** color.

- From the **Map**:
 - If you select assets, then navigate to the **Work** tab, the viewer only looks for linked work for the selected assets.
 - If no assets are selected when you navigate to the **Work** tab, the system looks for linked work on all assets within the boundary extent.

GIS Work List Companion

The GIS Work List Companion is used in conjunction with the Work List to enable users to plot objects, such as assets, activities, work orders, and so on, in the GIS Map Viewer.

NOTE: GIS Work List Companion is available only if your system is configured for GIS.

An example of how this might be used is as follows:

- Query the system for the objects to be plotted. For example, perform a search in either the Assets or Activity portal to retrieve a list of search results.
- Click the **Work List** icon to move the objects of the search results to the **Work List Dashboard** zone.
- **View Work List in Map** button is enabled in the **Work List Companion** zone on the **Dashboard**.
- Click **View Work List in Map** to plot the objects.

Chapter 6

Operational Device Management

About Operational Device Management

Oracle Utilities Operational Device Management provides functionality to handle large volumes of assets and to manage the receipt, installation, maintenance, tracking, and removal of those assets. Oracle Utilities Operational Device Management is purchased as a separate license to further enhance Oracle Utilities Work and Asset Management. However, since these two products are typically always sold together, this guide includes seamless documentation for both products. Contact your sales representative or Oracle Support for more information.

The system supports the following business processes:

- Capturing device attributes and supporting any type of device
- Supporting integration with other utility applications requiring Device information and configuration
- Supporting detailed smart device attributes
- Supporting detailed location management and tracking of individual devices throughout their lifecycle
- Capturing device configurations, settings and calibrations
- Tracking firmware and firmware versions on smart devices
- Managing configuration between devices, components and software
- Validating devices at the time of receipt
- Scheduling and capturing data related to periodic inspection history over time
- Defining cycle information to schedule inspections
- Defining procedures to follow when performing inspections
- Using field activities to track and manage the electronic or in-field maintenance of devices
- Creating field activity documents to be used in the field for inspection work
- Defining a field activity life cycle

Refer to the [Understanding Financials](#) section in the **Administrative User Guide** for configuration guidelines.

About 360 Degree Search

The **360 Degree Search** portal is used to search for assets, components, specifications, locations and contacts. Users can also search on other identifiers or characteristics as defined by your organization.

Refer to [About Operational Device Management](#) for more information.

You can access the portal from **Menu**, then select **360 Degree Search**. You are brought to the 360 Degree Search portal with options for searching.

Selecting different **Search By** criteria from the portal's **Main** tab page will change the search fields to present different options. From the search results list, you can navigate to the selected portal to view data for the object.

This feature is only used in **Oracle Utilities Operational Device Management**.

About 360 Degree View

The **360 Degree View** feature allows users to view data for assets, components, contacts and locations from a single set of portals.

Refer to [About Operational Device Management](#) for more information.

The **360 Degree View** has a separate portal for each type of object:

- Asset Information
- Asset ID
- Asset Specification
- Asset Location Information
- Storeroom Information
- Location ID
- Contact Name

Once you select one object, the **360 Degree View** provides related information so that you have full visibility into all information about that object. For example, if you select Asset Information and search on an asset type, the resulting view portal shows the components, service history, activity history, and location information for assets of that asset type.

This feature is only used in **Oracle Utilities Operational Device Management**.

About ODM Configurations

A single asset or component can have many parameters and only certain combinations of parameter values that are allowed. Multiple assets and components may work together as a group with only certain combinations of settings that are allowed or supported.

Configurations document and manage the valid set ups for assets and components. This includes a specification for each asset or component in the configuration as well as the valid value for each parameter, such as the firmware version of each asset.

Configuration reports can be generated to provide a summary of any currently installed configurations. These reports show whether the configuration in an installation is valid or invalid, they show the identifiers and values being used, and they display all of the assets, components, and other elements that make up the installed configuration.

If a configuration is found to be invalid or in need of reconfiguration, field activities can be created based on configuration criteria. Configuration functionality can also be tied in with specifications to manage firmware version testing and upgrades.

Configuration Statuses

Configurations also allow you to define particular statuses. Applicable statuses include:

- Pending
- Under Testing
- Rejected
- Approved
- Deprecated
- Obsolete

Use actions in the **Record Actions** section to transition the configuration to the appropriate status.

About ODM Configuration Reports

Configuration reports provide a summary of any currently installed configurations.

These reports show whether the configuration in an installation is valid or invalid, the identifiers and values being used, and display all of the assets, components, and other elements that make up the installed configuration. If the configuration is invalid, the configuration column is blank in the report results. Navigate to the configuration itself to review the set up. A button next to each configuration also allows you to generate a field activity to correct, re-configure, or otherwise maintain any of the configurations.

Configuration reports are defined by the following:

- **Reporting Mode:** The mode in which the report is run (One Time or Recurring). Batch processing and the setting of the **Reporting Mode** field on a report determines when a configuration report is generated after it is submitted. If a report is set to generate one time, the report status is set to *Pending* after it is generated and it will not be updated again until a user manually submits the report. If the report is recurring, the report is updated every time batch processing runs.
- **Status:** The current status of the report (Pending or Submitted)
- **Report Criteria:** Specific information used to establish the filter criteria when generating the report, such as the assets and components involved, the firmware, characteristics, identifiers and other important information.
- **Report Result:** Specific information for each configuration, including the number of configurations, the assets and components involved, and the specifications and firmware versions of each. If the configuration is invalid, the **Configuration** column is blank in the report results. Users can navigate to specific valid configurations to review and edit the setup as needed. In addition, users can generate a field activity to correct, reconfigure, or otherwise maintain any of the configurations by clicking **Generate Activity**.

Base Package Configuration Reports

The base package includes a number of configuration reports, each based on one of the supported configuration classes (see [Understanding Configuration Types](#)).

The table below lists the configuration reports and the business object for each.

Configuration Report	Business Object
Single Asset Configuration Report	W1-SingleAssetCfgRpt
Asset/Component Configuration Report	W1-AssetCompCfgRpt
Asset to Asset Configuration Report	W1-AssetToAssetCfgRpt

Configuration Report	Business Object
Asset/Component to Asset Configuration Report	W1-AssetCompToAssetCfgRpt
Asset to Asset/Component Configuration Report	W1-AssetToAssetCompCfgRpt
Asset/Component to Asset/Component Configuration Report	W1-AssetCompToAssetCompCfgRpt

About ODM Firmware

After a firmware is released from the vendor, new firmware may need to be tested and validated then either approved or rejected for roll out. The system provides the needed functionality to support validating and updating firmware versions as needed.

Associating Firmware and Firmware Versions to Assets

In most business processes, firmware and firmware version is entered as part of the manufacturer settings when assets are created. This information appears in the **Settings** section on the asset. Also during asset creation, a basic business process would associate an asset specification to the asset to keep track of detailed manufacturer information which might also include the firmware. Attached, as a child to that specification, would be another specification that indicates the firmware versions that are available.

As firmware versions change the child specification is updated with new versions. These versions can then be tested and accepted or rejected using the lifecycle functionality on the specification.

Next, field activities can be created directly from the **Field Activity** portal or by using the **Activity Generator** to update the firmware version on the assets that are associated with those specifications. For example, your business process might require that you create an *Asset Criteria Generator* type record to create a list of assets based on firmware as the selection attribute. The system would then apply the updates from the record to any assets that meet that criteria (for example, that have that firmware type or version).

Firmware Management

You can use the system to manage firmware in the following ways:

- **Create firmware groupings:** Record firmware versions into groupings or "families" on specifications so that related versions can be easily managed together.
- **Add firmware to assets:** Use peer specifications to associate assets to these firmware groups.
- **Review firmware:** Review firmware distribution and status in the **Firmware Deployment** and **Firmware Version Dispositions** zones on specifications.
- **Document firmware:** Manage related release notes or manufacturer information by creating attachments directly on the specification.
- **Test firmware:** Track testing and roll out of new firmware versions including the ability to approve or reject firmware version deployments.
- **Update firmware:** Create field activities to update firmware versions on assets.
- **Reject firmware versions:** After testing, set the firmware version to an inactive status to indicate that the firmware version cannot be used for assets.

The **Configurations** portal manages asset to component to firmware configurations that determine which versions are valid for which assets. **Configuration Reports** can be used to obtain details about the configurations and verify that the correct versions are in use.

Firmware on Specifications

The **Firmware Deployment** section on specifications provides statistics on the number of assets that are on each version of the firmware that is applicable to the specification. Click the pie chart for a version to see the number of assets on that versions grouped by the asset state.

The **Firmware Versions** section lists the firmware that has been added using the specification. Use the **Add Firmware Version** link in the zone to add a new version. Click on one of the listed versions to open a screen with more details about the firmware. This screen can also be used to add attachments for the specific firmware version. For example, you might attach the release notes that were delivered with the firmware release. On this screen, the **Firmware Version Dispositions** section lists the assets that use the firmware version indicated, if any.

Firmware Version Lifecycle

Statuses for firmware version updates can be managed on the specification by selecting the actions in the **Record Actions** section. Valid statuses include:

- Active Statuses
 - created
 - under testing
 - approved
- Intermittent Statuses
 - deprecated
 - quarantined
- Inactive Statuses
 - rejected
 - obsolete

If the firmware version is in an intermittent status, assets or components can reference that version. In general, these statuses are for documentation purposes. For example, an organization might use the **Deprecated** status to indicate that they do not want to roll out new meters on that version any longer, however, existing meters that are on that version are still valid. No assets or components can have the version associated if the version is in one of the inactive statuses.

About Activity Generators

Activity generators are used for mass generation of activities. An Activity Generator business object is selected when creating activity generators, and is used for entering the criteria values for searching assets and asset locations that will be linked to the activities to be created. Such criteria includes location type, asset identifiers, location identifiers, address components, asset disposition, indication if asset is installed or not, asset type, specification and among others. Required information also includes, the activity type, disposition of the created activities, maximum number of activities that will be created, and an indication if the activities will be created via batch processing.

The base package includes the following set of activity generator business objects containing the various criteria:

- **Asset Criteria** (W1-AssetCriteriaActGenerator): used to define asset-related criteria used to select the assets for which activities will be created (for example, activities for all assets for a certain asset type, a specified current disposition or condition, or whose identifiers contain specific characters, and so on).
- **Asset List** (W1-AssetListActGenerator): used to define a list of assets for which activities will be created.
- **Component Criteria** (W1-CompnCriteriaActGenerator): used to define component-related criteria to select the components for which activities will be created (for example, asset inspection activities for all assets for a certain component type, a specified current disposition or condition, or whose identifiers contain specific characters, and so on).

- **Component List** (W1–CompnAssetListActGenerator): used to define a list of assets for which activities will be created.
- **Location Criteria** (W1–NodeCriteriaActGenerator): used to define location-related criteria to select the asset locations for which activities will be created (for example, activities for all locations where assets are current installed, locations of a specific type, or whose identifiers contain specific characters, and so on).

Activity generators are defined by:

- **Status:** The current status of the activity generator.
- **Batch Generation Requested:** A flag that indicates if the activities created by the asset activity generator should be generated using batch processing.
- **Generator Information:** Details related to the specific activity generator business object (Asset List, Asset, Location, and so on) used by the asset activity generator.
- **Activity Information:** Details for the types of activities created by the asset activity generator, including activity type, default disposition, activation and work lead days (defaulted from the asset activity type), and the maximum number of activities to be created.

Specific activity generators contain additional information based on their business object. For instance, activity generators based on the **Location Criteria** business object includes **Location Information** and **Address Information** sections.

Maintaining ODM Configuration Reports

This portal is used to display and maintain configuration reports.

Refer to [About ODM](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Configuration Report**. You are brought to a query portal with options for searching for configuration reports. Once the report has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Configuration Report:** This zone provides details about the report.

Maintaining Activity Generators

The **Activity Generator** portal is used to view and maintain activity generators.

Refer to [About Work Orders](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Activity Generator**. You are brought to a query portal with options for searching for specific record. Once your record has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portals **Main** tab page:

- **Activity Generator:** This zone displays basic attributes of the selected activity generator record.
- **Activity Info:** This zone displays summary information about the activities created by the activity generator.
- **Notes:** This zone lists any notes that have been added for the activity generator.

The following zones may appear as part of the portals **Activities** tab page:

- **Activity:** This zone displays a list of activities created by the activity generator.

The following zones may appear as part of the portals **Cost** tab page:

- **Work Order Cost Summary:** This zone displays a summary of work order costs for the activity generator.

The following zones may appear as part of the portals **Log** tab page:

- **Activity Generator Log:** This zone displays the server logs for the activity generator.
- **Related Stock Transactions:** This zone displays the related stock transactions for the activity generator.

Chapter 7

Inventory

About Inventory

Inventory is tracked, ordered, and received from vendors and allocated in the inventory system. As 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.

Stock Items

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.

The stock item record can also provide a list of vendors and manufacturers who provide the item as well as appropriate alternative stock, as applicable. Only stock items in an “active” status can be purchased or entered as a material requirement on an activity. When they are entered in the system, stock items are categorized in some of the following ways:

- **General:** Regular stock items are kept in inventory until they are needed.
- **Direct Purchase:** Direct purchase stock items can only be directly purchased when needed for work.
- **Trackable:** 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, or installed on other assets, and are otherwise carefully dispositioned in the system.
- **Lot Managed:** 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 of 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 may have 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 Stock Item Detail (SID) for that lot number and creates one if one does not exist.

- **Capital Spare:** Capital Spare items are identified during accept and return as fixed assets so that an external Fixed Asset Management System can maintain their depreciation. A property unit is required on capital spare items.

Some stock items, such as Lot Managed stock items, should be inspected when they are received from the vendor. The “Accept on Receipt equals No” setting on the stock item record forces the inspection of the stock item upon receipt. This setting also enables a shipment deficiency to be created when there is an issue with the shipment. The deficiency can then be displayed in the **Shipping Deficiency** zone of the **Receipt Line** portal.

When a stock item is maintained in the inventory of a storeroom, a corresponding **Stock Item Detail** record must be created. Stock item details define how the stock item is managed in the storeroom and contain storeroom-specific information for the stock item such as pricing, reorder information, unit of issue, unit of purchase, expense code, and so on.

One stock item might be stored in multiple storerooms. When it is needed, stock transfers can be used to move items between storerooms. For example, an organization might have a central storeroom from which satellite storerooms request stock.

Stock Item Details

Stock Item Detail (SID) records define how a stock item is managed in the storeroom as well as tracking quantities for stock items. Each record contains storeroom-specific information for the stock item such as pricing, reorder information, unit of issue, unit of purchase, expense code, and so on.

Every stock item that is to be maintained in the inventory of a storeroom must have a corresponding **Stock Item Detail** record.

The Stock Item Detail record determines whether or not the type of stock can be purchased, repaired, tracked, and so on. A 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.

The **Stock Item Inventory** section provides a graphical representation of what is in stock with lines representing maximum quantity and reorder quantity. Use the filters to change the display on the chart. This chart can be changed to show projected inventory (taking into account pending purchases and pending activities), or historical inventory (showing actual inventory use and receipts).

Inventory Details

Inventory Details is an individual widget displayed within the **Dashboard** which displays the inventory quantities for the selected stock item detail record:

- **Inventory** is the quantity currently in the storeroom.
- **Average Unit Price** is the average unit price of the stock item detail.
- **Reserved** is the physical amount of stock items reserved in inventory. If it is not in inventory, it cannot be reserved.
- **On Demand** amounts are used in determining whether inventory can meet future needs. Amounts included in **On Demand** are “Active” activities with resource requirements and material requests, not referencing an activity, that are in a “Ready to Issue” state.
- **Pending Demand** indicates that an “Approved” activity includes a resource requirement.

- **In transfer** means stock is being transferred from one storeroom to another. The stock is considered “In Transfer” for the target storeroom after the originating storeroom completes their issue and before the target storeroom actually receives the stock.
- **On Order** is the amount derived from an “Issued” purchase order. This is before it has been received and accepted into inventory.
- **Pending Order** is the amount derived from an “Approved” purchase requisition before it is turned into a purchase order.
- **Available Quantity** is the amount of a stock in inventory plus what is due in (Pending Order, On Order, In Receipt, and In Transfer) and subtracting what is due out (On Demand, and Reserved).

Physical Inventory

Physical inventory is an auditing procedure within **Inventory Management** which your organization can use to ensure that the physical quantities of items, materials, and stock that are housed in your storerooms match the actual stock totals recorded within the system. This basically allows your organization to verify the accuracy of your inventory quantities and values.

Click **Generate Count List** to enable the system to begin the count. The system uses the **Days Since Last Inventory** parameter, if available, or the **Creation Date** parameter to ensure that the items that have not been counted in a while are retrieved first and displays them at the top of the list. The list is then sorted with the **Physical Count Master/Line Info** as the first column, sorted first by **Primary Bin** and then by **Stock Item** code. If actual counts are found to be out of sync with system inventory counts, they can be reconciled based on pre-defined tolerance thresholds.

Storeroom personnel can print a report of the physical inventory items listed for the storeroom. The listing displays the physical inventory number, storeroom, bin, and list of stock items physically available in the storeroom.

ORACLE

PHYSICAL INVENTORY

Physical Inventory: 

2020000000009

Storeroom: SR_Storeroom_TESTPI

Status Date/Time: 08-26-2020 04:44AM

Starting Bin:

Create Date/Time: 08-26-2020 04:25AM

Ending Bin:

Created By: System, English

Bin	Stock Item Code	AUP	UOI	Count	Stock Item Description
	SR_STK1	0.0000	GD Box		SR_STK1
	SR_STK	0.0000	GD Hour		SR_STK

Inventory Adjustments

Inventory adjustments can be used to adjust inventory counts or price of stock items.

When inventory is adjusted, stock transactions are created to adjust the price and quantity of the items being adjusted. Assets that are tracked are adjusted differently in that the unit price is directly adjusted and financial transactions are created if the total inventory value is changed (based on the relative price/quantity adjustment).

Stock Transfers

Stock transfers are used to move stock from one storeroom to another; essentially one store room is ordering stock from another. The ordering storeroom is considered as the receiving storeroom, and the source storeroom is considered as the issuing storeroom. Only items with the same stock type can be transferred.

Receiving Inventory

The receiving process is used to track inventory that is received as the result of a purchase order by recording basic information such as the quantity received, receiving location, packing slip details, carrier, and so on.

When items are delivered to your organization, they must be added into the system inventory by creating a receiving record then accepting the items. The receiving process tracks delivery items through several distinct steps from the time they arrive until they are placed in a storeroom. The steps in the process allow for the actual receipt, inspection of items, and acceptance. The process steps include:

- Record the delivery of items — create a receipt record which holds basic delivery information. This step does not include details about the items.
- Match items to the purchase order — review items on the purchase order and select, from a pick list, the items that were delivered. Items are counted and compared to what was ordered then placed “in receipt”. Items can also be received without a purchase order reference.
- Receive and accept the items — complete the acceptance process to record the items into the appropriate storeroom. The system adjusts inventory quantities for the items accordingly. Depending on your system configuration and business practices, the system may also include taxes and other incidental costs for the items at the time of receipt.
- (alternate) Create a shipping deficiency entry for receipt line items when there is a problem with materials received in the shipment. The deficiencies can be edited, added to, or resolved directly from the **Shipment Deficiency** zone of the **Receipt Line** record.
- (alternate) Return the items — As needed, return items for replacement or credit. When returns are processed, the appropriate costing and inventory adjustments are made to account for the credit.

Quick Receive

Your organization may not require a multi-step processing to receive items into your inventory. In these cases, the system includes a feature to receive and accept the items in one step. This feature is only available if it is enabled in your system implementation. If the stock item has “Accept on Receipt” set as yes, one-step receiving is allowed.

Forced Complete

In some cases, the quantity ordered on the purchase order may never be fulfilled. For example, if your organization ordered a quantity of 10 of items that were discontinued and the vendor only had 5 to send to fulfill the order. The system provides the ability to receive the five items then to force the system to consider the order complete. This processing flags the system that the line item receipt quantity is complete so that the purchase order does not stay open. After receiving partial items, select the **Force Complete** action on the **Receipt** record. Completing this action marks the purchase order line item as **Fully Received** and transitions it to the **Closed** state.

Returns

Returns track inventory that must be returned to a vendor. Returns are tracked in relationship to the purchase order for the original receipt of the items.

Note that this processing is different from a *material return* in that the items are being returned to an outside vendor rather than adjusting quantities in internal storerooms.

About Material Disposition

Asset disposition is a process designed to track repairable assets and stock items. The process tracks the asset or component from when it is removed from service to where it is either retired, scrapped, or repaired and returned to availability for re-use.

Material disposition processing involves creating a work order to remove then replace a broken asset or component. The asset must be flagged as repairable.

In creating the work order and related activities, the resource requirements must indicate that a replacement asset must be checked out from a storeroom to replace the broken one. In order to fulfill these requirements, users must create a material request and material issue for the asset. A **Material Disposition** record is automatically created as a result of these material requirements being fulfilled, and the replacement asset and the broken asset are added on the material disposition as line items. Note that new records are not created directly in the **Material Disposition** portal. They must be generated automatically by this processing.

The person responsible for material disposition should navigate to the **Material Disposition** portal to complete the steps within the system to remove the broken asset and move it to a pending disposition status while moving the replacement asset to an installed status. This is done using the **Complete Exchange** option. When completing the complete exchange process, the system prompts for information such as the actions to perform on the assets, status reason, new storeroom as applicable, and dispositioning dates and times.

The actual assets should also be swapped as applicable toward completing the initial work order and activity.

Your business process may then require that you create a follow up work order to repair the broken asset using the **Create Work Order** action from within the **Material Disposition** portal.

Once a 'repairable' asset has been pulled from the field and repaired through Material Disposition, it can be returned to a storeroom with a new valuation. The stock item detail for the asset will show a repaired price, as applicable.

You can review the costs and other statistics related to the replacement and repair of assets by reviewing the actual **Asset** record in the **Cost Information** section.

Material Requests

Material requests, or checkout requests, manage the process of specifying the need to issue stock items from a storeroom.

When an activity needs stock items, a material request can be created. The request includes a list of all the items that the activity needs from the storeroom with the stock code, the storeroom, and the needed quantity specified. Storeroom personnel can print a report that lists the stock items needed for pickup or delivery. The report includes the material request number, storeroom, and delivery locations. Also, stock items are displayed with barcodes, if applicable.

Material Request: 

20200000000015

Status: Created

Storeroom: 

Allmond Warehouse, 4801 Allmond Ave, Louisville, KY, 40214, US

Work Order / Activity:

Project:

Request Information:

**Required By
Date:** 08-10-2020**Requested By:** Kuchibhotla, Asrita**Deliver To Location:****Phone:****Deliver To:****Description:**Remarks:

The following processing must be completed:

- Create the material request
- Add line items for the stock items to be issued. Line items can be added manually or from a material request template.
- Prepare the material request for issue
- Issue the materials (Materials may be issued fully or partially)

Material Issue

Material issue, or stock checkout, manages the process of pulling and releasing stock items from a storeroom.

Material issue processing occurs as follows:

- The person from the issuing storeroom can verify availability within the system then release (issue) the items. The material issue is processed using the originating **Material Request** record.
- The system calculates the inventory count to decrease it accordingly.
- Issued items are be charged to the activity.

It is also possible to issue materials without a relationship to an activity. In this case the user must provide the accounting information that will be charged for the materials

Material Returns

Material returns manage the process of returning excess stock items that were previously issued to the storeroom.

Material return processing occurs as follows:

- A material return is created by navigating to the related material request and clicking the Return button for the relevant line item.
This can also be done by creating a new record in the **Material Return** portal which references the originating material request.
- Warehouse personnel check the quality of the stock item being returned. If it is in good condition, the item is accepted and returned to the storeroom.
- The system calculates the inventory count to increase it accordingly.
- Returned items are credited back to an account or work order activity.

Material Request Templates

A material request template is a collection of stock items that is used to easily populate a material request. For example, a template could exist for “office supplies” that contains the collection of office supply stock items that the user can add to the request. Likewise, you might have a template to request all of the items on a bill of materials.

- Each storeroom can have multiple templates.
- Only stock items in inventory (both Lot Managed and Not-Lot Managed) and tracked assets can be specified on a template. The stock items must have a stock item detail at the storeroom.
- Direct purchase stock items cannot be referenced on a material request template.
- An expense code, if populated on the material request template, overrides the expense code from the Stock Item Detail and will be populated on the request.
- A default count can be included on the template to indicate the default request quantity of the stock item. For example, your organization might always request the same amount of the stock item. This can be specified on the template.

After a material request is created, the line items can be added by referencing the template then selecting items from a picklist.

Bill of Materials

Bills of materials are used to simplify the identification of materials when planning work.

A Bill of Material (BOM) is a list containing a hierarchy of component parts that make up a piece of equipment or asset. The component parts, also known as BOM Parts, are the individual stock items or an assembly of stock items included in the BOM. An example of a BOM might be an Apple iMac computer. The BOM parts identified for this particular BOM might include a Monitor, Keyboard, mouse, and so on. Another example might be a BOM for a carburetor. The BOM parts included on the BOM might be a carburetor rebuild kit.

Your organization can build a BOM or define a BOM part and reference it on an asset, component, or fleet asset (vehicles). The BOM or BOM part can then be used as search criteria when searching for an asset.

A bill of materials can be a great advantage for maintenance personnel when they are planning or working on an asset. The bill of materials can be used to easily find the stock items related to component parts of the assets that need to be worked on. This relieves planners from doing extensive research to find the correct related stocked and non-stocked items needed.

As needed, you have the ability to add materials to activities from the asset's Bill of Materials. Selecting the add from BOM action opens a dialog where you can choose items from the BOM pick list.

Bills of materials can be multilevel, with a list of components and lists of parts for the components, lists of parts that make up the parts, and so on. The level of detail you build into them depends on your organization's resources and needs.

A Bill of Materials is similar to a **Specification** in that the same Bill of Material can be created as one source for specific criteria needed to maintain the asset, which is then referenced on many assets.

Bills of materials are classified in the following ways:

- **General Stock Item Part** — Defines a general piece of stock or equipment. A generic stock item BOM is automatically generated by the system when a stock item is created.
- **Independent Part** — Defines a generic part which is not a stock item, but can be composed of many parts. For example, a truck would be defined as an asset, not a stock item, but might reference a BOM with independent parts. These must be created manually.

Access the **Bill of Materials** tab on applicable stock items to review the associated **Bill of Materials** tree or the **BOM Part** record for the stock item.

About Shipping Memos

Shipping memo is the record of stock items being shipped from an internal storeroom, whether from storeroom to storeroom or from storeroom to vendor.

Shipping memos can be created for both standard and tracked stock items and enables storeroom managers and shipping clerks a way to manage and track stock items being shipped between locations.

Some uses for shipping memo include the following:

- Tracking stock items returned to vendors for credit, replacement, or repair
- Help manage the return of repaired or replacement stock items back into Inventory

There are four categories of shipping memos:

- **General** is the default shipping memo category that should be used when creating a shipping memo manually.
- **Repair** is used when returning stock items from a purchase order needing reconditioning or repair to a vendor.
- **Return** is used when returning stock items from a purchase order to a vendor for credit or replacement.
- **Transfer** is used when creating a Stores Transfer (transferring stock items between storerooms) to reference on the shipping memo.

Shipping memos can be created from various places in the system, such as Menu, Purchase Order, Return, Stock Transfer, and so on.

Maintaining Stock Items

The **Stock Item** portal is used to display and maintain a stock item record.

Refer to [About Inventory](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Stock Item**. You are brought to a query portal with options for searching for a specific stock item. Once a stock item has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Stock Item**: This zone displays information about the selected stock item.

- **Stock Item Availability:** This zone enables broadcasting of stock item availability per storeroom for the stock item record.
- **Unlink Specification:** This zone removes the specification from the stock item record.
- **Link Specification:** This zone links the specification to the stock item record which enables the stock item to be used to track the associated asset.
- **Attachments:** This zone enables users to manage all types of attachments for the stock item record.
- **Standard Notes:** This zone lists the standard notes associated with the stock item record and enables the viewing of standard note information.
- **Notes:** This zone displays any notes associated with the stock item record.
- **Referencing Compatible Units:** This zone lists the compatible units that reference the current compatible unit.

The following zones may appear as part of the portal's **Bill of Materials** tab page:

- **BOM Tree:** This zone displays a hierarchy tree view of the stock item.
- **Bill Of Materials:** This zone displays information about the bill of material associated with the selected stock item record.
- **Referencing BOM's:** This zone enables broadcasting of referenced parts for the stock item.
- **Attachments:** This zone enables users to manage all types of attachments for the stock item record.
- **Notes:** This zone displays any notes associated with the BOM associated with this stock item record.

Maintaining Stock Item Details

The **Stock Item Detail** portal is used to display and maintain a stock item detail record.

Refer to [About Inventory](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Stock Item Detail**. You are brought to a query portal with options for searching for a specific stock item detail. Once a stock item detail has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Stock Item Detail:** This zone displays information, such as purchase details, reorder information, and buyer information, for the selected stock item detail record.
- **Inventory Adjustment:** This zone lists adjustments to inventory, such as price or quantity, for this stock item detail.
- **Attachments:** This zone enables users to manage all types of attachments for the stock item detail record.
- **Notes:** This zone displays any notes associated with the stock item detail record.

The following zones may appear as part of the portal's **Availability** tab page:

- **Stock Item Availability:** This zone displays information, such as purchase details, reorder information, and buyer information, for the selected stock item detail record.
- **Stock Item Inventory:** This zone displays inventory quantity information for the selected stock item detail record.

Maintaining Stock Transfers

The **Stock Transfer** portal is used to display and maintain a stock transfer records.

Refer to [About Inventory](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Stock Transfer**. You are brought to a query portal with options for searching for a specific stock transfer record. Once a stock transfer record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Stock Transfer:** This zone displays information about the selected stock transfer.
- **Stock Transfer Quick Add:** This zone enables the addition of stock to the transfer request.

The following zones may appear as part of the portal's **Stock Transfer Receipt** tab page:

- **Stock Transfer Receipt Header:** This zone displays general receipt header information.

Stock Transaction Log Portal

The **Stock Transaction Log** portal is used to search for stock transactions related to processes that involve inventory stock items.

You can access the portal from **Menu**, select **Inventory**, and then select **Stock Transaction Log**. You are brought to a query portal with options for searching for a specific stock transaction log. Once a record has been selected, you are brought to the transaction log for the stock item where you can view financial transactions and other details for the selected record.

The zones on the portal's **Main** tab page will vary according to the related process.

The following zone appears as part of the portal's **Financial Transaction** tab page:

- **Financial Transaction:** This zone displays details about the financial transaction for this stock item transaction.

Refer to [About Inventory](#) for more information.

Maintaining Physical Inventories

The Physical Inventory portal is used to display and maintain a physical inventory record.

Refer to [About Inventory](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Physical Inventory**. You are brought to a query portal with options for searching for a specific record. Once a record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Physical Inventory Header:** This zone displays information about the selected physical inventory record.
- **First Count Information:** This zone displays the inventory count records that have been generated.
- **Recount Information:** This zone displays the list of Physical Inventory Count records along with additional fields for entering the Recount value and a First Count field displaying the First Count value.
- **Completed Count Information:** This zone displays the list of completed Physical Inventory Count records.

Maintaining Receipts

The **Receipt** portal is used to display and maintain receipts.

Refer to [About Inventory](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Receiving — Receipt**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Receipt Header:** This zone provides information about the receipt, such as receipt information, purchase order information, additional processes, and a receipt line status summary.
- **Open Receipt Lines:** This zone enables a search for open receipt lines filtered by storeroom and resource type.
- **In Receipt Lines:** This zone displays the in receipt lines in a table format. Users are also able to search by storeroom and resource type to filter the results.
- **Accept/Return Detail:** This zone displays the receipt line information that has been accepted or returned. Receipt lines also display deficient in red text in the info string if the receipt line includes a deficiency.
- **Purchase Order Line Activity:** This zone displays the purchase order lines and receipt lines associated with the purchase order line activity.
- **Notes:** This zone lists the notes associated with this entity.
- **Attachments:** This zone lists the attachments associated with this entity.

Maintaining Shipping Memos

The **Shipping Memo** portal is used to display and maintain shipping memos.

Refer to [About Shipping Memos](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Shipping Memo**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Shipping Memo:** This zone provides information about the shipping memo, such as shipping category, shipping memo number, ship to details, and tracking information.
- **Shipping Lines:** This zone provides links to the shipping line and stock items associated with the shipping memo.
- **Attachments:** This zone enables users to manage all types of attachments for the shipping memo record.
- **Notes:** This zone displays any notes associated with this shipping memo record.

Maintaining Returns

The **Return** portal is used to display and maintain receipt return records.

Refer to [About Inventory](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Receiving — Return**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Return:** This zone provides information about the return record, such as purchase order, stock item detail, type of return, and the quantity.
- **Shipping Memos:** This zone displays any shipping memos associated with the return record.

The following zones may appear as part of the portal's **Financial Transaction** tab page:.

- **Financial Transaction:** This zone provides information about the shipping memo, such as shipping category, shipping memo number, ship to details, and tracking information.

Maintaining Material Issue

The **Material Issue** portal is used to display and maintain material issue records.

Refer to [About Material Disposition](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Material Issue**. You are brought to a query portal with options for searching for a material issue record. The query search displays a list of material requests. Once a material request record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

Maintaining Material Requests

The **Material Request** portal is used to display and maintain a material request record.

Refer to [About Material Disposition](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Material Request**. You are brought to a query portal with options for searching for a specific material request record. Once the record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Material Request Header:** This zone displays information about the selected material request record.
- **Material Request Quick Add:** This zone enables the addition of stock items to the material request.
- **Open Request Lines:** This zone displays the open request lines associated with the material request.
- **Stock Item Inventory:** This zone displays stock item inventory information after broadcasting an open request line.
- **Stock Item Availability:** This zone displays upon broadcasting an open request line and displays stock item information per storeroom.

Maintaining Material Request Templates

The **Material Request Template** portal is used to display and maintain a material request template record.

Refer to [About Material Disposition](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Material Request Template**. You are brought to a query portal with options for searching for a specific material request template. Once the template has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Material Request Template Display:** This zone displays information about the selected material request template record.

Maintaining Material Returns

The **Return** portal is used to display and maintain receipt return records.

Refer to [About Material Disposition](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Material Return**. You are brought to a query portal with options for searching for a material return record. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Return:** This zone displays information about the material return.
- **Material Return Quick Add:** This zone enables the addition of material to be returned.
- **Open Return Lines:** This zone displays the open return lines associated with the material return.
- **Closed Return Lines:** This zone displays the closed return lines associated with the material return.
- **Stock Item Inventory:** This zone displays stock item inventory information after broadcasting an open return line.
- **Stock Item Availability:** This zone displays upon broadcasting an open return line and displays stock item inventory information per storeroom.

Maintaining Material Dispositions

The **Material Disposition** portal is used to display and maintain material disposition records.

Refer to [About Material Dispositions](#) for more information.

You can access the portal from **Menu**, select **Work Management**, then select **Material Disposition**. You are brought to a query portal with options for searching for a material disposition record. Once a record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Material Disposition:** This zone displays asset disposition information.

Maintaining Bill of Materials

The **Bill of Materials** portal is used to display and maintain a bill of material record.

Refer to [About Material Disposition](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, then select **Bill of Materials**. You are brought to a query portal with options for searching for a specific bill of material. Once a bill of material has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **BOM Tree:** This zone displays a hierarchy view of the selected BOM and BOM parts.
- **Bill of Materials:** This zone displays information about the bill of material record.
- **Referencing BOM's:** This zone lists the BOMs that reference this particular BOM record.
- **Attachments:** This zone enables users to manage all types of attachments for the BOM record.
- **Notes:** This zone displays any notes associated with the bill of material record.

Maintaining BOM Parts

The **BOM Part** portal is used to display and maintain a bill of material parts record.

Refer to [About Material Disposition](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, then select **BOM Part**. You are brought to a query portal with options for searching for a record. Once a record has been selected, you are brought to the **Bill of Materials** tab on the **Stock Item** record to view and maintain the selected record.

The following zones may appear as part of the portal's **Bill of Materials** tab page:

- **Bill of Materials:** This zone displays information about the component parts for the particular stock item.
- **Referencing BOM's:** This zone lists the BOMs that reference this particular BOM part.
- **Attachments:** This zone enables users to manage all types of attachments for the BOM part.
- **Notes:** This zone displays any notes associated with the BOM part record.

Maintaining Reorder Reviews

The **Reorder Review** portal is used to display and maintain a reorder review record.

Refer to [About Inventory](#) for more information.

You can access the portal from **Menu**, select **Inventory**, then select **Reorder Review**. You are brought to a query portal with options for searching for a specific reorder review. Once a reorder review has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Reorder Review:** This zone displays information, such as reorder details and vendor information, for the selected reorder review record.

The following zones may appear as part of the portal's **Availability** tab page:

- **Stock Item Inventory:** This zone displays information about inventory quantity for the selected reorder review record.
- **Stock Item Availability:** This zone enables broadcasting of stock item availability per storeroom for the reorder review record.
- **Storeroom Alternative Stock Items:** This zone displays once a storeroom is broadcast from the **Stock Item Availability** zone and provides a list of alternate stock items.
- **Storeroom Stock Item Lots:** This zone displays once a storeroom, having inventory lot managed records, is broadcast from the **Stock Item Availability** zone. The zone displays the expiration date, lot stock item detail, and inventory quantity for the corresponding lot records.
- **Material Issue History Summary:** This zone displays a summary of all the material issued for the current stock item.

Reorder Review Mass Approval

The **Reorder Review Mass Approval** portal is used to search for reorder review records ready for approval.

You can access the portal from **Menu**, select **Inventory**, and then select **Reorder Review Mass Approval**. You are brought directly to the **Reorder Review Mass Approval** page.

The portal's **Main** tab page enables users to filter search criteria by Purchase Commodity, Buyer, or Create Date Range. User can select any number of reorder review records and click **Approve** to mass approve the records.

Refer to [About Inventory](#) for more information.

Chapter 8

Purchasing

About Purchasing

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.

Manufacturers

Use the **Manufacturer** portal to define attributes for product manufacturers.

The use of this portal depends heavily upon your business practices, however, in general the manufacturers listed can be associated to specifications which can then be included on assets or components to provide additional reference information related to those objects. The manufacturer usually provides specification information for assets and components.

Manufacturer Specifications defines a list of asset specifications that belong to the manufacturer.

A manufacturer might also be referenced on regular stocked inventory.

Vendors

Vendors are responsible for providing goods and services to your company.

Use the **Vendor** portal to maintain information about vendors, including their capabilities, payment terms, delivery terms, and various vendor sites. The vendor location distinguishes between multiple sites for one vendor.

Vendor Locations

One or many vendor locations are defined for each vendor to distinguish between multiple vendor sites. Each location provides more detailed information about the vendor site such as address, capabilities, payment details, contacts, and so on.

For example, it may be valuable to understand which location is shipping the goods that you have ordered from the vendor. Also, some vendors might ship from one site, but your payment terms are set with an alternate site. Capabilities would provide information such as when one location offers pick up at that location whereas another location may not.

When a vendor has multiple locations, alternate locations are listed in the **Vendor Locations** zone on the vendor record.

Purchase Orders

Purchase orders are contracts placed with vendors for the purchase of materials or service. Purchase order records contain general information about the purchase including the requestor, when items are required, the vendor, and more.

Details about each item or service are entered and maintained on purchase order line items. Your system configuration determines which stock items, services, and other types of items can be added as line items, how tax rates are handled for each line item, and so on.

Additional processing and record actions on the purchase order allows you to print the purchase order record, create a new revision, create a shipping memo, issue the purchase order to the vendor, expedite the purchase order, complete a **Revision to the PO**, and submit the purchase order for approval. After the purchase order is approved within your organization and issued to the vendor, next steps involve waiting for the items to be received, then receiving them into the system. Refer to [About Receiving](#) for more information.

When changes are made to the purchase order, the document may or may not need to re-enter the approval process depending on your business practices and system settings. Your system may also be set so that the purchase order will transition directly to the approved or issued state, for instance, if the order was created by batch stock reorder. Note that only certain fields can be edited on an issued purchase order. If further updates are needed, such as changes that affect the pricing or quantity ordered, you have to create a new revision and resubmit to the vendor. Refer to [Understanding Purchasing](#) for more information.

Purchase orders affect the **On Order** quantity for stock.

Common purchasing terms indicated on purchase order and purchase order line items include:

- **Unit of Purchase:** The unit of purchase for the item, such as “each”, “box”, “gallons”, or “cubic feet”..
- **Unit of Issue:** The way in which the item is issued in your storerooms.
- **Purchase to Issue Ratio:** The difference between how you buy the item and how you issue it. You may, for instance, issue in individual items but order the items by the box where one box would contain 10 individual items. The **Unit of Purchase** would then be *box*, the **Unit of Issue** would be *each*, and the **P/I Ratio** would be *10*. The system uses the **Unit of Purchase**, **Unit of Issue**, and **Purchase to Issue Ratio** information to maintain quantities as parts are issued and new parts are received.

Purchase Requisitions

Purchase requisitions are used to store and track requests for material resources and services.

Requests can be created manually or by the system for the following situations:

- For work orders that require materials or services
- By the batch stock reorder process when the number of items fall below the reorder point
- Manually to replenish stock

After the purchase requisition is reviewed and approved, it can be transitioned to a purchase order to initiate the actual purchase with the vendor. Requisitions can also be used to create blanket contracts.

While purchase orders can be created without a purchase requisition, the requisition can be used according to your business practices to formalize the purchasing process or to create a more stringent chain of approvals for auditing or other purposes.

Purchase requisitions affect **Pending Order** quantity for stock.

Blanket Contracts

A blanket contract is a purchasing agreement that your organization makes with a vendor to pre-plan the purchase of materials or services. These contracts typically contain multiple delivery dates over a period of time, often negotiated to take advantage of predetermined pricing.

Blanket contracts are also often used when there is a recurring need for expendable goods or when your organization buys large quantities from one vendor. Based on the purchase request or purchase order, invoice items can be created as needed until the contract is fulfilled.

As long as the blanket contract is in an *Active* status, it can be referenced by new purchasing documents to order materials or services. Contracts can be set to a *Suspended* status so that they are temporarily unavailable for new purchases. This might be used while re-negotiating terms with a vendor, if you are researching new vendors for a particular item or service, or for other business requirements.

The following sections further describe concepts related to blanket contract management.

Blanket Contract Access Lists

Blanket contract access lists identify which users have the authority to create charges against a blanket contract. The list also establishes a maximum dollar amount that the user is authorized to charge for a single purchase order.

When a blanket contract is created, it references this list to provide the reference of authorized users. When a blanket contract is being applied toward creating a purchase order, the system verifies that the user is valid based on this list and that the person's purchasing limit is not exceeded.


Your system can be configured with lists of users who are authorized to issue purchasing documents against a blanket contract. Settings on the blanket contract determine whether or not the contract is restricted to a particular access list, which list of users has access or, if no access list is indicated, a default dollar value limit for the contract. When an access list is in use, a limit can be set for each individual user.

Blanket Contract Usage

Spending against each blanket contract is tracked using the **Limit Used** indicator on the **Activity and Usage** tab. When a new blanket contract revision is created, these values reset for the new revision and are tracked as new.

Vendor Bid Download

Vendor bid download can be used as a way to request new bids for vendor materials and services prior to the expiration of the blanket contract.

Users can click on the **Explorer Zone Menu**  icon and select **Export to Excel** to retrieve an excel file listing the current bid on materials or services. This file can then be sent out to have vendors populate the "Quoted" columns with new bid information. When the bids are returned, they would need to be manually assessed and the winning quotes uploaded, by selecting the "Upload Lines From File" hyperlink in the **Additional Process** section, to a new BC Header to create the line items. See [Creating CSV Files](#) for more information about creating and formatting csv files.

Invoices

Invoices are used for entering and processing bills for materials or services. They ensure that the quantities being billed do not exceed what has been received and that the prices at which they are billed match what is on the originating purchase order.

The invoice also specifies payment information such as the purchase order number, the vendor, any discounts that have been applied and so on. Cost allocation information indicates the taxes, freight, and any other additional costs or discounts applied to the invoice.

Line Items

Invoices specify a number of line items which may originate from a number of purchase order line items. Invoice line items must conform to the following rules:

- If the invoice header references a purchase order, then all purchase order lines added to the invoice must reference that same purchase order.
- If the invoice header does not reference a purchase order, then multiple invoice line items can be added to the invoice using the **Invoice Line Item** zone.
- If the invoice header does not reference a purchase order because it references multiple purchase orders, the purchase order line items can be added for each purchase order using the **Purchase Order Line Items** zone on the invoice record for each of the invoice lines.
 - If an invoice references a purchase order, the total invoiced for a purchase order line item cannot exceed the receipt quantity. If the invoice quantity is more than the net received, the invoice goes into “pending resolution” state.
 - If a unit price is not the same as last posted invoice price, if any, invoice will also go into pending resolution.
 - Also, when the invoice amount does not match the PO amount within a certain range defined in master configuration, the user will see a warning or error (also based on setting in master configuration).

The invoice amount is calculated as product of quantity and unit price. Unit price is locked down to latest posted invoice if there is one, otherwise it comes from the purchase order. If master configuration settings trigger an error, the invoice will go into pending resolution status. The tolerance value indicated in Purchasing Master Configuration regulates this behavior.

Invoice Groups

Invoice groups can be used to group invoices together before they are sent to Accounts Payables (A/P). They can be used as an intermediate step to review a set of invoices before those invoices are processed for payment. If no invoice group is configured on the invoice header, individual invoices post as soon as they are approved.

Automatic Invoicing

In order for the system to complete automatic invoicing against purchase orders, the vendor on the purchase order must be set as an “auto pay” vendor. When the purchase order is created, it must also be set as “auto pay”. After items are received against the purchase order, the system automatically creates an invoice with line items. When an invoice is automatically created, no additional line items can be added to the created invoice and none of the line items can be deleted. If the receipt included more items received than ordered, these invoice line items are placed in *Pending Resolution* status when the invoice is submitted for approval. An authorized user must then change the quantity on the purchase order before the invoice can be reprocessed successfully.

Related Financial Transactions

All invoice line items result in a financial transaction. As the financial transactions are created they zero out the clearing account and make up differences, if any, if information such as the price or tax amount on the invoice is different from the purchase order.

Converted Total and Exchange Rates

Note that if your organization works with foreign currencies, there may be a discrepancy shown in the **Invoice Summary** section on the Invoice header between the converted total on the invoice header and the converted total calculated by totaling the line items. The converted total is derived by multiplying the invoice total by the exchange rate. This header value might be off by \$.01 or more due to the slight offset that results from rounding each individual line item then calculating the header converted total. You can see a more accurate converted total by reviewing each line item individually.

Expedite Purchase Orders

The expedite order process helps buyers and other purchasers to ensure that required goods arrive at the appointed date in the agreed quality at the agreed location.

The **Expedite Order** portal is used to track purchase order items that have not yet been received. It allows users to modify select information on an issued purchase order without needing to create and process a **Revision to the PO**. It is typically used when locations and delivery dates need to be altered, not when there are changes that would affect the cost of items or the total amount for the purchase order.

Maintaining Purchase Orders

The **Purchase Order** portal is used to display and maintain a purchase order record.

Refer to [About Purchasing](#) for more information.

You can access the portal from **Menu**, select **Purchasing**, then select **Purchase Order**. You are brought to a query portal with options for searching for a specific purchase order. Once a purchase order has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Purchase Order:** This zone displays information about the purchase order, such as cost information, order line item information, and accounting information.
- **Purchase Order Lines:** This zone lists the purchase order line items associated with the purchase order record.
- **Blanket Contract Lines:** If the purchase order is associated with a blanket contract, this zone lists lines from that contract that are not already included in the purchase order. You may use this zone to add additional lines to the purchase order. As such, the zone is only visible while lines can still be added.
- **Purchase Order Transactions:** This zone lists all transactions for all revisions of the purchase order record.
- **Related Purchase Requisitions:** This zone lists purchase requisitions related to the purchase order record.
- **Notes:** This zone displays any notes associated with the purchase order record.
- **Attachments:** This zone enables users to manage all types of attachments for the purchase order record.

The following zones may appear as part of the portal's **Expedite** tab page:

- **Purchase Order:** This zone displays information about the purchase order.
- **Expedite Activity:** This zone lists the expedited activities associated with the purchase order.

The following zones may appear as part of the portal's **Revision History** tab page:

- **Purchase Order:** This zone displays revision history information about the purchase order.

The following zones may appear as part of the portal's **Log** tab page:

- **Purchase Order Header Log:** This zone displays log details for the purchase order.
- **Approval Request History:** This zone displays approval history for the purchase order.

Maintaining Purchase Order Lines

The **Purchase Order Line** portal is used to display and maintain a purchase order line record.

Refer to [About Purchasing](#) for more information.

You can access the portal from the **Purchase Order** portal when reviewing a specific line.

The following zones may appear as part of the portal's **Main** tab page:

- **Purchase Order Line:** This zone displays information about the purchase order line.
- **Related Purchase Order Lines:** This zone displays other lines from the purchase order.
- **Attachments:** This zone displays attachments related to this purchase order line record.
- **Standard Notes:** This zone displays standard notes related to this purchase order line record.
- **Notes:** This zone displays additional notes related to this purchase order line record.

The following zones may appear as part of the portal's **Expedite** tab page:

- **Purchase Order Line:** This zone displays information about the expedited purchase order line.
- **Expedite Activity:** This zone displays the expedited activity information associated with the expedited purchase order line.

The following zones may appear as part of the portal's **Log** tab page:

- **Purchase Order Line Log:** This zone displays log details for the expedited purchase order line.
- **Related Stock Transactions:** This zone displays the related stock transaction information for the purchase order line.

Maintaining Purchase Requisitions

The **Purchase Requisition** portal is used to display and maintain a purchase requisition record.

Refer to [About Purchasing](#) for more information.

You can access the portal from **Menu**, select **Purchasing**, then select **Purchase Requisition**. You are brought to a query portal with options for searching for a specific record. Once a purchase requisition has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Purchase Requisition:** This zone displays information about the purchase requisition, such as vendor information, requisition information, and accounting information.
- **Open Purchase Requisition Lines:** This zone lists all the open purchase requisition lines.
- **Blanket Contract Lines:** If the purchase requisition is associated with a blanket contract, this zone lists lines from that contract that are not already included in the purchase requisition. You may use this zone to add additional lines to the purchase requisition. As such, the zone is only visible while lines can still be added.
- **Closed Purchase Requisition Lines:** This zone lists all the closed purchase requisition lines.
- **Related Purchase Orders:** This zone lists the related purchase orders.
- **Purchase Requisition Header Notes:** This zone displays any notes associated with the purchase requisition record.
- **Notes:** This zone displays any notes associated with the purchase requisition.
- **Attachments:** This zone enables users to manage all types of attachments for the purchase requisition record.

The following zones may appear as part of the portal's **Log** tab page:

- **Purchase Requisition Log:** This zone displays log details for the purchase requisition.
- **Approval Request History:** This zone displays approval request histories for the purchase requisition.

Maintaining Purchase Requisition Lines

The **Purchase Requisition Line** portal is used to display and maintain a purchase requisition line record.

Refer to [About Purchasing](#) for more information.

You can access the portal from the **Purchase Requisition** portal when reviewing a specific line.

The following zones may appear as part of the portal's **Main** tab page:

- **Purchase Requisition Line:** This zone displays information about the purchase requisition line.
- **Tax Calculation Details Info:** This zone displays tax information related to the purchase requisition line.
- **Related Purchase Requisition Lines:** This zone displays other lines from the purchase requisition.
- **Attachments:** This zone displays attachments related to this purchase requisition line record.
- **Standard Notes:** This zone displays standard notes related to this purchase requisition line record.
- **Purchase Requisition Line Notes:** This zone displays additional notes related to this purchase requisition line record.

Maintaining Expedite Purchase Orders

The **Expedite Purchase Order** portal is used to display and maintain expedited purchase order records.

Refer to [About Purchasing](#) for more information.

You can access the portal from **Menu**, select **Purchasing**, then select **Expedite Purchase Order**. You are brought to a query portal with options for searching for a specific record. Once an expedited purchase order has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Expedite:** This zone displays expedite information associated with the purchase order.

Maintaining Blanket Contracts

The **Blanket Contract** portal is used to display and maintain a blanket contract record.

Refer to [About Purchasing](#) for more information.

You can access the portal from **Menu**, select **Purchasing**, then select **Blanket Contract**. You are brought to a query portal with options for searching for a specific record. Once a blanket contract has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Blanket Contract Header:** This zone displays information about the blanket contract, such as contract and vendor information.
- **Blanket Contract Lines:** This zone lists the blanket contract lines associated with the blanket contract record.
- **Blanket Contract Header Notes:** This zone displays any notes associated with the blanket contract record.
- **Attachments:** This zone enables users to manage all types of attachments for the blanket contract record.

The following zones may appear as part of the portal's **Activity and Usage** tab page:

- **Blanket Contract Header:** This zone displays information about the blanket contract.

- **Blanket Contract Activity:** This zone lists the purchasing activities associated with the blanket contract.
- **Blanket Contract Usage Summary:** This zone displays a summary list of orders associated with the blanket contract.

The following zones may appear as part of the portal's **Revision History** tab page:

- **Blanket Contract Header:** This zone displays information about the blanket contract.
- **Blanket Contract Header Revision History:** This zone lists all revisions associated with the blanket contract.

The following zone appears as part of the portal's **Vendor Bid Download** tab page:

- **Blanket Contract Lines Download:** This zone allows you to download blanket contract lines information.

Maintaining Blanket Contract Lines

The **Blanket Contract Line** portal is used to display and maintain a blanket contract line record.

Refer to [About Purchasing](#) for more information.

You can access the portal from the **Blanket Contract** portal when reviewing a specific line.

The following zones may appear as part of the portal's **Main** tab page:

- **Blanket Contract Line:** This zone lists the blanket contract lines for the blanket contract. This zone provides multi-select capability for blanket contract lines, creation of purchase requests and purchase orders, and re-sequencing of blanket contract lines.
- **Attachments:** This zone displays attachments related to this blanket contract line record.
- **Standard Notes:** This zone displays standard notes related to this blanket contract line record.

Maintaining Invoices

The **Invoice** portal is used to display and maintain an invoice record.

Refer to [About Purchasing](#) for more information.

You can access the portal from **Menu**, select **Purchasing**, then select **Invoice**. You are brought to a query portal with options for searching for a specific record. Once an invoice has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Invoice Header:** This zone displays information about the invoice, such as payment information, cost allocation, and tax information.
- **Invoice Lines:** This zone lists the invoice lines associated with the invoice record.
- **Notes:** This zone displays any notes associated with the invoice record.
- **Attachments:** This zone enables users to manage all types of attachments for the invoice record.

The following zones may appear as part of the portal's **Financial Transaction** tab page:

- **Invoice Header Financial Transaction:** This zone lists the financial transactions associated with the invoice record.

Maintaining Invoice Groups

The **Invoice Group** portal is used to display and maintain an invoice group record.

Refer to [About Purchasing](#) for more information.

You can access the portal from **Menu**, select **Purchasing**, then select **Invoice Group**. You are brought to a query portal with options for searching for a specific record. Once an invoice group has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Invoice Group:** This zone displays information about the invoice group.
- **Invoices:** This zone lists the invoices associated with the invoice group record.
- **Tag Invoices:** This zone enables the user to tag invoices to this invoice group.

Maintaining Manufacturers

The **Manufacturer** portal is used to display and maintain a manufacturer record.

Refer to [About Purchasing](#) for more information.

You can access the portal from **Menu**, select **Asset Management**, then select **Manufacturer**. You are brought to a query portal with options for searching for a specific manufacturer. Once a manufacturer has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Manufacturer:** This zone displays information about the manufacturer.
- **Manufacturer Specifications:** This zone lists the specifications associated with the manufacturer record.

Maintaining Vendors

The **Vendor** portal is used to display and maintain a vendor record.

Refer to [About Purchasing](#) for more information.

You can access the portal from **Menu**, select **Purchasing**, then select **Vendor**. You are brought to a query portal with options for searching for a specific vendor. Once a vendor has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Vendor:** This zone displays information about the vendor.
- **Vendor Location:** This zone lists the locations associated with the vendor record.

The following zones may appear as part of the portal's **Blanket Contracts** tab page:

- **Vendor Active Blanket Contracts:** This zone lists the blanket contract for this vendor record.

The following zones may appear as part of the portal's **Open Purchase Requisitions** tab page:

- **Vendor Open Purchase Requisition:** This zone displays the open purchase requisitions associated with this vendor record.

The following zones may appear as part of the portal's **Purchase History** tab page:

- **Vendor Purchase Order History Summary:** This zone displays a summary of purchase orders associated with this vendor record.

Maintaining Vendor Locations

The **Vendor Location** portal is used to display and maintain a vendor location record.

Refer to [About Purchasing](#) for more information.

You can access the portal from **Menu**, select **Purchasing**, then select **Vendor**. You are brought to a query portal with options for searching for a specific vendor. Once on the **Vendor** portal, scroll to the **Vendor Locations** zone and select the appropriate vendor location hyperlink where you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Vendor Location:** This zone displays general vendor location information as well as location, purchasing, payment, and pay to vendor details.
- **Notes:** This zone displays any notes associated with the vendor location record.
- **Attachments:** This zone enables users to manage all types of attachments for the vendor location record.
- **Related Vendor Location:** This zone displays all vendor locations associated with the current vendor location record.

The following zones may appear as part of the portal's **Blanket Contracts** tab page:

- **Vendor Active Blanket Contracts:** This zone lists the active blanket contracts associated with the current vendor location record.

The following zones may appear as part of the portal's **Open Purchase Requisitions** tab page:

- **Vendor Open Purchase Requisition:** This zone displays the open purchase requisitions associated with this vendor location record.

The following zones may appear as part of the portal's **Purchase History** tab page:

- **Vendor Purchase Order History Summary:** This zone displays a summary of purchase orders associated with this vendor location record.

The following zones may appear as part of the portal's **Shipping Deficiencies** tab page:

- **Shipment Deficiencies:** This zone lists the shipment deficiencies associated with this vendor location record.

The following zones may appear as part of the portal's **Commodities** tab page:

- **Vendor Location Commodities:** This zone lists the commodities associated with the current vendor location record.

Chapter 9

Service History

About Service History

Service history is information regarding some type of service or maintenance performed on an asset. Information typically associated with service history include record inspection feedback, pass/fail details, downtime, parts failure information, maintenance or service logs, or other information regarding service on the asset.

Service history can create follow-up actions, such as work orders or work activities, when work for an originating issue needs to be monitored. An example of follow-up activities to an originating gas leak call might be to monitor, repair, and to verify repairs were made.

Service history can also create follow-up actions to stop or cancel follow-up activities once the problem has been fixed or the severity level has changed.

Downtime

Downtime is a period of time when an asset or item is not in active service. Downtime can occur as a result of an unexpected asset failures or downtime can be planned as with assets taken out of service for planned upgrades, repairs, or routine maintenance.

Failures

Failures describe an asset or item that is no longer working as designed. For example, a relief valve that does not close and blows all the time would be considered as having failed. Failed assets or items can either be exchanged for a properly functioning replacement or repaired on site.

Inspections

Inspections of assets can occur throughout an asset's lifecycle; when the asset is received from the manufacturer, when alert is raised for failing high or low readings, routine inspections, and so on.

Inspections are performed to assess the condition of the asset.

Refer to [Condition Assessments](#) for more information.

Service history is used to record inspection events for the asset.

Condition Assessments

Service history questionnaires are created to address typical inspection points, concerns, or issues based on the kind of asset, type of inspection, and so on, and is used by the system to assess the condition of the asset.

See [Questionnaires](#) for more information.

Technicians would enter answers to the predefined questions on the service history record. Answers are typically chosen from a set of pre-defined values that are configured when the questions are designed for the assessment. **Condition Assessment Groups** and **Condition Assessment Classes** are used to 'bundle' the various kinds of inspections that are available for an asset. All of the data that is collected is used to calculate a final condition score.

Condition scores are on a 1–5 scale where 1 is Very Good and 5 is Very Poor. Also, scores have an associated “confidence rating” which drops as the input data ages. This is based on the premise that the older the data, the less reliable it is.

Refer to [Understanding Asset Types](#) in the **Administrative User Guide** for more information about **Condition Assessment Classes** and **Condition Assessment Groups**.

There are four kinds of condition assessment types that can be performed:

- **Percent of New** (maps a percent value to the 1-5 scale): The condition of the asset is based on passing or failing characteristics of a new asset. A “like new” asset would pass all characteristics and be awarded the maximum points resulting in a 100% rating. A severely broken asset would fail all characteristics of a new asset and be awarded zero points resulting in a 0% rating.
- **General Weighted**: The condition of the asset is based on a general assessment of potential defects using a weighted scale of 1 - 5. Each potential defect is assigned a weight value based on the impact to the service level. A potential defect that has significant impact on the service level will have a higher weight value than a potential defect with minimal impact on the service level. A “like new” asset would exhibit none of the potential defects and be awarded the minimum points resulting in a rating of 1. A severely broken asset would exhibit the highest severity of each potential defect and would be awarded the maximum points resulting in a rating of 5.
- **Deficiency Scoring** (noting various defects on the asset, which can be summed): The condition of the asset is based on accumulated defect points. As with the general weighted method, each potential defect is assigned a weight value based on the impact to service level. Then each defect is broken down into specific severity levels with a point value assigned to each. The more severe the nature of the defect, the higher the associated point value. Normally, the point scale is a range of 1 - 10, allowing for at least 10 different definitions or levels of severity. Unlike the general weighted method, each defect and severity identified during inspection is accumulated for the final “deficiency score”. That is, if there are 5 defects with a severity of 2 points each, then the deficiency score for the asset would be 10. A “like new” asset would exhibit none of the potential defects and be awarded a deficiency score of 0 (i.e., absolutely no defects were noted). A severely broken asset would exhibit the highest severity of each potential defect, probably numerous times, and would be awarded a higher deficiency score.
- **Reset Condition Rating**: After the asset is rebuilt or renewed it is treated as if it were a new asset. The assessment process is started over and the condition of the asset is based, once again, on the passing or failing characteristics of a new asset.

Questionnaires

Questionnaires are a previously defined set of questions and answers used to guide inspectors and field personnel through the inspection process of an asset.

See [Understanding Questionnaires](#) for more information.

Maintaining Service History

This portal is used to display and maintain service history records.

Refer to [About Service History](#) for more information.

You can access the portal from the **Menu**, select **Asset Management**, then select **Service History**. You are brought to a query portal with options for searching for a specific service history. Once a service history has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Service History:** This zone supplies details about the service history for an asset.
- **Attachments:** This zone enables users to manage all types of attachments for the service history record.

Chapter 10

Work Management

About Work Management

Work and Asset Management (WAM) application is able to support the performance of work by crews in the field through the use of activities. An activity is a task used to track changes and other actions performed in the field, such as inspections, changes to assets, or even tree trimming and, if configured, initiates a completion event to perform the appropriate change to the data in the system.

Activities are categorized into three broad classes of work:

- Field Activity
- Work Activity
- Construction Work Activity

All three activity classes share the same Activity Type business object which specifies definitions, such as default service class, scheduling information, handling of errors or exceptions, completion details, and so on, for activities of that type. The uniqueness between each of the activity classes is derived from the Activity BO linked to the activity type. The Activity Type business object includes two flags, the **Work Role** flag and the **Track Cost** flag, that are hard-coded on the BO Schema and used to differentiate between the activity classes whether planning and approval is required:

- For Field Activity, the Work Role flag is set to “No” and the Track Cost flag is set to “No”.
- For Work Activity, the Work Role flag is set to require planning and the Track Cost flag is set to “Yes”.
- For Construction Work Activity, the Work Role flag is set to require planning and the Track Cost flag is set to “Yes”.

NOTE: Both flags should be set to *No* for **Operational Device Management** only customers.

NOTE: Also note that customers that use both **Operational Device Management** and **Work and Asset Management** have the option to set the Track Cost flag for field activities only to *Yes*.

See [Understanding Activity Types](#) in the **Admin Guide** for more information.

All activities share some common details across all activity classes:

- **Unique ID:** The system creates a unique number that identifies the individual activity and enables the system to track the activity details.
- **Work Location:** An activity's work location is used to describe the physical location of where the work is to be performed. This can include such information as main or sub addresses, building or room numbers, and intersections, cross streets, or a latitude and longitude. For example, a college campus might have a main address for the entire campus but the work location of the work to be done would be at a particular building on campus.
- **Schedule Details:** The system looks to the Schedule Details for determining when the work is scheduled to start and end.
- **Attachments:** Activities can include schematics, pictures, procedures, notations, and more, in the form of documents or attachments, to assist crew members with additional information about the work to be performed.
- **Completion Events:** Completion events are used to execute the business processes resulting from the activity, such as asset updates, status transitions, and so on.

Activities also include some behaviors that are unique to the individual class of activity. See the individual activity topic for more information.

- See [About Field Activities](#) for more information about field activities.
- See [About Work Activities](#) for more information about work activities.
- See [About Construction Work](#) for more information about construction work activities.

About Field Activities

Field Activity is used primarily by Operational Device Management (ODM) users to perform basic activities that do not require any resource requirements.

Work is accomplished by creating either an emergency field activity or a regular type field activity to accomplish the work relative to the originating activity, asset, or project.

This section describes concepts related to Field Activities.

Please see [Work Management](#) for more information.

Planning Field Activities

This class of activity is a very basic and independent type of activity. Field activities are not related to any work orders nor require any resource requirements so do not require any advanced planning.

Approvals for Field Activities

Field activities do not follow an approval process since no resource requirements or work orders are planned for a field activity.

Scheduling Field Activities

Schedule Details can be defined for field activities however, it is expected that the external integration will perform the work of scheduling the field activities.

The disposition of the field activity indicates when the activity is ready to be worked. Setting the disposition of a field activity to “On Hold” means that the activity is not yet ready to be activated for work. Setting the disposition to “Released” means that the activity is active, work can start, and completion events can be processed.

Field Activity Cost

There are no cost estimates associated with field activities because they do not include resource requirements.

Field Activity Workflow

The following is the typical workflow states of a field activity:

Pending

- This is the initial status of the activity

Validated

- This status validates the activity disposition value

Validation Error

- This status indicates there is a problem with the information on the activity

Sent

- This status indicates the activity has been sent to the external system

Resend

- This status indicates the activity has been resent to the external system

Send Error

- This status indicates there was error in sending the activity to the external system

Execute Completion Event

- This status indicates the work has been finished and the completion events are being executed

Completion Error

- This status indicates there was an error in processing the completion events

Executed

- This status checks for the required completion events and completes the activity

Completed

- The activity is set to the *Completed* state once the work is performed and the completion information is recorded

Discarded

- This status indicates the activity was discarded

About Work Orders

Work orders group similar activities and are used to manage activities to perform any kind of work such as inspection, install, replacement, upgrade, and so on. They are mainly used to facilitate approval processing for work projects.

Work orders might also be created to manage a set of activities where the activities are manually linked to the work order. Completion events are used on activities to document and process the work that was done and to update data associated to the activity. Work orders also provide summary information about the various activities and the assets that are being worked on those activities.

A work order and an associated work activity can be created directly from an approved work request.

- To create a work order and the initial activity, click the **Create Work Order** button on the approved work request. If only one work order business object exists, the user is immediately prompted to select an appropriate activity type. If more than one work order business object exists, select *work order* as the **Work Order Business Object** and then select the appropriate **Activity Type**. As long as either the Asset/Asset Location or Supplemental Work Location is defined, the system will automatically create the work order and the first activity, populating the activity with information from both the work request and work order.
- To create a work order and the initial activities from a template, click the **Create Work Order from Template** button on the approved work request and search for and select the desired template work order. Next, select the appropriate template work activities from the list of activities displayed on the template work order. Keep in mind that an Asset/Asset Location or Supplemental Work Location and a Cost Center is required in order for the system to automatically create the work order and activities from the template. Selecting the **Copy Cost Center from Asset Location** box will automatically populate the cost center information defined on the asset location to the template activities when created, otherwise, the cost center information defined on the template activities will be used.

Work Order Templates

It is possible to create templates from most types of work orders so that the general work model can be reused. Work order templates can also be referenced for scheduled maintenance.

Refer to [Template Work Orders](#) for more information.

Activity Information

The **Activity Information** zone provides summary information regarding the activities that are included on the work order.

Refer to the [About Work Activities](#) section for more information and field definitions.

Unplanned Work

Since unplanned work simplifies small, uncomplicated, and one-time jobs, foregoing planning can be effective when the task to be completed is straightforward and does not involve many details.

Completing Work Orders

Work orders are automatically completed when every activity linked to the work order arrives at a final state based on the completion events for the activities.

Work Priority/Asset Location Criticality/Total Risk Priority

Work orders contain a work priority value that, when used in conjunction with other values, calculates an overall criticality value for accomplishing the work. The following fields are used to perform those calculations:

- **Work Priority** defines the importance of the work in terms of the potential impact of not addressing issues. For example, emergency type work is a higher priority than routine maintenance work.
- **Asset Location Criticality** defines the importance of the location in terms of the potential impact of having issues or problems at that location. For example, a hospital might have a high criticality rating since losing power there might result in the death of patients and a city park would have a low criticality since there are few consequences if the lawns are not mowed on time.
- **Total Risk Priority** provides a total risk value in terms of scheduling work at a particular location. Total risk is calculated as work priority multiplied by asset location criticality. For example, if **Work Priority** is 6 and **Asset Location Criticality** is 7, then **Total Risk Priority** equals 42 ($6 \times 7 = 42$). This value can be overridden by the user.

Field is able to be updated to any value up to 99.

If a user overrides the value, then **Total Risk Priority Source** flag changes from *System* to *User Override*.

Please see [Work Management](#) for more information.

Work Order Includes One or More Activities

Work orders include one or more activities. Each activity identifies a specific task to be performed at a particular location and includes all the information that the crew will need to complete the job.

Work Order Workflow

Work orders are used to plan for all types of work that need to be performed. The following is the typical workflow of a work order:

Planning

This is the initial status of a work order and includes all the requirements needed to complete the type of work being planned. Requirements, such as activities, checklists, permits, attachments, resource requirements, and dependencies are all considered during the planning stage of the work order. If the work order references a work request, many of the work order fields are pre-populated by the system.

Pending Approval

This status indicates that the planning of the work order is finished and ready to be submitted to an approver.

Approved

This status indicates that the approver has reviewed the work order details and has approved the work to be done.

Active

This status sets the work order and associated work activities to active so the work can be performed and the activity can accept charges. Only activities where the **Activation Date** has passed are set to *Active*.

Reopened

This status indicates that a previously closed work order is now reopened.

Rejected

This status indicates that the work order was not approved but rejected by the approver.

Completed

This status indicates that all work activities associated with the work order have been completed and all completion details have been entered.

Closed

This status indicates that work order and ALL associated work activities is complete and no more work is to be done.

Canceled

This status is used to indicate that the work should not be performed.

Template Work Orders

Template work orders, or benchmark work orders, are work orders that can be re-used to help your organization avoid repeating planning the same work for similar jobs. They can be used for basic work planning, but are also typically used with preventive maintenance.

The template work order contains header work order details. Template activities and template activity resource requirements are used to plan those aspects of the work plan.

- **General Template Work Orders** are used to plan for activities and resources which can be common to assets regardless of the asset disposition. This type of template can be used in any type of Preventive Maintenance Trigger.
- **Asset Location Specific Template Work Orders** are used to plan for activities and resources which are specific to a list of locations where the assets are installed. Some refer to the list of locations as a route. A work order created from this type of template will automatically populate assets, of the asset type, if specified, installed at the locations. This type of template can only be used on Calendar Anniversary Preventive Maintenance Triggers and Calendar Interval Preventive Maintenance Triggers.

The template work order provides users with the ability to create a work order directly from the template work order and then update the work order with the selected activities from the work order template. Refer to [About Work Orders](#) for information on how to create a work order and associated activities at the same time from a template.

Preventive Maintenance

Template work orders are used with preventive maintenance and event-based maintenance to create new work orders when the maintenance is triggered. The appropriate work order template is referenced on the maintenance trigger so that the appropriate work records are automatically created when the trigger is cycled.

Usually, the work orders that are created from the template can be modified to represent the actual work requirements. For example, modifications may be needed if the requirements from the template don't match the exact requirements for the execution of the work.

Projects

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.

When a project is initiated and funds are approved, the project can be broken down into a hierarchical arrangement of smaller subprojects. Each subproject represents a controllable unit of work within the project. The ability to enter and manage information for the units of work, scheduling details, cost estimates, and target dates for each project and subproject forms the basis for planning the project and monitoring its progress to completion. Work orders and activities can be associated to the project and are automatically included in the cost of the project.

Projects can also be divided into subprojects – facilities construction, wiring, installation of equipment, testing of automated systems, shakedown, and so on. You can then plan, budget, and individually authorize each subproject.

Projects can be configured to go through regular approval processing, so that after the project is created it must be submitted for approval then approved.

Allow Charge

When the **Allow Charge** indicator is set to “yes”, other records can charge against the project. The project must be in *Approved*, *Started*, or *Complete* state for charges to be applied against it. For example, a resource, such as a craft or equipment, might reference the Project ID under the accounting information.

Note that if subprojects are planned, the project itself should have **Allow Charge** set to “No”. Charges should be tracked by subproject.

Project Categories

Your organization can define project categories to create classifications for projects such as maintenance, capital, construction, and so on. Project categories are initially created in an *Active* status but can be toggled to an *Inactive* status or back again to *Active*, depending on their future use.

About Work Activities

Work activities are work tasks that need to be completed and originate from a work order or a work order template. As examples, activities may be created for maintenance and inspection of assets, to create service history, and to install or exchange devices.

Work Activities can be created at the same time as the work order when the **Create Work Order** button on the approved work request is selected. Refer to [About Work Orders](#) for more information.

The Work Activity portal enables planners to manage such details as resource requirements, permits, materials, labor, and much more that control key aspects of how the work activity is managed.

Many settings are shared between the activity classes because they share the same activity type BO however, a setting unique to work activities and construction work activities is the “Construction Related” flag which defines whether the activity is to be a regular activity or a construction work activity.

See the various sub-topics in this section for concepts related to the management of work activities.

Planning Work Activities

Planners control key aspects of how a work activity is managed:

- The type of work to be performed, for example, maintenance, installs, replacements, and inspections.
- Where the work is to be performed.
- The types of resources that are required; labor, equipment, materials, and so on.
- List of assets the work is to be performed against.
- Checklist of tasks to be performed against the work activity or permit.
- Whether or not to create service history.

NOTE: The benefit of using activities to process work is to keep a history of the work being done. Users can specify on the work order activity and template work order activity whether a service history for a specific service history type should be mandatory or optional. If set as mandatory, the service history would need to be completed before the work order can be closed. This history can then be used for tracking and auditing.

Resource Requirements

Every work activity is defined with one or more resource requirements. Resource requirements are the material, crafts, equipment, and other miscellaneous resources needed to complete the work.

- Material requirements specify materials or stock items to be used for the activity (pipe, pump, bolt, etc.). Material requirements might also be fulfilled using the procurement process.

- Labor requirements, or crafts, specify the type of skills needed for the activity (technician, plumber, electrician, etc.). This might include premium or regular labor.
- Equipment requirements specify the machinery, tools or other equipment required for the activity (crane, bulldozer, drill, etc.).
- Other requirements cover miscellaneous requirements for the activity (meals, miles, hotel stay, etc.).

Activity resource requirements can only be navigated to and added from the **Activity** portal's **Resources** tab.

Note the following with respect to activity resource requirements:

- Material requests (issues and returns) can be applied against the resource requirement when the activity is active.
- If materials are not in stock and need to be purchased to fulfill the requirement, a purchase order or requisition might be issued. These materials are typically tracked as direct purchases.
- Timesheets are used to track labor requirements. The labor requirement establishes an estimated number of work hours required. As timesheets are created and posted, the actual work hours are tracked.
- Other direct charges are used to track equipment and other requirements.

Approving Work Activities

Typically, planned work orders specify resource requirements on the related activities. Once planning is complete the planner can submit the work order and related work activities to an authorized approver to verify the requirements before any work or purchases are submitted. Authorized approver can **Approve** the work order, **Withdrawal the Approval Request**, or edit the resource requirements and approve the changes. Also, approved work orders can be reopened to add new activities and resource requirements and then resubmitted for approval. The status of the related work activities is changed based on the state of the work order, for example, Approving the work order automatically sets the status of the related work activities to **Approved** also.

Scheduling Work Activities

Work activities can be scheduled by an external system (determined by the existence of an external system defined on the work activity type) or from within the Work and Asset Management application. Within Work and Asset Management, a **Scheduling** portal is used to schedule the labor requirements associated with a work activity.

Scheduling accomplishes this in the following way:

- Looks at the service class listed on the work activity and matches that service class with a crew shift having the same service class.
- Matches the labor requirements and estimated duration of the activity with a crew shift having those same capabilities.
- Work activities that are not completed on the current shift are reassigned to the next available shift.
- The system is queried for new activity labor requirements and any unfinished backlog of work.

Once the shift assignments are complete, the schedule is activated.

A dispatcher may override scheduler decisions at any time.

Work Activity Cost

The system tracks usage and reports the costing associated with that usage against the total cost of completing the activity.

Resource Estimates

Activity resource requirements determine the original estimates or planned amount for labor, equipment, materials and other on activities. Even canceled activity resources requirements are calculated as part of the original estimate. The original estimate is the value of the resource requirement at the time the work order was initially approved and is snapshot into a separate column when the initial approval happened. If an approved work order is reopened to add new activities and resource requirements and resubmitted for approval, those new resource requirements won't have an initial estimate (it will be zero). The system calculates a revised estimate to reflect the current value of the resource requirement. For example, a resource requirement was initially planned for 10 pumps at \$10.00 each. Upon the initial approval, the initial estimate will be stamped with \$100.00 and the revised estimate is also \$100.00. If the planner changes the requirement to 15 pumps instead, the original estimate will stay at \$100.00 and the revised estimate will be \$150.00. Canceled requirements are not calculated as part of the revised estimate.

Committed Amount

Committed amount is the total of costs applied to the activity due to direct purchase costs that are committed when a purchase order is issued. This value uses the issued amount, not closed or completed, since the costs are considered as committed once the purchase order is issued to the vendor. This amount reflects the total due to the vendor, as full or partial invoice amounts are closed. It does not reflect amounts that are completed or closed.

For example, a purchase order line has been issued for 10 pumps at \$10 each. Assume 4 pumps were received and a partial invoice line of \$40 was issued for the 4 items. This partial invoice for \$40 reduces the committed amount — \$100 minus \$40 = \$60. Once the other 6 items are received, the purchase order line moves to closed status and it does not have an impact on the committed amount.

Unplanned Resources

Unplanned resources are defined as issued or returned materials or timesheet entries that were not part of the original resource requirements or estimates. Activities include a zone which lists any unplanned activity resources.

Fulfilling Material Requirements

Material requirements can be fulfilled in the following ways:

- For inventory, if the issued quantity is greater than or equal to the planned quantity, the system automatically changes the resource requirement to Fulfilled. This happens in cases where the crew has checked out all the planned quantity from the storeroom.
- For direct purchased material, if the received quantity is greater than or equal to the planned quantity, the system updates the resource requirement to Fulfilled. This method of automatic fulfillment is only applicable to material.
- For inventory material, when the resource requirement is approved (as part of the WO approval), the system puts the quantity as Pending Demand on the storeroom. Once the activity material requirement becomes Active, that quantity becomes On Demand (and Pending Demand is closed out or becomes zero).
- Materials can also be “reserved”. If the storeroom has enough quantity, then that quantity becomes reserved for the given amount. The “reserved” material can still be issued to a different activity since it is only an indicator and not “On Hold”. The reserved quantity provides storeroom managers with visibility into what is to come so they can order materials accordingly.

Note that, although it may not be part of work, when material is issued from a storeroom, material issue will reduce demand by the issued quantity or the original planned quantity, whichever is less. This is to prevent a negative on demand quantity (if the issue quantity is more than the planned quantity).

When an activity is completed, it updates all its resource requirements, including material requirements, to Fulfilled to close out any remaining **On Demand/Reserved** quantity.

Checklists

Checklists are forms that define the prerequisites or procedural steps that need to be accomplished during the lifecycle of another object. For example, checklists can be used for managing permits, information purposes only, or contain the actual steps the user must take prior to completing work in the field.

Planners can use the activity portal to manage checklists directly from the activity record. Here's how it works:

- A checklist type can contain one to many checklist line items.
- Planner creates a checklist for the activity using a checklist type.
- A checklist cannot be completed until all the mandatory items have been marked as complete.
- An activity cannot proceed to certain status until all the required checklists have been completed.

There are two classes of checklist enabled for work activities in the base product:

General Checklist

General checklists are previously defined sets of details or steps. Each line item in the checklist is just a description. All mandatory line items in a checklist must be marked as complete before the user can complete the checklist.

For example, Circle of Safety Checklist

- Park vehicle on level surface.
- Place cones on front, side, and rear of vehicle.
- Turn on emergency lights.

NOTE: Each of the bulleted items would appear in the checklist in sequence and have a Mandatory definition (Yes/No).

Permit Item Only Checklist

Permit checklist is a new class of checklist to manage permit requirements. Each line item of this checklist refers to a permit.

Permit checklists have additional rules that govern how and when the checklist or checklist items can be completed. A checklist item can only be marked as complete when its associated permit has been activated.

Permit checklists are available for both work activities and construction work activities and are managed from the activity's **Permit Checklist** zone on a separate **Permit** tab.

Though it's probably unlikely, you can also enable such checklists for other objects, just like the General Checklist above. Please refer to WAM Implementer training for how to enable checklists for other objects.

Work Activity Workflow

Work order activities are associated with a work order and are the tasks and resources required for completing the work. The following is the typical workflow states of a work activity:

Planning

This is the initial status of the work activity which includes all the resource and cost information needed to work the type of work being planned for and to appropriately charge cost centers for resources and materials.

Pending Approval

This status indicates that the planning of the work activity is finished and ready to be submitted to an approver.

Approved

This status indicates that the approver has reviewed work activity details and has approved the tasks to be done.

Active

This status sets the work activity to active so the work can be performed and the activity can accept charges.

In Progress

This status indicates the work activity has been started by the crew but the work is not yet finished

Finished

This status indicates the crew has finished the work

Execute Completion Events

This status indicates the work is finished and the completion events are being executed.

Completion Error

This status indicates there was an error in processing the completion events.

Completed

The activity is set to the *Completed* state once the work is performed and completion information is recorded

Canceled

This status is used to indicate that the task should not be performed

Rejected

This status indicates that the work activity was not approved but rejected by the approver.

The following statuses are only applicable to “outbound” activities:

Sent

This status indicates that the activity has been sent to an external system for scheduling and processing, for example, Oracle Field Service Cloud (OFSC).

Resend

This status indicates an error was corrected and resent to the server.

Send Error

This status indicates that the activity does not have all the necessary details to enable it to be properly scheduled in an external system.

Work Activity Templates

Template activities are the activities generated as part of a template work order. In this way, template activities can be re-used to help your organization avoid repeat planning of the same work for similar jobs. Template activities can be used for the planning of basic work but are also typically used for preventive maintenance.

Template activities are part of a single template work order and cannot be re-used by multiple template work orders.

Template Activity Resource Unit Price

Resource cost estimates on a template activity can change over time and should be updated to reflect the current pricing for each resource. The system uses a [W1-TMPWO](#) periodic monitor batch process to obtain the current pricing for resources.

Asset Location Specific Template Activities

Template activities that reference an asset location can be set in advance to trigger preventive maintenance work. A list of asset locations can be added to the template activity which can later be used as a route for preventive maintenance with

costs for performing the work distributed across both primary and secondary assets. The percentage of cost distribution across all the assets must equal 100 percent.

About Work Requests

Work Requests are used to request various types of work or services to be performed.

Work requests can be indirectly linked to a new activity when the activity adds a work order that is linked to a work request. Work requests can be created manually, as a follow-up to an activity, or by an external system. Work requests are mainly used to implement an approval chain for the requested work.

When a work request is created it is reviewed to determine if the request is valid and can be turned into a work order. If the work request is rejected or canceled the work requestor is notified and the work request can be resubmitted. If the work request is approved, the planner can either add the request to an existing work order as a new activity or create a new work order and work activity specifically for the work being requested.

Once approved, users can select **Create Work Order** and select the appropriate work order business object and the associated activity type to automatically create both the work order and the first work activity which is then populated with information linked to and obtained from the work request. Cost Center details and at least one of the following is required to create the work order and work activity:

- Asset Location
- Work Location
- Asset

The system retrieves the cost center information, if populated, from the following:

- Asset location specified on the work request
- Template work order
- If not populated, the work activity is rendered and cost center information can be entered directly

If necessary, it is possible to create a work order directly if you have the proper authorization.

Possible Duplicate SC/WR/WO/Activity

The system identifies any records which could possibly be duplicate service calls, work requests, work orders, or activities in this zone. Any work records that are in “non-final” statuses (such as *created*, *pending approval*, *approved*, *active*, *in progress*, or *finished*) with the same asset and location (including address) of the current work request are listed as duplicates. You can choose to include or exclude work records related to preventive maintenance. If a record is found to be a duplicate, the work request can be canceled under **Record Actions**.

About Work Planning

Work Planning is a portal for those employees responsible for planning work. This portal contains a variety of zones that enable planners to view multiple summary graphs for open work activities and to search for, update, and schedule multiple work activities at one time. Planners can use this portal to manage their long to mid-term planning needs while also resolving any backlog of activities.

The **Work Planning** portal shows planners the number of open work activities in a variety of graphical views:

- Open Work Activity by Status
- Open Work Activity by Status (Without Planner)
- Open work Activity by Phase

- Open Work Activity by Held for Parts grouped by Required by Date

Planners can help coordinate many types of specialized activities and manage many other types of work activities by using the various search criteria inherent in this portal:

- Activity Information
- Activity Work Windows
- Activity Checklist Information
- Activity Permit Information
- Activity Maintenance Information
- Activity Labor Requirement Information
- Previously Scheduled Unfinished Work

Planners can also use the results of the search to select multiple activities and mass update activity values at one time:

- Work Window Start and End Date
- Phase
- Work Priority
- Work Class
- Work Category
- Planner
- Outage Type

The search results can also be used to select multiple activities with labor requirements and schedule them directly to a crew shift.

Planners can also use the Log Entry to add notes in the audit log.

Please see [Work Management](#) for more information.

About Permits

A permit is a form of documentation that proves a company has met certain safety or regulatory requirements before any work has begun.

Some examples of this kind of work might be tasks that require OSHA certification or specialized handling of hazardous waste materials or specialized equipment.

Planners decide what permits are needed for the work activity and then adds the required permits using a special type of checklist. Refer to [Checklist Types](#) and [Understanding Permit Templates](#) for more information.

Although configurable, most often no work activity can start until the permit has been obtained and activated.

This section describes concepts related to managing permits.

Please see [Work Management](#) for more information.

About Communication Logs

Communication logs show all the communications made against a work document and its related work documents. For example, the Communications Log zone on a work request will display communication details when a service call led to the creation of that work request.

Communication logs are displayed on a variety of work records, such as work designs, work requests, work orders and construction work orders, and service calls. Log details provide users with the date and contact person who created the communication, any comments or responses, and the ability to view pertinent files and attachments. Users are able to add new, edit existing, or completely delete communication logs as additional information is received.

Refer to [About Construction Work](#) for more information.

About Service Calls

A service call enables work to be created when a call from either an internal or external resource is received by a company representative.

Some examples of these types of calls would be as follows:

- Customer calls in to report a streetlight out, odor in area, sewage overflow, service run to new home, etc.
- Contractor requests new service to be run for a new subdivision or a business requests an upgrade in service for an expansion project.
- Field employee finds an additional problem after responding to an original call.

Service calls can be created by navigating to **Menu**, click **Work Management**, and click **Service Call** or directly from the **Contact** record. Users can avoid creating duplicate service calls by checking the **Possible Duplicate SC/WR/WO/Activity** zone on the work request or from the **Service Call** record itself.

Work records, such as work designs, work requests, work orders, and work activities, can all be created as a result of the service call. Work records initiated from or associated with the service call will display in the **Service Call Related Documents** zone of the **Related Documents** tab where users can click on the work document hyperlink and go directly to the individual work record. Users can create contacts and view and manage service call details, such as a description of the service call, contacts, attachments, etc., directly from the work record. Also, the **Possible Duplicate SC/WR/WO/Activity** zone on the service call record displays any service calls that are duplicates.

Service calls are able to be linked and unlinked to other service calls.

Configure Service Category and Service Code objects to automatically create follow-up documents such as work requests and work orders.

Refer to the [Understanding Service Call](#) section of the **Administrative User Guide** for more information on service categories and service codes.

Please see [Work Management](#) for more information.

Maintaining Field Activities

The **Field Activity** portal is used to view and maintain field activities.

Refer to [About Field Activities](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Field Activity**. You are brought to a query portal with options for searching for specific record. Once the field activity has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Field Activity:** This zone displays general information, such as location, asset, service class, and so on about the selected field activity record.
- **Completion Information Entry:** This zone displays a list of completion events for the field activity.
- **Attachments:** This zone enables users to manage all types of attachments for the field activity record.

The following zones may appear as part of the portal's **Cost** tab page:

- **Activity Cost Summary:** This zone displays a summary of costs associated with the field activity record.

Maintaining Work Orders

The **Work Order** portal is used to view and maintain work orders.

Refer to [About Work Orders](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Work Order**. You are brought to a query portal with options for searching for specific record. Once the work order has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portals **Main** tab page:

- **Work Order:** This zone provides information about the selected work order record, such as work details and accounting information.
- **Activity Info:** This zone lists the activities associated with the work order.
- **Follow Up Work Order and Work Request:** This zone lists all follow up work orders and work requests associated with the work order record.
- **Activity Dependency:** This zone lists the activity dependencies for the current work order record.
- **Notes:** This zone lists any notes that have been added for the work order record.
- **Scheduled Activities:** This zone displays a list of the scheduled activities associated with the work order record.
- **Contact Relationship:** This zone displays contact information associated with the work order record.
- **Communication Log:** This zone displays any communications associated with the work order record.

The following zones may appear as part of the portals **Cost** tab page:

- **Work Order Cost Summary:** This zone displays a summary of costs associated with the work order record.
- **Work Order Cost Detail:** This zone displays upon broadcast from the Work Order Cost Summary and displays a hyperlink to the work activity.

The following zones may appear as part of the portals **Activity Resource Tree** tab page:

- **Activity Resource Requirement Tree:** This zone displays a tree view of required resources for the work order.

The following zones may appear as part of the portals **Log** tab page:

- **Work Order Log:** This zone displays log details for the work order.
- **Approval Request History:** This zone lists approval history for the work order.

Maintaining Template Work Orders

This portal is used to view and maintain template work orders.

Refer to [About Work Orders](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Template Work Order**. You are brought to a query portal with options for searching for specific work order templates. Once the template has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portals **Main** tab page:

- **Template Work Order:** This zone provides information about the selected template and enables user functions to define projects, work priorities, work order approval profiles, and more.
- **Template Activity:** This zone lists the template activities associated with the work order template.
- **Notes:** This zone lists all the notes that have been added for the work order template.

The following zones may appear as part of the portals **Cost** tab page:

- **Template Work Order Cost Summary:** This zone displays cost summary information associated with the selected work order template.

Maintaining Work Activities

The **Work Activity** portal is used to view and maintain work activities.

Refer to [About Work Activities](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Work Activity**. You are brought to a query portal with options for searching for specific record. Once the work activity has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Work Activity:** This zone provides information about the selected work activity record, such as schedule details, special handling, and accounting information.
- **Asset List:** This zone lists the assets associated with the work activity.
- **Checklists:** This zone displays all checklists associated with the work activity record.
- **Work Order Activities:** This zone displays when there is more than one activity on the work order.
- **Outstanding PM Activity:** This zone displays any outstanding preventive maintenance activities associated with the work activity record.
- **Notes:** This zone lists notes associated with the work activity.
- **Attachments:** This zone enables users to manage all types of attachments for the work activity record.
- **Asset Attachments:** This zone lists attachments associated with any of the activity's assets.
- **Active Warranty:** This zone displays the active warranties associated with the work activity record.
- **Active Dependency:** This zone lists the active work activities that are dependent on the current work activity record..
- **Scheduled Labor Requirements:** This zone displays the scheduled labor requirements associated with the work activity record.
- **Service History:** This zone displays only when there are service histories that reference this activity.
- **Follow Up Work:** This zone displays the activities created from a service history associated with this work activity.

The following zones may appear as part of the portal's **Resources** tab page:

- **Quick Add:** This zone enables additional resources to be added to the work activity record.
- **Activity Resource Action:** This zone displays the activity's resources. Users can select the resource hyperlink and go to the activity's resource requirement to make changes.
- **Activity Resource Detail:** This zone displays resource details associated with the activity record.
- **Unplanned Activity Resource:** This zone displays resource details generated from actual charges, like timesheets that were not planned or estimated.

The following zones may appear as part of the portal's **Cost** tab page:

- **Activity Cost Summary:** This zone displays a summary of costs associated with the work activity record.

- **Activity Cost Detail:** This zone displays the cost details associated with the work activity record when a row from the **Activity Cost Summary** zone is broadcast.

The following zones may appear as part of the portal's **Permit** tab page:

- **Activity Permit Checklists:** This zone displays any permit checklists associated with the work activity.
- **Permit Attachments:** This zone displays permit attachments associated with the work activity.
- **Service History Type:** This zone lists planned service history types for the activity, indicating for each service history type whether it is required to be filled out or not. When an asset is in context the list shows all valid service history types for the asset.
- **Service History:** This zone lists service history records associated with the activity and the asset in context.

The following zones may appear as part of the portal's **Completion** tab page:

- **Asset List:** This zone lists locations and assets associated with the activity, including assets installed, removed or attached as part of this activity. The zone only appears for an asset related activity. Broadcasting an asset and location row in this list sets is as the current context.
- **Completion Actions:** This zone displays valid completion related actions that may be performed on the activity, asset and location in context.
- **Activity Service History:** This zone lists service history records associated with the activity and reference no asset. This zone typically appears when the activity is not asset related. The zone may also appear when such service histories were entered for an activity that has started as non-asset related and later was updated to become asset related.
- **Service History Types:** This zone lists valid service history types and condition assessment service history types associated to the assessment class for the asset in context or planned service history types for the activity.
- **Asset Service History:** This zone lists service history records associated with the activity and either the asset or location in context.
- **Measurements:** This zone lists measurements associated with the activity and asset in context. The zone appears only if recording measurements is applicable to the asset in context.
- **Completion Events:** This zone lists completion event records that may have been initiated by processing completion information sent from an external system.

Maintaining Work Requests

The **Work Request** portal is used to view and maintain work requests.

Refer to [About Work Requests](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Work Request**. You are brought to a query portal with options for searching for a specific record. Once the work request has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portals **Main** tab page:

- **Work Request:** This zone provides information about the selected work request record, such as asset and contact information.
- **Work Detail:** This zone provides a hyperlink to the work order associated with the work request.
- **Attachments:** This zone displays attachments associated with the work request record.
- **Contact Relationship:** This zone displays contact information associated with the work request record.
- **Communication Log:** This zone displays any communications associated with the work request record.

The following zones may appear as part of the portals **Possible Duplicates** tab page:

- **Possible Duplicate SC/WR/WO/Activity:** This zone enables a search for possible duplicate records.

The following zones may appear as part of the portals **Log** tab page:

- **Work Request Log:** This zone displays log details associated with the work request record.
- **Approval Request History:** This zone lists all the approval requests associated with the work request record.

Maintaining Work Location

The **Work Location** portal is used to display and maintain a work location record.

Refer to [About Work Activities](#) for more information.

You can access the portal from **Menu**, select **Work Management**, then select **Work Location**. You are brought to a query portal with options for searching for a specific work location. Once a work location has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Work Location:** This zone displays information about the work location, such as address and latitude and longitude coordinates.
- **Attachments:** This zone enables users to manage all types of attachments for the work location record.

Maintaining Projects

The **Projects** portal is used to view and maintain projects.

Refer to [About Work Orders](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Project**. You are brought to a query portal with options for searching for a specific record. Once the project has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Project:** This zone provides information about the project, such as whether to allow other records to charge against the project, scheduling information, and accounting information.
- **Child Project Info:** This zone lists the subprojects associated with the project.
- **Attachments:** This zone enables users to manage all types of attachments for the project record.
- **Notes:** This zone lists notes associated with the project.

The following zones may appear as part of the portal's **Cost** tab page:

- **Project Cost Summary:** This zone displays a summary of costs associated with the project record.
- **Project Cost Detail:** This zone is originally hidden but displays cost details by broadcasting the cost category in the **Project Cost Summary** zone.

The following zones may appear as part of the portal's **Tree View** tab page:

- **Project Hierarchy:** This zone displays a tree structure of the project hierarchy for the current project.

The following zones may appear as part of the portal's **Work** tab page:

- **Work Order Status Summary:** This zone displays a graph of both regular work orders and construction work orders associated with the project and subprojects in its hierarchy by status.
- **Work Order:** This zone displays both regular work orders and construction work orders for the current project and associated subprojects in its hierarchy.

- **Work Activity Status Summary:** This zone displays a graph of regular work activities and construction activities associated with the project and subprojects in its hierarchy by status.
- **Work Activity:** This zone displays a list of both regular work activities and construction activities associated with the project and associated subprojects in its hierarchy.

The following zones may appear as part of the portal's **Timesheet** tab page:

- **Timesheet:** This zone displays non-cancelled timesheet detail records associated with the current project and subprojects in its hierarchy.

The following zones may appear as part of the portal's **Work Design** tab page:

- **Work Design:** This zone displays non-cancelled work designs associated with the current project and subprojects in its hierarchy.

The following zones may appear as part of the portal's **Purchase Order** tab page:

- **Purchase Order Line:** This zone displays non-cancelled purchase order lines associated with the current project and subprojects in its hierarchy.

The following zones may appear as part of the portal's **Log** tab page:

- **Project Log:** This zone displays log details for the current project.
- **Approval Request History:** This zone lists the approval history for the current project.

Maintaining Permits

This portal is used to view and maintain permits.

Refer to [About Permits](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Permit**. You are brought to a query portal with options for searching for specific permits. Once the permit has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portals **Main** tab page:

- **Permit:** This zone provides general information about the selected permit, such as permit locations, labor, and work start and end times.
- **Asset/Location:** This zone displays the asset or location information associated with the permit.
- **Permit Holder History:** This zone displays history details about the permit holder.
- **Notes:** This zone lists notes associated with the permit.
- **Attachments:** This zone displays attachments associated with the permit. Attachments added here are made available as reference information related to the permit and are available on work activity.

The following zones may appear as part of the portals **Checklist** tab page:

- **Checklists:** This zone displays the checklists associated with this permit.

The following zones may appear as part of the portals **Service History** tab page:

- **Permit Service History:** This zone displays service history and activity information associated with this permit.

The following zones may appear as part of the portals **Related Activity** tab page:

- **Permit Related Activity:** This zone displays the related activities associated with this permit.

Maintaining Service Calls

This portal is used to view and maintain service calls.

Refer to [About Service Calls](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Service Call**. You are brought to a query portal with options for searching for specific service calls. Once the service call has been selected you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Service Call:** This zone provides information about the selected service call and enables user functions to create additional contacts, create work orders, work designs, work requests, or activities.
- **Contact Relationship:** This zone lists the contacts associated with the service call.
- **Communication Log:** This zone lists all the communications that have transpired for the service call.
- **Attachments:** This zone displays attachments associated with the service call. Attachments added here are made available as reference information related to the service call and are available on work requests, work orders, and work activities.

The following zones may appear as part of the portal's **Possible Duplicates** tab page:

- **Possible Duplicate SC/WR/WO/Activity:** This zone lists the service calls, work requests, work orders, and activities that might be possible duplicates of the service call.

The following zones may appear as part of the portal's **Related Documents** tab page:

- **Service Call Related Documents:** This zone displays hyperlinks to work documents, such as work requests, work designs, and work orders, that were initiated from or associated with the service call.
- **Associated Service Calls:** This zone displays all service calls associated with the service call record being displayed.
- **Link Service Call:** This zone links service calls to the service call being displayed.

Work Planning Dashboard

Work Planning is a portal for those employees responsible for planning work.

Refer to [About Work Planning](#) for more information.

You can access this portal from **Menu**, select **Work Management**, and then select **Work Planning**. You are brought to a Work Planning dashboard displaying summary graphs for managing the planning of work.

The following zones may appear as part of the portals **Main** tab page:

- **Open Work Activity by Status:** This zone displays a count of work activities by status for a particular planner.
- **Open Work Activity by Phase:** This zone displays a count of work activities by phase for a particular planner.
- **Open Work Activity by Status: (Without Planner).** This zone displays a count of work activities by status.
- **Open Work Activity by Held for Parts:** This zone displays a count of work activities that are Held for Parts and grouped by how many weeks they are from being overdue or if they are already overdue.
- **Work Activity Search:** This zone enables a search for activities using a variety of criteria, such as Planner, Crew, Craft, and so on.

Maintaining Activity Generators

The **Activity Generator** portal is used to view and maintain activity generators.

Refer to [About Work Orders](#) for more information.

You can access this portal from **Menu**, select **Work Management**, then select **Activity Generator**. You are brought to a query portal with options for searching for specific record. Once your record has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portals **Main** tab page:

- **Activity Generator:** This zone displays basic attributes of the selected activity generator record.
- **Activity Info:** This zone displays summary information about the activities created by the activity generator.
- **Notes:** This zone lists any notes that have been added for the activity generator.

The following zones may appear as part of the portals **Activities** tab page:

- **Activity:** This zone displays a list of activities created by the activity generator.

The following zones may appear as part of the portals **Cost** tab page:

- **Work Order Cost Summary:** This zone displays a summary of work order costs for the activity generator.

The following zones may appear as part of the portals **Log** tab page:

- **Activity Generator Log:** This zone displays the server logs for the activity generator.
- **Related Stock Transactions:** This zone displays the related stock transactions for the activity generator.

Chapter 11

Preventive Maintenance

About Preventive Maintenance

Preventive maintenance programs can be utilized to keep your business in compliance with regulatory requirements and business needs. Users can define compliance rules to ensure the work, such as inspections, repairs, cleaning, part replacements, and so on, is completed in a timely manner and to track and report against compliance work.

Preventive maintenance can help your organization manage costs by reducing downtime due to unexpected asset failures, reducing incidents where maintenance is needed in response to a problem (reactive maintenance), and ensuring the proper care and maintenance of assets. One common example of preventive maintenance is routine maintenance on a pump. It is clear that proper maintenance depends on timely visual inspections and regular cleaning to prevent costly problems, failures and repairs. Preventive maintenance programs are based on time or operational usage.

Template work orders are set up to plan detailed work requirements, and the preventive maintenance functionality manages when and how the related work orders are automatically created based on the established preventive maintenance cycles.

Maintenance Forecasts

The upcoming maintenance for an asset is forecasted based on the maintenance schedule and triggers defined for that asset. This information is displayed in the **Asset** portal on the **Maintenance Schedule Forecast** zone

Maintenance Specifications

Assets inherit their maintenance schedule from their **Maintenance Specification**. One **Maintenance 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 “Out of Service”.

Event Based Maintenance

Event based maintenance is work that is performed based on trigger dates. Events differ from maintenance schedules because the system does not manage when to cycle the work. Instead, planners decide upon the next schedule date for the event and enter it on the **Maintenance Event** record. Work performed can include inspections, repairs, replacements, and

so on, but is typically associated to planned outages or other event-based maintenance. Work records are generated on the trigger date based on the indicated defaults and templates.

The maintenance event defines the trigger date on which the maintenance work orders and activities are generated according to the indicated maintenance plans associated to the maintenance event. The event also specifies the work activity schedule defaults which will be set on the generated activities when they are created.

Maintenance Plans

Maintenance plans define the event category for maintenance events where the plan is referenced. The list of plan triggers on the maintenance plan then defines what work orders are generated for the event.

Maintenance Plan Triggers

Maintenance plan triggers are used to configure a specific work order for a particular location for use by a maintenance event. They define the maintenance plan, template work order, activity type, service history type, work order defaults and asset location for the work being created for the maintenance event, as applicable. The work order template indicated on the trigger defines the pre-planned requirements for the work that is created for the event.

Maintenance Schedules

Preventive maintenance schedules contain a list of maintenance triggers and a schedule of when maintenance should occur based on those triggers. Maintenance schedules provide a comprehensive set of analytics which provide information on failure types, failure modes, and repairs.

Duplicate Maintenance Schedules

The system provides a utility for duplicating a maintenance schedule and allowing the user to choose which maintenance triggers to also copy to the new schedule via a pick list. This action is available from the **Maintenance Schedule** record actions.

Maintenance Triggers

Preventive maintenance triggers indicate a certain interval which, when it occurs, maintenance should be done on an asset. For example, typical maintenance triggers would be when a vehicle reaches a certain number of miles, when a certain amount of time has passed, or when an asset reaches a certain number of hours of runtime.

Refer to [Defining Compliance Types](#) for more information about linking and unlinking preventive maintenance triggers.

Maintenance triggers are generally defined as one of the following:

- **Calendar Anniversary** is a *fixed* calendar interval type where regardless of the time lapse between the work creation and completion, the future maintenance schedule date remains the same. This type of interval scheduling is generally used to fulfill regulator compliance for inspections and licensing requirements on fixed date schedules. For example, this would trigger an inspection every three months regardless of when the last inspection was completed.
- **Runtime Anniversary** is a *fixed* interval type maintenance where regardless of the metered operation between the recent activity or service history and completion of work, the maintenance is not re-forecast. For example, this would trigger an inspection after every 100 hours of runtime service.
- **Calendar Interval** is a *fluid* interval type where the forecasted scheduled date is adjusted and re-forecast based on the completion date of the previously completed work which is managed through service history. For example, this would trigger a pump valve cleaning every 3 months with the next trigger based on the most recent cleaning.
- **Runtime Interval** is a *fluid* interval type where the maintenance schedule is adjusted and re-forecast based on the completion runtime readings of the previously completed activity. For example, this would trigger a pump bearing bracket cleaning every 100,000 gallons flow of water with the next trigger based on the gauge reading at the time of the last cleaning.

Maintenance Cycles

Preventive maintenance is triggered, or cycled, based on the maintenance schedule defined for the asset type. Batch processing determines when and how to create work orders depending on whether or not non-final activities already exist for the activity type and service history detailed on the maintenance trigger. Settings on the maintenance trigger such as the **Allow Multiple In Process Activities** flag determine whether or not a new activity is triggered if there are any non-final activities for the activity type and the asset being evaluated. If new activities should be created based on all criteria being met, the system automatically creates a work order based on the template work order indicated on the maintenance trigger.

Work records automatically generated by the system can be automatically scheduled to the field. From the maintenance trigger record, users can configure the system to automatically schedule preventive maintenance work to go to one of the crew's "Weekly" shifts. Configuration for automatic scheduling of work to a crew looks for the following conditions:

- In the **Work Order Defaults** zone, **Auto Schedule** value is set to Yes and the **Crew** value is defined with a crew name. If the **Crew** field is vacant, scheduling looks for a crew name defined on the template work order or work activity.
- Scheduling only considers those shifts that are in a *Planning* status.
- If the generated activities have a **Required By Date**, only those shifts with a start date prior to the activity's **Required By Date** are considered. The system starts with the earliest eligible shift having enough available and remaining hours to work the job. If no shift satisfies the requirement, the system schedules the activity to the shift with the most available hours.

NOTE: This feature is not applicable for activities whose activity type specifies an external system.

Users with the necessary authority in the system can manually execute maintenance triggers, as necessary, using actions on the **Asset** portal. Upcoming triggers are displayed in a zone in the **Asset** portal where users can execute the **Cycle Now** action to manually cycle the maintenance

Maintaining Maintenance Events

The **Maintenance Event** portal is used to display and maintain a maintenance event record.

Refer to [About Preventive Maintenance](#) for more information.

You can access the portal from **Menu**, select **Preventive Maintenance**, then select **Maintenance Event**. You are brought to a query portal with options for searching for a specific record. Once a maintenance event has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Maintenance Event:** This zone displays information about the maintenance event record, including maintenance plans and work activity schedule defaults.
- **Maintenance Event Information:** This zone displays summary information about the work activities associated with the maintenance event.

The following zones may appear as part of the portal's **Work Detail** tab page:

- **Plan Trigger:** This zone displays plan trigger information about the maintenance event.
- **Activity:** This zone displays the work activities associated with the maintenance event.

The following zones may appear as part of the portal's **Inventory** tab page:

- **Stock Item Availability:** This zone displays stock item availability information for the maintenance event.

Maintaining Maintenance Plans

The **Maintenance Plan** portal is used to display and maintain a maintenance plan record.

Refer to [About Preventive Maintenance](#) for more information.

You can access the portal from **Menu**, select **Preventive Maintenance**, then select **Maintenance Plan**. You are brought to a query portal with options for searching for a specific record. Once a maintenance plan has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Maintenance Plan:** This zone displays information about the maintenance plan.
- **Plan Trigger:** This zone lists all the triggers associated with the maintenance plan.
- **Maintenance Event:** This zone lists the maintenance events associated with the maintenance plan.

Maintaining Maintenance Plan Triggers

The **Maintenance Plan Trigger** portal is used to display and maintain a maintenance plan trigger record.

Refer to [About Preventive Maintenance](#) for more information.

You can access the portal from **Menu**, select **Preventive Maintenance**, then select **Plan Trigger**. You are brought to a query portal with options for searching for a specific record. Once a plan trigger has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Plan Trigger:** This zone displays information about the plan trigger record, such as processing information and work order defaults.
- **Activity:** This zone lists the work activities associated with the plan trigger record.

Maintaining Maintenance Specifications

The **Maintenance Specification** portal is used to display and maintain a maintenance specification record.

Refer to [About Preventive Maintenance](#) for more information.

You can access the portal from **Menu**, select **Preventive Maintenance**, then select **Maintenance Specification**. You are brought to a query portal with options for searching for a specific record. Once a maintenance specification has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Specification:** This zone displays information about the maintenance specification and identifies the maintenance schedule.
- **Specification Assets:** This zone lists all the assets associated with the specification record.
- **Attachments:** This zone lists attachments associated with the specification.

Maintaining Preventive Maintenance Schedules

The **Maintenance Schedule** portal is used to display and maintain a maintenance schedule record.

Refer to [About Preventive Maintenance](#) for more information.

You can access the portal from **Menu**, select **Preventive Maintenance**, then select **Maintenance Schedule**. You are brought to a query portal with options for searching for a specific record. Once a maintenance schedule has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Maintenance Schedule:** This zone displays information about the maintenance schedule record.
- **Maintenance Trigger:** This zone lists the triggers associated with the maintenance schedule record.
- **Asset List Maintenance Trigger:** This zone lists the asset list maintenance triggers associated with the maintenance schedule record.
- **Maintenance Specification:** This zone lists the specifications associated with the maintenance schedule record.
- **Activity:** This zone lists all the non-finaled activities associated with the maintenance schedule record.

The following zones may appear as part of the portal's **Analytics** tab page:

- **Failure Type Analytics:** This zone displays a graphical view of analytics for the maintenance schedule where corrective maintenance has occurred for the assets associated to the maintenance schedule with a failure type detailed.
- **Failure Mode Analytics:** This zone displays a graphical view of analytics for the maintenance schedule where corrective maintenance has occurred for the assets associated to the maintenance schedule with a failure mode detailed.
- **Failure Type and Failure Mode Analytics:** This zone displays a graphical view of analytics for the maintenance schedule where corrective maintenance has occurred for the assets associated to the maintenance schedule with a failure type and mode detailed.
- **Failure Type, Mode and Repair Analytics:** This zone displays a graphical view of analytics for the maintenance schedule where corrective maintenance has occurred for the assets associated to the maintenance schedule with a failure type, mode, and repair detailed.

Maintaining Preventive Maintenance Triggers

The **Maintenance Trigger** portal is used to display and maintain a maintenance trigger record.

Refer to [About Preventive Maintenance](#) for more information.

You can access the portal from **Menu**, select **Preventive Maintenance**, then select **Maintenance Trigger**. You are brought to a query portal with options for searching for a specific record. Once a maintenance trigger has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Maintenance Trigger:** This zone displays information about the maintenance trigger record, such as processing information, asset qualification conditions, and failure statistics.
- **Activity:** This zone lists the work activities associated with the maintenance trigger record.
- **Failure Information Quick Add:** This zone enables the addition of failure information to the maintenance trigger record.
- **Planned Failure Information:** This zone displays planned failure information associated with the maintenance trigger record.

Chapter 12

Construction Work

About Construction Work

Construction work involves such types of work as pole extensions, excavations, subdivisions, and so on and require planning and coordination between many different internal and external departments.

This section describes concepts related to construction work.

Refer to the [Understanding Construction Work](#) section in the **Admin User Guide** for more information.

Planning Construction Work

In a standard workflow, an organization begins by determining what materials, labor, and equipment are needed to install (construct) or remove (demolish or retire) an asset at a given location.

Compatible Units (CUs) are then configured based on the resources (materials, labor, and equipment) needed to complete all phases of the work.

Next, work designs are created to estimate the construction work using all of the materials, labor, equipment, and any other costs associated to the work. The compatible units are usually utilized on the design to define these requirements. Multiple design versions can be created so as to compare cost estimates and resource requirements.

Also, the term “asset”, in the context of construction work, is used to refer to fixed assets, also called capital assets. Not all fixed assets are Oracle Utilities Work and Asset Management assets; a record in the asset table with an asset ID. For example, you may not track wires as individual assets within Oracle Utilities Work and Asset Management. Whether an Oracle Utilities Work and Asset Management asset should also be created as a result of construction work is determined by the presence or lack of presence of an asset type on the CU definition.

Work Designs

Work designs are used for construction projects to produce one or many construction estimates with resource requirements. Designs can ensure that all necessary requirements are accounted for before the actual work begins. A single project can have multiple versions so that you can compare different approaches to the project and how they might affect costs. Resources included on the original work design are carried over to each new version where users can make the needed adjustments.

Design Elements

Design elements “build” a work design by indicating the compatible units and other resources that are needed for one construction location where one or many assets are planned.

Construction Work Costs

Costs are managed just as they are for regular work but with some additional details:

Work Design Estimates

The **Estimates** tab on the work design tabulates totals for all of the resource types, overtime, overhead, and any additional line items. Calculated totals include the following:

- **Customer Built Contributions:** Total derived value of customer supplied compatible units (display as credit)
- **Fixed Price Adjustment:** (Fixed Price – Derived Price)
- **Estimate Grand Total:** (Resource Cost + Additional Cost + Fixed Price Adjustment) – Customer Built Contributions
- **Overhead:** Work Design % per resource class first then apply the additional overhead % to the total

Direct Overhead Cost

An “Overhead” activity can be created to collect direct overhead costs associated with the work design and construction work order, such as planner time, easements, and permits, and then allocate those costs across the actual construction activities based on the actual construction costs during the closeout process.

Approving Construction Work

Once ready, work designs can be submitted for standard approval processing. Authorized approver can **Approve** the work design or **Withdrawal the Approval Request** and make changes or add new versions to the work design. Also, approved work designs can be reopened to add new design elements and then resubmit the design for approval.

Once approved, the work design can transition to a “work generated” status and construction work orders and construction work activities can be created from the work design to plan and manage the work. Additional construction work activities can be created on that construction work order or an existing construction work order that has not been associated to any work design as long as the construction work order is in a non-final state. Existing construction work activities that have been associated to the work design can also be updated from the work design.

Construction Work Orders

Construction work orders are a collection of work activities which manage the actual construction work and may or may not be related to compatible units. Construction work orders and construction work activities can be created directly or from an approved work design. Construction work activities can also be created from a work design for any existing construction work order that has not been associated to a work design as long as the work order is in a non-final state.

Construction Work Activities

Construction work activities manage construction work, typically involving compatible units (such as installing, removing, or maintaining a compatible unit). They can manage the creation of new assets resulting from construction, construction related repairs, or maintenance. Construction work activities can be created from a work design or can be created directly.

Activity type is shared between the activity classes (Field Activity, Work Activity, and Construction Work Activity) to control the purpose and configuration of the activity. A setting unique to both work activities and construction work activities is the “Construction Related” flag on the activity type BO which determines whether the activity is to be a construction work activity. Every construction work activity must reference a service class to identify whether or not the activity is an “overhead” activity or an activity used for construction. Overhead activities are created to collect direct overhead costs for planner time, easements, permits, and so on that are associated with the work design and construction work order. Construction work in progress (WIP) cost center on the service class is defaulted to the activity.

Each construction work activity references one or many construction locations which have one or more compatible units which define the work and resources needed at the location.

Construction Work Locations

Construction locations define either a work location or an asset location (not both). Work locations identify where compatible units are used to build new assets. Typically, a work location will become an asset location once the asset is constructed. When work is done on an existing asset, the identified construction location should be identified as an asset location. If construction work occurs at a construction location that is known to cost more in labor, an **Overtime Type** can be specified to apply to all crafts on the construction location. For example, construction work on a major commute bridge can only occur at night (versus regular hours) to minimize traffic disruptions. Construction Locations are added and accessed from the **Construction Location Details** zone on the **Construction Work Activity** portal **Resource** tab.

Scheduling Construction Work

Construction work activities are scheduled from within the **Work and Asset Management** application using the **Scheduling** portal to schedule the labor requirements associated with the construction work activity.

Construction Work Activity Reconciliation

An activity reconciliation is created when a construction work activity is closed. It is created to summarize costs and identify assets that were by created or contributed to the project. It immediately gets routed for approval and placed in *Pending Approval* status. Typically, the first person on the approval route will review and update the activity reconciliation. The approver can make updates to the reconciliation record, as needed, such as updating cost distribution or adding construction or contributed assets. Once the record is finalized and approved, the reconciliation record moves to *Closed*, and financial transactions are created. Any constructed assets or customer contributed assets are recorded for export to integrated fixed asset systems, as applicable.

Communication Logs

Communication Logs show all the communications made against a work document and its related work documents. Refer to [About Communication Logs](#) for more information.

Maintaining Work Designs

The **Work Design** portal is used to display and maintain a work design record.

Refer to [About Construction Work](#) for more information.

You can access the portal from **Menu**, select **Construction Work Management**, then select **Work Design**. You are brought to a query portal with options for searching for a specific record. Once a work design has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Work Design:** This zone displays information about the work design, such as required by date, originating designer, and handling details for the design elements.
- **Design Elements:** This zone lists the design elements associated with the work design record.
- **Contractor Bids:** This zone lists contractor bids associated with the work design record.
- **Construction Work Order:** This zone displays a hierarchical view of the construction work orders and activities created from the work design.
- **Attachments:** This zone enables users to manage all types of attachments for the work design record.
- **Work Design Notes:** This zone displays any note information associated with the work design record.
- **Contact Relationship:** This zone displays any contact information associated with the work design record.
- **Communication Log:** This zone displays any communications associated with the work design record.

The following zones may appear as part of the portal's **Estimate** tab page:

- **Work Design Estimate:** This zone displays all cost associated with the work design.

The following zones may appear as part of the portal's **Compatible Units** tab page:

- **Work Design — CU Assignment:** This zone enables users to search for and then add compatible units to multiple design elements.

The following zones may appear as part of the portal's **Log** tab page:

- **Work Design Log:** This zone displays log details for the work design.
- **Approval Request History:** This zone lists the approval history for the current work design.

Maintaining Construction Work Orders

The **Construction Work Order** portal is used to display and maintain a construction work order record.

Refer to [About Construction Work](#) for more information.

You can access the portal from **Menu**, select **Construction Work Management**, then select **Construction Work Order**. You are brought to a query portal with options for searching for a specific construction work order. Once a construction work order has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Work Order:** This zone displays information about the work order, such as required by date, work details, and access controls.
- **Construction Work Order Activity Info:** This zone displays construction work order activity information for the construction work order record.

- **Follow Up Work Order and Work Request:** This zone displays follow up work orders and work requests for the construction work order record.
- **Notes:** This zone displays any note information associated with the construction work order record.
- **Scheduled Activities:** This zone displays the scheduled activities associated with the construction work order record.
- **Contact Relationship:** This zone displays any contact information associated with the construction work order record.
- **Communication Log:** This zone displays all communications associated with the construction work order record.

The following zones may appear as part of the portal's **Cost** tab page:

- **Work Order Cost Summary:** This zone displays a summary of cost information for the work order.

The following zones may appear as part of the portal's **Log** tab page:

- **Work Order Log:** This zone displays log information for the construction work order.
- **Approval Request History:** This zone lists the approval requests for the current construction work order.
- **Related Stock Transactions:** This zone displays the related stock transactions for the construction work order.

Maintaining Construction Work Activities

The **Construction Work Activity** portal is used to display and maintain a construction work activity record.

Refer to [About Construction Work](#) for more information.

You can access the portal from **Menu**, select **Construction Work Management**, then select **Construction Work Activity**. You are brought to a query portal with options for searching for a specific construction work activity. Once a construction work activity has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Work Activity:** This zone displays information about the work activity, such as work site details, reconciliation information, special handling, and schedule details.
- **Checklists:** This zone lists the checklists associated to the construction work activity record.
- **Work Order Activities:** This zone lists the activities associated with the construction work activity record.
- **Notes:** This zone displays note information associated with the construction work activity record.
- **Attachments:** This zone enables users to manage all types of attachments for the construction work activity record.
- **Scheduled Labor Requirements:** This zone displays the labor requirements for the construction work activity record.

The following zones may appear as part of the portal's **Resources** tab page:

- **Quick Add:** This zone enables additional resources to be added to the construction work activity record.
- **Activity Resource Action:** This zone displays the construction work activity's resources. Users can select the resource hyperlink and go to the activity's resource requirement to make changes.

The following zones may appear as part of the portal's **Cost** tab page:

- **Activity Cost Summary:** This zone displays a summary of costs associated with the construction work activity record.

The following zones may appear as part of the portal's **Permit** tab page:

- **Activity Permit Checklists:** This zone displays any permit checklists associated with the construction work activity.
- **Permit Attachments:** This zone displays permit attachments associated with the construction work activity.
- **Service History Types:** This zone lists planned service history types for the construction work activity, indicating for each service history type whether it is required to be filled out or not. When an asset is in context the list shows all valid service history types for the asset.

- **Service History:** This zone lists service history records associated with the construction activity and the asset in context.

Maintaining Activity Reconciliations

The **Activity Reconciliation** portal is used to display and maintain an activity reconciliation record.

Refer to [About Construction Work](#) for more information.

You can access the portal from **Menu**, select **Construction Work Management**, then select **Activity Reconciliation**.

You are brought to a query portal with options for searching for a specific activity reconciliation record. Once an activity reconciliation record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Activity Reconciliation Display:** This zone displays information about the activity reconciliation, such as constructed assets, cost distribution, and approval profile.

The following zones may appear as part of the portal's **Financial Transactions** tab page:

- **Activity Reconciliation Financial Transaction:** This zone lists the financial transactions associated with the activity reconciliation.

The following zones may appear as part of the portal's **Log** tab page:

- **Activity Reconciliation Log:** This zone lists log information for the activity reconciliation.
- **Approval Request History:** This zone lists the approval requests associated with the activity reconciliation.

Chapter 13

Scheduling

About Scheduling

The goal of the scheduling process is to produce the most effective schedule based on the requirements of the work. Scheduling accomplishes this goal by assigning tasks to crew shifts, matching work requirements to crew capabilities.

Planners can use the **Scheduling** portal to search for and to review new activity labor requirements and unfinished back log work in order to make allocations to available crew shifts. Users can also use this portal to view the capacity of shifts based on crafts. First search for the appropriate crew shift, then use the zones to search for activities, review the crew members, or access a graphical representation of craft capacities.

NOTE: Planners have the ability to add work activities to active crew shifts. Refer to [Break In](#) for more information.

Planners can also use the **Scheduling** portal to search for an activity scheduled to a shift. From the activity they can then add a single employee into the schedule or schedule more than one employee at a time.

Daily and Weekly Shifts

Weekly and Daily shift schedules are planned periods of time used to manage crews as they perform their work. Refer to [Daily Versus Weekly Schedules](#) for more information.

Shift Plan Templates

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. Refer to [About Shift Plan Templates](#) for more information.

Scheduling Work

The system schedules an activity to a crew shift, taking into consideration the activity's labor requirements and estimated duration. Refer to [The Scheduling Process](#) for more information.

Finding Backlog Work and Assigning

Use the **Scheduling** portal or **Crew Shift** portal's **Scheduling** tab to search for all unfinished and active labor requirements for the selected crew. Unfinished labor requirements are those which are still active but linked to completed or canceled crew shifts. When there are active labor requirements pending for the crew shift, the system provides a prompt where you can allocate the labor to the next available shift. Follow the prompts to complete the process of allocating the labor requirements either to the next available shift, or to a different crew shift.

The Scheduling Process

The goal of the scheduling process is to produce the most effective schedule based on the activity's labor requirements.

The system considers the following during the scheduling process:

- Weekly and daily shift templates which are used to generate shifts into the future. Shift scheduling also takes into consideration work calendars and holidays.
- Capacities of a shift which is the sum of all the available principle and secondary crafts and their respective number of hours.
- Schedule Priority which determines when the principal craft time should be used for making assignments to a secondary craft.

How is the work scheduled?

- Once an activity is created with labor requirements, Planners or other authorized users can schedule work by navigating to either the **Schedule** tab on the crew shift or to the **Scheduling** portal. In either case, once the crew shift is identified, search for the activity to be scheduled, check the appropriate shifts and click **Apply**. A dialog prompts for the scheduled hours for each labor requirement on the activity.
- Planners can review the shift assignment in the **Scheduling Details** section on the activity. There are also multiple other ways to search for labor active work such as by activity, by work order, by location, by crew, and so on.
- Using the **Schedule** portal, planners can pull already scheduled work from another shift. For example, the search can be used to transfer assigned work from a weekly shift to a daily shift. Furthermore, the **Previously Scheduled Unfinished Work** zone displays work already assigned to the crew shift being scheduled.
- Each of these various details and views can be used according to your business practice to create the most efficient scheduling process for your organization.
- In addition, when a shift is completed, all unfinished activities assigned to the shift are copied to the next open shift for the crew. A dialog prompts the user for details on expected duration left to complete the work.

Daily Versus Weekly Schedules

A weekly shift is a weekly schedule plan containing a collection of daily shifts. A daily shift is the plan for a single shift on a specific day.

Please note the following with respect to shifts:

- Weekly shifts are referred to as “schedule plans” in the system since the plan can come in any multiple of weeks.
- A weekly shift's capacities, service areas, and service classes are derived from its set of daily shifts.
- A weekly shift specifies a type, as does a daily shift
- A weekly shift specifies a crew name, as does a daily shift
- Both a weekly shift and daily shift can be scheduled
- A weekly shift does not specify employees, as daily shift does (instead, it has a collection of daily shifts)

- A weekly shift does not specify a template, as a daily shift does
- A weekly shift can have many daily shifts attached to it - just like a week has many weekdays
- A daily shift can only specify one weekly shift - just like a specific day belongs to just one week

Break In

There are times when users will need to modify the work on a crew shift that is already activated. The activities that are added to an active shift schedule are known as Break In activities and they essentially override the original schedule.

The **Activity Labor Requirements** zone on the crew shift record is stamped with the date and time of the break in activity.

Planners can also use the **Activity** portal to manually assign an active shift directly to a work order activity when the following conditions are true:

- Activity is in either an *Approved* or *Active* status
- None of the activity's labor requirements has been scheduled to any shift.

Planner would click **Add to Schedule** and search for and select a shift once the dialog box appears.

About Crew Shifts

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 a van, working on a specific date from 9am — 5 pm.

Refer to [Understanding Scheduling](#) for more information.

This section describes concepts related to managing crew shifts.

Creating Shifts

Typically, crew shifts are generated by shift plan templates that define the basic characteristics of the shifts. A crew's shifts may be generated by a single shift weekly template or daily shift templates at any given point in time.

Refer to [Understanding Shift Plan Templates](#) for more information.

Shifts can also be manually created to address special situations on particular days.

Service Area

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.

Crew Shift Workflow

The following describes key stages in the overall lifecycle of a crew shift.

Planning

A shift in this state means that the schedule is still being planned and activity resource requirements being added.

Active

A shift in an active status means that the shift's schedule is approved for work. Any schedule changes made at this state is considered "[break in](#)".

Incomplete

A shift having an incomplete status means that the shift has reached its planned start date and time and some work scheduled to it is not yet completed nor covered by a future schedule.

Completed

A shift in a completed state means that it has reached its planned end date and time and all the work scheduled to it is either completed or already scheduled to a future shift.

Cancelled

This state indicates the shift is canceled and should no longer be used for scheduling. If any activities are currently scheduled to the shift, they need to be rescheduled.

A user or any process can cancel a shift that has not started yet. Once started, it can only be replanned or completed.

Planned shifts that were not started before their planned end date and time are automatically canceled.

Controlling the Schedule

This section describes the various aspects of a shift used by the scheduler to control the crew's schedule of work on that shift.

Fixed Shifts

By default, the duration of a shift is fixed and defined as the time between its planned start and end date and times. The crew is assumed to start their shift at the planned start time and end it at the planned end time. While typically shifts are shorter than a day, there is no such limitation in the system; they can be set up to any length as the business requires. For example, night shifts could be shorter than day shifts or special day shifts could be longer than regular day shifts.

Employee Unavailability

Employee unavailability records establish periods of time where an employee has been given leave from their regular job duties. This might be due to circumstances such as vacation, sick leave, 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 they are associated with and to create their timesheet, if desired, during the time when they are unavailable. Users can enable automatic timesheet creation for employee unavailability, by selecting the **Auto Timesheet Creation** box on the **Employee Unavailability** record and entering the **Date**, **Hours**, and **Timesheet Detail** information in the **Timesheet List** section. Timesheets are only created for an "Approved" unavailability period if the **Timesheet Date** is less than or equal to the **Current Date**. Then the daily scheduled *Employee Unavailability (WI-EU)* batch would create the records going forward.

Employee unavailability types define the kinds of leave that might be used when creating an **Employee Unavailability** record.

Standard approval processing facilitates approval or rejection of employee leave. After the leave is approved, 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.

Employees who need to cancel their unavailability are able to do so by selecting **Cancel** on the unavailability record. The unavailability is then ignored during the scheduling process and the Craft Capacity and Available hours are increased so that the employee is once again considered for the shift.

Maintaining Crew Shifts

This portal is used to view and maintain crew shifts.

Refer to [About Crew Shifts](#) for more information.

You can access this portal from **Menu**, select **Resource Management**, and then select **Crew Shift**. You are brought to a query portal with options for searching for specific crew shifts. Once the crew shift has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portals **Main** tab page:

- **Crew Shift:** This zone provides information about the selected crew shift, including crew status, crew shift type, subscriptions, crew members, service classes, service areas, and more.
- **Activity Labor Requirements:** This zone displays the work activity and labor requirements associated with the crew shift.
- **Mobile Subscriptions:** This zone displays a list of employees currently subscribed to the crew's daily shift from their mobile device.

The following zones may appear as part of the portals **Schedule** tab page:

- **Crew Shift Craft Capacity:** This zone displays a graphical representation of the various crafts, craft capacities, and remaining hours for the particular crew shift.
- **Crew Members:** This zone lists the crew members and their relationship to the crew. For example, crew member or supervisor .
- **Activity Search:** This zone displays a query portal with options for searching for activities.

Maintaining Employee Unavailability

This portal is used to view and maintain employee unavailability.

You can access this portal from **Menu**, select **Resource Management**, and then select **Employee Unavailability**. You are brought to a query portal with options for searching for a specific employee unavailability record. Once the employee unavailability has been selected, you are brought to the maintenance portal where you can view and maintain the selected record.

The following zones may appear as part of the portals **Main** tab page:

- **Employee Unavailability:** This zone provides information about an employee's leave including definitions for creating a timesheet during the unavailability period.

Scheduling Portal

This portal helps planners manage craft capacities and activity labor requirements for crew shifts.

You can access this portal from **Menu**, select **Work Management**, and then select **Scheduling**. You are brought to a query portal with options for searching for a specific crew shift. Once the crew shift has been selected, you are brought to the **Scheduler** portal where you can view and manage the craft capacities and labor requirements for the selected record.

The following zones may appear as part of the portals **Main** tab page:

- **Craft Capacity.** This zone displays a graphical representation of the various crafts, craft capacities, and remaining hours for the particular crew shift.

- **Activity Search.** This zone displays a query portal with options for searching for activities.
- **Scheduled Labor Requirements.** This zone displays the work activity, labor requirements, and crew allocation associated with the crew shift.

Refer to [About Scheduling](#) for more information.

Chapter 14

Approvals

About Approvals

Approvals simplify the routing of work orders, purchase requests, timesheets, and other documents that require approval.

Approval requests use approval profiles to define the requirements for the approval:

- Who receives the approval
- Method for notifying the approver
- Which work calendar to use

When the approval request is submitted, the system sends a notification to all approvers defined on the approval profile according to the **To Do Roles** and the **Sequence** defined on the profile. Notifications can be sent using email, To Do List, or both, depending on the system notification settings.

Approvers can use the **Approval Dashboard** to manage the approvals that require their attention.

This section describes how various processes work together to manage approvals throughout the system.

Refer to [Maintaining Approval Requests](#) in the **Business User Guide** for information on maintaining approval requests.

Refer to [Understanding Approval Processing](#) in the **Administrative User Guide** for more information.

Approval Profiles

Approval profiles are used to define the approval hierarchy within your organization and controls which approvers are required to approve the record.

Approval Profiles based on **To Do Roles** define:

- the approvers level of authority to approve a document
- 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 can receive an email, an alert, or both 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.

Approval Routing

The approval request, once submitted, is routed for approval based on the roles of the approvers and their approval decisions. For example:

- **One Standard Role:** If the approval profile consists of one To Do Role with mandatory or standard approval type, the users with that role receive the approval notification and any one of them can approve the request as long as the To Do Role has authority for the needed dollar amount.
- **Multiple Roles/All Standard:** If the approval profile consists of multiple To Do Roles and all are standard, any To Do Role can approve the request as long as they have the authority for the needed dollar limit.
- **Mix of Mandatory and Standard Roles:** If the approval profile consists of multiple To Do Roles with a mix of mandatory and standard, the users with mandatory must approve the request even if the standard approvers have authority based on the dollar limit or other factors. In other words, the mandatory approval role must take an action by agreeing or rejecting the request.
- **Request Time-out:** If approvers do not respond to the request within the time allotted by the time-out threshold, the system sends another notification to the approvers. If action is still not taken, the request moves to the next approval role, as applicable and the next role receives notification.
- **One Approver with Multiple Roles:** If one approver has multiple To Do Roles on one approval profile for the request, the system prompts that person to indicate which role they are approving. For example, approver Jon is in both the supervisor and manager To Do Roles. When an approval request comes in and Jon approves the request, a selection box opens to allow Jon to choose in which **To Do Role** the request is being approved. If Jon selects multiple approval requests to approve at once in the **Approval Dashboard** Jon may receive a warning to specify an **Acting To Do Role** before selecting an action. This means you must pick a role from the **Acting To Do Role** drop down and click **Search** to let the system know which role you are using to approve the requests.
- **Insufficient Approval Amount:** If an approval request comes in for a record with a higher dollar amount than anyone on the approval path is authorized to approve, the requestor is notified that no one on the approval profile has sufficient authority. The requestor must choose a different approval profile and resubmit.
- **One Standard Role:** If an approver rejects the approval request, the request terminates and additional approvers on the approval path are not notified. The requestor is notified that the request has been rejected. Any approver in the route can reject and terminate the approval process regardless of their approval authority or limit amount. However, this does not apply for “notification only” approvers.
- **Not Authorized to Approve:** Some roles might be set as “not authorized to approve” but they still appear on the approval path. These approvers receive notification of the request and can approve or reject, but their decision is simply recorded for the next approver on the approval path to be aware of their decision. This setting is only applicable for an approval profile that is for no amount.
If this type of approver rejects the request, the same processing applies as described for “one standard role”.

Outstanding Approvals at a Glance

The **Approval Dashboard** provides approvers with a quick glance, by document type, of the status of approvals that are currently due or are now past due. Approvers can then take appropriate action to reject or approve the requesting document.

Also, the system includes an **Approval Request Action** dashboard zone in the side dashboard which highlights any new approval requests that require the logged in user's attention. The user can approve or reject the request directly from this dashboard, if desired.

Approval Dashboard

The approval dashboard provides a summary of approval requests that have been sent to the broad list of approvers with a breakdown of the approval profiles. This overview provides approvers with a simple summary of the records requiring their attention.

The summary list includes the counts of approval requests that have been sent out to any approvers that are not **Notification Only** and shows the current approvers and the mandatory future approvers. The list is broken down by document type and approval profile.

It is possible to filter the list by current or future approvers using the **Queue** field. The filter is used to distinguish current and future approval requests which may require the attention of the approver. For example, if an approver is about to go on vacation, they may want to take action on an approval request early. In this case the list can be filtered by "future" approval requests where this approver is not necessarily next in the approval route. This is a way for the approver to obtain a "preview" of requests that are coming from the route.

Broadcasting an individual line opens a zone where the approver can check off records and choose a button to make their approval decision (approve or reject). The approver can also click the link to open the approval request itself, and to open the individual record.

Note that the dashboard contents are specific to the logged in user, so one person may see a different list than another depending upon their approval authority.

Refer to [About Approvals](#) for more information.

Maintaining Approval Requests

This portal is used to view and maintain approval requests.

Refer to [About Approvals](#) for more information.

You can access the portal from **Menu**, select **Home Page**, then select the **Approvals** tab. Broadcast the **Document Type** and then select the **Approval Request ID** hyperlink from within the **Approval Detail List** zone.

The **Approval Request** zone on the portal's **Main** tab page provides information about the approval request, including approvers, approval amounts, approval profile, etc.

About Change Requests

Change requests simplify the process associated with managing changes respective to assets, asset locations, vendors, vendor locations, stock items, stock item details, and any other records as appropriate.

The **Change Request** portal controls the full cycle of the change request, from request creation, review, approval, application, and closure. The portal includes an **Original** zone, which holds all the original data elements, and a **Proposed** zone, which holds all the changed data. The updated data elements are saved as temporary until the change is approved.

When the change request is submitted for approval, the system sends a notification to all approvers defined on the approval profile according to the **To Do Roles** and the **Sequence** defined on the profile. Notifications can be sent using email, To Do List, or both, depending on the system notification settings.

Approvers can use the **Approval Dashboard** to manage the change request approvals that require their attention.

Refer to [Maintaining Change Requests](#) in the **Business User Guide** for information on maintaining change requests.

Refer to [Understanding Change Request Types](#) in the **Administrative User Guide** for more information.

Change Request Approval Profiles

Change request approval profiles are used to define the approvers who are required to approve a change request record.

Approval Profiles based on **To Do Roles** define:

- the approvers level of authority to approve a record
- the types of documents the role can approve
- the working calendar to use when determining approval time lines and when to send To Dos

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. Valid types of approval profiles include the following:

- Non-Amount Based
- No Approval Required

Depending on the notification setting designated on the approval profile, the approver can receive an email, an alert, or both 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**.

Change Request Approval Routing

The change request approval request, once submitted, is routed for approval based on the roles of the approvers and their approval decisions. For example:

- If the approval profile consists of one **To Do Role** with mandatory or standard approval type, the users with that role receive the approval notification and any one of them can approve or reject the request.
- If the approval profile consists of multiple **To Do Roles** and all are standard, any valid **To Do Role** can approve or reject the request.

The approver is able to review the proposed changes and approve or reject the change request. Rejected requests can be resubmitted for approval once the issue is resolved. Once approved, a user can click the **Apply Changes** button to apply the change. This action completes the change request and updates the appropriate record.

Maintaining Change Requests

The **Change Request** portal is used to display and maintain a change request record.

Refer to [About Change Requests](#) for more information.

You can access the portal from **Menu**, select **Resource Management**, then select **Change Request**.

- If **Add** is selected:
 - Only a “Generic” change request can be created.

NOTE: Creation of any other type of change request must be on the change request associated record. Once on the associated record, user permissions drive who is able to create the change request. Users having full permissions can select **Edit** to make their changes or select the **Create Change Request** button directly from the **Dashboard**. Users having limited permissions can only create a change request by selecting the **Create Change Request** button directly from the **Dashboard**. Once the change request is created, users having full permissions can view the change request by going directly to the associated record and clicking the change request hyperlink from the zone on the **Dashboard**.

- If **Search** is selected:
 - User is brought to a query portal with options for searching for a specific change request. Once a change request has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Change Request:** This zone displays information about the change request, including the description of the change and any warning information.
- **Original:** This zone displays the original details of the request.
- **Proposed:** This zone initially displays original data. Selection of an **Edit** hyperlink enables the user to update the data to proposed values. This zone highlights the updated **Field** names and also the **List Headers** on the associated record if the content of the list has changed.
- **Attachments:** This zone enables users to manage all types of attachments for the change request record.

Chapter 15

Resource Usage

About Timesheets

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 these workers to receive proper compensation for their work and labor charges are applied to the appropriate cost buckets.

Generally, each of these individual workers and the persons designated as their supervisor are the only ones who can access or enter timesheet information for that individual, however, authorized “timekeepers” who are not supervisors of these individuals can be configured to manage timesheet information for these workers. Refer to [Timekeepers](#) for more information.

Timesheet Access

Access to an employee’s timesheet is controlled in the following manner:

- Employees can always access their own timesheet
- An employee having a supervisor role can access any of their employee’s timesheets. This access is hierarchical such that a supervisor’s designated supervisor is also able to access timesheets for lower-level employees.
- Users can be assigned to the role of Timekeeper and will have access to all employee’s timesheets listed on the *Active* Timekeeper record.
- Users can be designated as a special group, usually payroll employees, on their User record that gives them access to all employee timesheets.

Unrestricted Timesheet Search (WIUTMSTS) application service can be associated with a User Group which allows any employee to search for any employee and create or access their timesheet. The employee is not restricted to a list of valid employees they are responsible for but can simply click the **Search** icon next to the **Employee** field and search for and create or access timesheets for that employee.

Shift Types

When workers book time, they also specify the shift type they worked. Different shift types have different multipliers and adders which impact the labor costs. Timesheets can only specify daily shift types (as opposed to weekly shifts).

To restrict access to only those employees that are valid for the Employee, remove the Unrestricted Timesheet Search (WIUTMSTS) application service from the User Group to provide a drop-down list of valid employees the employee can create and edit timesheets for.

Timesheet Quick Entry

A new **Timesheet Quick Entry** portal allows a user to enter any number of timesheet detail entries for any number of employees and for any charge types, for example activity, cost center, project, leave, and process function. The employees available to be chosen by the logged on user is defined as below:

- The employee record of the logged on user
- If the logged on employee is a supervisor, all of their employees
- If the logged on user is a timekeeper, employees on the timekeeper record
- If the logged on user has payroll security authorization, any employee

New timesheet details can be entered manually, or user can create a .csv file in which to upload the new entries. See [Creating CSV Files](#) for more information.

Once the entries are complete, the system checks whether there is an existing timesheet for the same time period for each employee:

- If yes, the system creates a new timesheet detail under that same timesheet header.
 - If in a **Created** status, new line items are added in **Created** status.
 - If in an **Approved** status, timesheet is returned to **Created** status, existing line items are left alone, and new line items are added in a **Created** status.
 - If in **Posted** status, timesheet is returned to **Created** status, existing line items are left alone, and new line items are added in a **Created** status.
- If no, new timesheet header is created and timesheet details for the same employee and time period are added to that timesheet header.

Timekeepers can use the **Timesheet Quick Entry Search** option to search for timesheet entries they have created.

NOTE: This search option restricts timekeepers to only those timesheet entries they have created.

About Resource Charges

Each activity on a work order is defined with one or more resource requirements which is needed to perform the activity. Resources are referenced on the activity so the system can track what is charged for each resource against the total cost of completing the activity.

Equipment Charges

Equipment is 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.

Other Direct Charges

Other direct charges are defined by your organization to be referenced as a charge type to charge work against. This might include miscellaneous charges for activities such as meals, miles, a hotel stay, car or truck rentals, and so on.

Other direct charges are incurred on activities as a material resource requirement from “miscellaneous resources”. The system tracks usage and reports the costing associated with that usage against the total cost of completing the activity.

Maintaining Timesheets

The **Timesheet** portal is used to display and maintain a timesheet record.

Refer to [About Timesheets](#) for more information.

You can access the portal from **Menu**, select **Work Management**, then select **Timesheet**. You are brought to a query portal with options for searching for a specific timesheet. Once a timesheet has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Timesheet Display:** This zone displays information about the timesheet. This zone displays the status, employee, and time period of the timesheet. Timesheet summary section displays a chart that includes the craft and regular, overtime, and total hours summarized for the timesheet.
- **Timesheet Detail Quick Add:** This zone enables a user to quickly enter timesheet details.
- **Timesheet Details:** This zone displays information from the timesheet details.

Maintaining Timesheet Quick Entry

The **Timesheet Quick Entry** portal is used to create timesheet detail records.

Refer to [About Timesheets](#) for more information.

You can access the portal from **Menu**, select **Work Management**, then select **Timesheet Quick Entry**. You are brought to a query portal with options for searching for your timesheet quick entry records. Once a record has been selected, you are brought to the maintenance portal where you can create, view, and update the associated timesheet detail records and then process and complete the entry using the completion event to create the appropriate timesheet header and timesheet detail records.

The following zones may appear as part of the portal's **Main** tab page:

- **Timesheet Quick Entry:** This zone displays the status and new timesheet detail records created by the user.
- **Timesheet Quick Entry Completion Event:** This zone displays timesheet quick entry completion events associated with the record.

Maintaining Other Direct Charges

The **Other Direct Charges** portal is used to display and maintain an other direct charge record.

Refer to [About Timesheets](#) for more information.

You can access the portal from **Menu**, select **Work Management**, then select **Other Direct Charge**. You are brought to a query portal with options for searching for a specific other direct charge. Once a record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Other Direct Charges:** This zone displays information about other direct charges including an other direct charges details summary.
- **Other Direct Charges Detail Quick Add:** This zone enables a user to quickly enter other direct charge details.
- **Other Direct Charges Details:** This zone displays other direct charge detail information.

Chapter 16

Financials

About Financial Transactions

Financial transactions are created in the system whenever an object generates a cost transaction. For example, the posting of a timesheet or other direct charges can create a financial transaction. Authorized users can access this accounting log at any time to review transactions.

Users can navigate to the **Financial Transaction** portal and search for financial transactions according to the corresponding work order, activity, creation date, status, and so on.

Financial transactions are automatically frozen upon creation and the transactions that created them, sibling transactions, are finalized. For example, a material issue or a timesheet detail are considered the siblings to the financial transactions that they created. Once frozen, a financial transaction and its details are ready to be interfaced to the general issues. The system does not depend on batch processing to transition financial transactions between states. Instead, a batch GL interface program extracts frozen financial transactions and creates a flat file.

If there happens to be any accounting setup issues or other issues, a financial transaction will enter an error state until an authorized user adjusts the configuration or other source of the issue. Refer to the **Administrative User Guide** for more information. The error state allows any sibling transactions to continue and complete without being affected by the related transaction which is in error.

Refer to the [Understanding Financials](#) section in the **Administrative User Guide** for configuration guidelines.

About Cost Adjustments

Cost adjustments enable the correction of erroneous applied charges. Costs can be transferred from one activity to another, project to project, cost center to cost center, activity to cost center, and so on.

Each line item represents one entity, for example, activity, project, or cost center involved in the transfer of charges.

Maintaining Financial Transactions

The **Financial Transaction** portal is used to display and maintain financial transaction records.

Refer to [About Financial Transactions](#) for more information.

You can access the portal from **Menu**, select **Financial**, then select **Financial Transaction**. You are brought to a query portal with options for searching for a specific transaction. Once a record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Financial Transaction:** This zone display information about the financial transaction such as, currency code, accounting date, GL details, and more.

Maintaining Cost Adjustments

The **Cost Adjustment** portal is used to display and maintain cost adjustment records.

Refer to [About Cost Adjustments](#) for more information.

You can access the portal from **Menu**, select **Financial**, then select **Cost Adjustment**. You are brought to a query portal with options for searching for a specific cost adjustment. Once a cost adjustment has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Cost Adjustment:** This zone lists information about the cost adjustments and details such as cost center, expense code, amount, and so forth.
- **Attachments:** This zone displays the attachments associated with this specific record.

The following zones may appear as part of the portal's **Financial Transaction** tab page:

- **Cost Adjustment Financial Transaction:** This zone lists the financial transactions for the cost adjustment.

Chapter 17

To Do Processing

There are numerous events that trigger the creation of email-like messages describing work that requires human attention. For example, if a bill segment has an error, a message is created to highlight such. We refer to each message as a **To Do Entry**. In this section, we describe how to view, manage and work To Do entries.

FASTPATH: We recommend familiarizing yourself with the topics described in [The Big Picture of To Do Lists](#) to fully appreciate how To Do entries are managed by the system.

Understanding To Do Processing

The topics in this section provide background information on To Do processing functionality.

To Do Role

Each To Do entry is assigned a specific To Do Role. The role defines the users who may work on the entry. The permissible roles that may be assigned to a To Do entry are defined on the entry's To Do type. One or more users within a role may be designated as a supervisor, who is allowed to assign To Do entries to other people. Supervisors are determined through application security access.

Refer to [To Do Entries Reference A Role](#) to learn more about role assignment.

After an entry is created, its role may be changed to any role defined as valid for the entry's To Do type. Manual changes of a To Do entry's role are recorded on the [To Do entry log](#).

To Do Priority

A To Do entry is associated with a priority level representing its processing order compared to other entries. Refer to [The Priority Of A To Do Entry](#) for more information on how an entry's priority is initially assigned and how it may be automatically updated by the system.

A user may manually override a To Do entry's priority at any time. Once overridden, automatic recalculation of the priority is no longer performed so as not to override the value explicitly set by the user. Manual overrides of a To Do entry's priority are recorded on the [To Do entry log](#).

NOTE: Only users that are currently assigned to a **Being Worked On** To Do entry or ones that are valid for the role of an **Open** To Do entry may manually change the priority.

Work Distribution

This topic goes over the various possible actions a user can take to adjust what role or what user is associated with a To Do entry while it is in a non-complete state.

Adjusting the Role

A To Do entry must always reference a role. When a To Do entry are created in the **Open** state and is associated with a [role](#). The following points highlight ways to adjust the role associated with a To Do entry.

- A user viewing a record on the [To Do Entry](#) page may use the Forward or Send Back action to change the role.
- A supervisor using the [To Do Search](#) page may assign a user not linked to the current role. In this case, the system will update the role to one valid for the assignee.

Assigning a To Do to a User

There are several ways to assign a To Do entry to a user (causing its status to transition to **Being Worked On**):

- The [To Do Management](#) portal provides many filters and viewing options to review To Do entries for a give To Do Type. In addition, it displays a rich amount of detail for the resulting To Do entries, helping users to make decisions. Users may use this page to assign work to themselves and supervisors may use this to assign work to others.
- A supervisor can use the [Supervisor Functions](#) pages to review, assign and reassign To Dos to users.
- A user viewing an entry on the [To Do Entry](#) page may use the Forward to a User or the Send Back action to assign the entry to a different.
- A user may use the [To Do list](#) pages to review and assign To Dos to themselves.
- All of the above assignment functions may also be performed using the [To Do Search](#) page.

The system can also be configured to dynamically assign the highest priority work to users on demand whenever they are available for fixing errors.

NOTE: Assignment on demand is in a plug-in. You can customize the next assignment plug-in on the [installation options](#) to follow your organization's work distribution rules. The product provides an algorithm out of the box (**F1-NEXT-TDNS**) that can assign work solely based on priority. Another algorithm (**F1-NEXT-TDSK**) may be used to only consider To Do entries the user has the proper [skills](#) to resolve. Your implementation may provide a different algorithm based on your own business rules.

One of the main benefits of the assignment on demand approach is that work is allocated real-time based on most current information such as skills, priority, user availability, etc. This also means that work remains unassigned until a user is actually going to work on it making it available to any qualified user.

Assignment "on demand" is possible from the [Current To Do](#) zone on the Dashboard portal. Refer to this zone to learn about all the functions it provides.

Reopening / Unassigning a To Do

It is possible to transition a To Do entry from **Being Worked On** back to **Open**.

- A user viewing a record assigned to them on the [To Do Entry](#) page may use the Forward to a role action to move the entry back to **Open**.
- The **Unassign** button on the [To Do Management](#) portal or the **Reopen** button on the [To Do Search](#) page causes the To Do entry to go back to **Open**.

A User's Current To Do

When you are actively working on a specific To Do entry, there are tools to make this To Do the one that is displayed in the [Current To Do](#) dashboard zone. The act of making a To Do entry your current To Do, also navigates you to the related object (or if the To Do type is configured to launch a script, the script is launched).

If your organization's practice is to assign work to users up front, then a typical approach is that you would start by reviewing a list of entries assigned to you using one of the base pages and drill down on an entry in the list to start working on it. In some pages, there is a way for you to enable a list of To Do Entries that you can work through without needing to return to the query page to get the next entry.

The following points highlight the available base pages and their supported functionality.:

- [To Do Management](#) allows you to use filters to view a list of entries assigned to you or open. (If you plan to work on a To Do entry that is open, you should first assign it to yourself. No action automatically assigns it.) Click the Work List icon in the Action column to populate the [Work List](#) zone in the dashboard with the results of the search. Click the **Work** button to choose the To Do entry you wish to start with. This drills you into the related object and populates the Current To Do zone with this To Do. Use the entries in the Work List zone to subsequently work on other To Do entries.
- [To Do List](#) allows you to start working on a To Do by clicking the entry's message description link. If the To Do entry is Open, this action will assign it to you automatically. Any other To Do entries that were in that results list on that page become an internal work list and the **Previous** button and **Next** button are enabled in the Current To Do zone to work through that list.
- [To Do Search](#) allows you to start working on a To Do by clicking the entry's message description link. (If you plan to work on a To Do entry that is open, you should first assign it to yourself. No action automatically assigns it.) Any other To Do entries that were in that results list on that page become an internal work list and the **Previous** button and **Next** button are enabled in the Current To Do zone to work through that list.

Before working on a new To Do, if the problem is resolved, use the actions in the Current To Do zone to mark the entry [Complete](#).

Additionally, users may have the option to click on the **Assign me a To Do** button on the Current To Do dashboard zone to request their next task. The returned To Do entry automatically becomes the user's **Current To Do**. This does not enable any work list functionality. When finished with

To Do Characteristics

Characteristics are used to associated related information to a To Do entry. Refer to [Linking Additional Information](#) to a To Do Entry for more information.

Several pages in the system allow you to view a To Do entry's characteristic values and / or search for To Do entries by their characteristics.

- The [To Do Management](#) portal includes up to five filters to search for To Do entries by a characteristic value. In addition, the search results display all the current characteristics as name / value pairs.
- The [To Do Entry](#) includes a zone to display any existing characteristics. In addition, if a user is allowed to modify a To Do Entry, the edit dialogue allows a user to add or change characteristics. Note that for manual [To Do types](#), the valid characteristic types are driven by the To Do type. For system To Do types, if characteristic types are configured on the To Do type, only those are available. Otherwise, any valid To Do characteristic type may be linked.
- The [To Do Search](#) allows includes a single characteristic type / value filter.

Related To Do Entries

A user working on a To Do may want to know of other To Dos that may be related to the same problem. If they can be worked by the same user they may get resolved faster. It seems almost impossible to programmatically determine whether two To Dos really stem from the same problem. The user should therefore be presented with a list of potentially related To Dos from which they can select the truly relevant ones.

A To Do entry is considered potentially related to a given To Do if they reference a common entity via a characteristic. For example, if a given To Do references a certain account then any To Do referencing that same account is considered related to it.

NOTE: The common entity is any characteristic value linked to the To Do entry where the characteristic type is a **Foreign Key** and the main table of the foreign key is classified as a **Master** or **Transaction** table.

There are various places where the user is informed as to whether or not the current To Do Entry is related to other To Do Entries.

- To Do entry maintenance displays in insight indicating the number of related To Do Entries for the current To Do. In addition, a Related To Do Entries zone shows information about each related To Do.
- The [Current To Do](#) dashboard zone includes related To Do information and actions.
- The [To Do Management](#) query includes a count of related To Do entries.
- The [To Do Search](#) page supports a **Related To Do filter** that lists all related To Do entries for a given entry.

Complete A To Do

After you finish working on an entry, you can mark it as **Complete**. At this point, you may want to review potentially [related](#) To Do entries and complete those that have been addressed by resolving the problem associated with the current entry.

When you complete a To Do, a [log entry](#) is created that indicates when the To Do was completed and by whom.

There are multiple ways in which you can complete a To Do.

- The [To Do Management](#) page provides many filters in addition to To Do Type. One or more entries may be completed in this page.
- Use the [To Do Entry](#) page to complete a single To Do Entry. The user may also choose to add comments when completing a To Do on this page.
- The [To Do List](#) page allows you to complete one or more selected entries of a specific To Do Type.
- The [To Do Search](#) page allows you to search for To Do entries using various search criteria and complete selected ones.
- You may also use the [Current To Do](#) zone to complete the current entry you are working on and/or all [related](#) entries assigned to you.

NOTE: It is possible for the system to automatically mark an entry as **Complete**. Refer to [The Lifecycle Of A To Do Entry](#) for more information.

Log Entry Events

Any user may add a **User Details** log entry to any To Do at any time (even after it's completed). Refer to [To Do Entry](#) page for more information.

The [To Do Management](#) portal also allows users to add a **User Details** log entry to one or more selected entries.

In addition, there are several log entries created automatically by the system. The following table contains the list of all possible log events.

Event	Log Type	To Do Entry Status	Content of Details	Content of Assigned To (on To Do entry and To Do Log)	Log User ID	Contents of Sent By (on To Do entry)
A To Do is created (by the system or by a user)	Created	Open			ID of user who created the To Do	
A To Do is created (by the system or by a user) in Being Worked On status	Created	Being Worked On		User ID of user who was assigned this To Do	ID of user who created the To Do	Creating user's ID
A To Do is completed (by the system or by a user)	Completed	Complete			ID of user who completed the To Do	
A user takes an open To Do	Assigned	Being Worked On	Selected by user	Assigned To User ID	Assigned To User ID	ID of user who assigned the To Do
A supervisor assigns a To Do to a user (also happens if they reassign)	Assigned	Being Worked On	Assigned to user	Assigned To User ID	ID of user who assigned the To Do	ID of user who assigned the To Do
A user forwards an entry to a user	Forwarded	Being Worked On	<i>Log details entered by user</i>	Forwarded To User ID	ID of user who forwarded the To Do	ID of user who forwarded the To Do
A user forwards an entry to a role	Forwarded	Open	Forwarded to role <i>Role Description (Role ID) + Log details entered by user</i>		ID of user who forwarded the To Do	ID of user who forwarded the To Do
A user sends back a To Do	Sent back	Being Worked On	<i>Log details entered by user</i>	Sent Back To User ID	ID of user who sent back the To Do	ID of user who sent back the To Do
A user adds a log entry	User Details	Status remains the same	<i>Log details entered by user</i>		ID of user who added the entry	
A user manually overrides priority	Override Priority	Status remains the same	Priority changed from <i>old value</i> to <i>new value</i>	Assigned To User ID	ID of user who made the change	
Priority is updated by a calculate priority algorithm	Calculated Priority	Status remains the same	Priority changed from <i>old value</i> to <i>new value</i>		ID of user that added or updated the To Do or the user associated with the Recalculate Priority batch job.	

Managing To Do Entries

This section describes the various pages available to view and work on To Do entries.

To Do Management

The To Do Management portal provides a multi-query zone that includes a search for To Do entries using To Do Type and other To Do information. Users may search for entries using multiple criteria and perform common actions, such as assignment and completion, on selected entries.

It is intended to be used by both supervisors and non-supervisor To Do users. Supervisors are identified by their security access to the portal.

Use **Menu > To Do > To Do Management** to navigate to the portal.

To Do Information Search

This zone allows you to search for non-closed To Do entries using various attributes of the entries, and to perform actions on selected ones.

NOTE: Results are restricted by role. For supervisors, this search returns To Do entries that meet the search criteria and are in a role of which the supervisor is a member. For non supervisors, the search returns open To Do entries that meet the search criteria and are in a role of which the user is a member or entries currently assigned to the user. Refer to [To Do Entries Reference A Role](#) for more information about roles.

The following sections describe the features of this zone.

Search Filters

The following table describes each of the different filters in the To Do Information query zone.

Search Filter	Description
To Do Type	The zone requires the user to enter a To Do Type filter. Only To Do Types where the user is a member of at least one valid role for the type will be available for selection.
To Do Entry Status	Use this filter to restrict the search to To Do entries with a status of Open or Being Worked On . Note that the underlying query excludes completed To Do entries.
Days Old	Use these filters to restrict the search to To Do entries whose age falls within a given range in days. For example, entering values of 2 - 5 will restrict the search to entries that are greater than or equal to two days old and less than or equal to five days old.
Assign To User	Use this filter to restrict the search to To Do entries assigned to a particular user. This filter only appears if the current user has To Do supervisor access.
Priority	Use this filter to restrict the search to To Do entries with a specific priority.
Key Value	Use this filter to restrict the search to To Do entries with a sort key value containing the entered value.
Message	Use this filter to restrict the search to To Do entries with a specific message. This filter appears after the To Do Type filter has been populated and shows a list of the message texts from all non closed To Do entries of the selected To Do Type.
Characteristic	Use these filters to restrict the search to To Do entries with a specific Characteristic Type and value. Up to five characteristic type and

Search Filter**Description**

value pairs may be entered. For manual To Do types, the list of characteristic types will be restricted to the characteristic types defined for the To Do Type. For system To Do Types, if characteristic types are configured for the To Do type, only those are shown; otherwise, all valid characteristic types for the To Do Entry entity are shown.

The system shows the total number of To Do entries displayed and the total number that satisfy your search criteria immediately below the set of results.

NOTE: For performance reasons, the search zone is configured to restrict the number of rows retrieved by the query. The 'Total Records' number will indicate whether there are more To Do entries that meet the criteria than can be displayed.

Search Results


The zone will display all To Do entries that meet the search criteria, up to the limit defined as a parameter to the zone. The following columns are displayed initially:

- The Action column includes the button Work. Click this button to drill into the page on which the entry's respective object is displayed. If the To Do type has been configured to associate a script with this To Do entry's message number, then instead of being taken to the respective page, the associated script is launched. Refer to [Working On A To Do Entry](#) for more information.


NOTE: Current To Do. Drilling into an entry from this page sets it to be the user's current To Do. Refer to [A User's Current To Do](#) for more information.

CAUTION: Automatic Assignment! When you click the Work button for an open entry, the system changes the status of the entry to **Being Worked On** and assigns it to you.

NOTE: Work List. This column has a Work List icon. Clicking this icon populates the [Work List](#) dashboard zone with the results. Clicking on entries in the Work List zone performs all the logic described here when clicking the Work button for one entry. It will drill into the related object or launch a script, if configured. It will make this To Do the 'current' To Do and will assign it to you if it is **Open**.

- Entries are sorted by their **Priority**. Highest in priority are listed first.
- **Days Old** is the number of days since the To Do entry was created by the system. Entries are sorted from youngest to oldest within priority.
- **Status** is the current status of the To Do entry.
- **Assign To User** is the user currently assigned to the To Do entry.
- **Message** is the message associated with the To Do entry.
- **Related** displays a count of entries that are potentially related to the current entry based on having the same foreign key characteristic values. Click the hyperlink to navigate to the [To Do Search](#) page with the current entry as the Related To Do Filter value.
- **Note** displays a Comment icon  if there are either comments or a log entry of type **Forwarded, Sent Back** or **User Details** on the To Do Entry. Click the icon to open a pop-up window which displays the To Do entry information, message, comments and logs.
- **Characteristics** displays the characteristic types and values for the entry as a concatenated string.

- **Related Entity** displays the information for the object related to this entry. The information string allows you to navigate to the page on which the entry's respective object is displayed.
- **To Do ID** is the system generated key of the entry. Click the hyperlink to navigate to the [To Do Entry](#) page to view or maintain the current entry.

Click the **Column Configuration** icon  in the zone header to show the [Column Configuration Area](#) in which additional columns can be added to the results. The additional columns include the following:

- **Create Date/Time** contains the date and time the To Do entry was created by the system.
- **Key 1 to Key 5** contain the first five sort key values for the entry.
- **To Do Role** displays the description of the role for the To Do entry.

Actions

The zone provides the ability to perform certain actions on one or more selected To Do Entries.

Click the check box in the results list heading to select all entries displayed in the grid or click the box in a row to select a specific entry. The following actions are available:

- Use the **Update** button to open a pop-up window in which you can update the priority and/or add a [User Details](#) log entry to the selected To Do entries.
- Use the **Assign** button to assign a user to the selected To Do entries. Supervisors will be presented with a pop-up window in which they can choose the assignee from a list of users who are members of at least one role for the To Do Type. The user search includes information about the number of To Do entries already assigned to each user. For users without supervisor access, the current user is assumed to be the assignee and the window is not displayed. Note that the system does not allow a non supervisor to assign themselves a To Do that is being worked on by another user.
- Use the **Unassign** button to reopen the selected To Do entries and remove the assignee. Note that the system does not allow non supervisors to reopen a To Do that is being worked on by another user.
- Use the **Complete** button to [complete](#) the selected To Do entries. Both supervisors and non supervisors may complete any To Do entry, including those assigned to other users.

To Do Supervisor Functions

This section describes several To Do transactions that aid supervisors in the monitoring and assignment of To Do entries.

Supervisor To Do Summary

This Supervisor To Do Summary page summarizes To Do entries by To Do Type. Only entries that are **Open** (i.e., not assigned to a specific user) or **Being Worked On** are summarized.

NOTE: Only To Do entries that are assigned to a role to which you belong are summarized on this page. Refer to [To Do Entries Reference A Role](#) for more information.

Use **Menu > To Do > Supervisor To Do Summary** to open this page.

Description of Page

This page summarizes all To Do entries that are **Open** or **Being Worked On**.

- **To Do Type** contains the description of the To Do type.

- **Priority** provides a total count of To Do entries of a given type by priority. For each distinct priority the description includes the priority value followed by the corresponding number of entries. If all entries are of the same priority no additional count is displayed.
- **Total** contains the total of **Open** and **Being Worked On** entries. The hyperlink navigates to the [Supervisor To Do Assignment](#) transaction, setting it to display all **Open** and **Being Worked On** entries for the given To Do Type.
- **Open** contains the number of To Do Entries whose status is **Open**. The hyperlink navigates to the [Supervisor To Do Assignment](#) transaction, setting it to display all **Open** entries for the given To Do Type.
- **Being Worked On** contains the number of To Do entries whose status is **Being Worked On**. The hyperlink navigates to the [Supervisor To Do Assignment](#) transaction, setting it to display all **Being Worked On** entries for the given To Do Type.

The "age bars" that appear to the right of the totals highlight the age of the To Do entries. The colors red, yellow and green are used to highlight the relative number of old, middle-aged and new To Do entries. The age associated with each of these colors is [user-configurable](#).

Supervisor User Summary

The Supervisor User Summary page summarizes To Do entries by the user who is assigned to work on the entries. Only entries that are **Being Worked On** are summarized.

NOTE: Only To Do entries that are assigned to a role to which you belong are summarized on this page. Refer to [To Do Entries Reference A Role](#) for more information.

Use **Menu > To Do > Supervisor User Summary** to open this page.

Description of Page

This page contains a grid that displays all assigned To Do entries by User ID and To Do type. It indicates how many To Do entries of the displayed Type are assigned to the user (Assigned this Type) and how many To Do entries in total are assigned to the user (Assigned Total).

Clicking the hyperlinks transfer you to the [Supervisor To Do Assignment](#) transaction. On this transaction, you can assign / reassign To Do entries to a specific user. The difference is that clicking the To Do Type hyperlink causes only **Open** entries to be displayed on the Assignment transaction. Clicking the count in the Assigned this Type column causes **Being Worked On** entries assigned to the displayed user to be shown.

Supervisor To Do Assignment

This page lists To Do entries of a given type and allows a supervisor to assign / reassign them to a designated user.

NOTE: Refer to the [To Do Management](#) portal for a page that provides many more filters to narrow down the results. In addition, it provides more data in the results and allows the user to adjust the priority of To Do entries.

Use **Menu > To Do > Supervisor To Do Assignment** to open this page.

Description of Page

This page contains **Open** or **Being Worked On** To Do entries of a given type. These entries can be assigned to a specified user one-by-one or several at a time. The following points describe this functionality:

- You can select a different **To Do Type** by typing in a value or pressing the search button in the upper right corner. An **Info** description of the To Do type appears on the left.
- You can change the list of entries displayed according to their status. Select the **Filter by** status to be All (blank), **Open**, or **Being Worked On**.

- For **Being Worked On** entries, you may also **Filter by User** to limit the search to a specific user assigned to the To Do entries.
- If you want to re-sort the entries by a different value, click the search button adjacent to **Extra Information**. Each row's respective sort value is displayed in the **Extra Information** column.

NOTE: Extra Information. Each To Do list's default **Extra Information** is defined on the list's [To Do Type](#). Also note that each To Do list's available **Extra Information** differs.

- Click the **Select All** button to select all entries displayed in the grid. Click the **Clear All** to remove the selections in the grid.
- Use the checkbox if you want to assign specific entries.
- Entries are further sorted by their **Priority**. Highest in priority are listed first.
- **Create Date/Time** contains the date and time the To Do entry was created by the system. Click the Create Date/Time hyperlink to navigate to the To Do page where additional information is visible and where additional actions are possible.
- Click the number of **Related** entries hyperlink to navigate to the [To Do Search](#) query to list potentially related entries to the current entry.
- **Worked By** shows the user currently assigned to the To Do entry.
- **To Do Entry Status** is the current status of the To Do entry.
- **Message** contains a description of the To Do entry. Use the message description hyperlink to drill into the page on which the entry's respective object is displayed. If the To Do type has been configured to associate a script with this To Do entry's message number, then instead of being taken to the respective page, the associated script is launched. Refer to [Working On A To Do Entry](#) for more information.
- You can assign entries to a user:
 - Enter a user ID or search for a user using the **Assign To User** field. You can only select a user that is associated with a role that is associated with this **To Do Type**. Note that the current number of assignments is displayed adjacent.
 - Select one or more entries to assign to this user and click the **Assign** button. The entries' status becomes **Being Worked On**.

To Do List Functions

This section describes To Do pages that aid users in the assignment and working of entries in To Do lists.

To Do Summary

This page summarizes To Do entries that meet the following qualifications:

- The entry is **Open** and you belong to the entry's To Do role. Refer to [To Do Entries Reference A Role](#) for more information about roles.
- The entry is **Being Worked On** and you are the user to which it is assigned.

Use **Menu > To Do > To Do Summary** to open this page.

Keyboard Shortcut	Alternate Shortcut
Alt+X	Alt+Shift+X

NOTE: Refer to [Shortcut Key Summary](#) for information about the alternate shortcut.

FASTPATH: Refer to [The Big Picture of To Do Lists](#) for an overview of To Do functionality.

Description of Page

This page is dedicated to a grid that summarizes the To Do entries on which you can work. The following information is displayed:

- **To Do Type** contains the description of the To Do type.
- **Priority** provides a total count of To Do entries of a given type by priority. For each distinct priority the description includes the priority value followed by the corresponding number of entries. If all entries are of the same priority no additional count is displayed.
- **Total** contains the total of **Open** and **Assigned to You** entries. The hyperlink navigates to the [To Do List](#) transaction, setting it to display all **Open** and **Being Worked On** entries assigned to you for the given To Do Type.
- **Open** contains the number of To Do Entries whose status is **Open**. The hyperlink navigates to the [To Do List](#) transaction, setting it to display all **Open** entries for the given To Do Type.
- **Assigned to You** contains the number of To Do entries that are assigned to you whose status is **Being Worked On**. The hyperlink navigates to the [Supervisor To Do Assignment](#) transaction, setting it to display all **Being Worked On** entries for the given To Do Type.

The "age bars" that appear to the right of the totals highlight the age of the To Do entries. The colors red, yellow and green are used to highlight the relative number of old, middle-aged and new To Do entries. The age associated with each of these colors is [user-configurable](#).

To Do List

This page lists To Do entries of a given type that are either **Open** (e.g., not assigned to a specific user) or are assigned to you.

NOTE: Refer to the [To Do Management](#) portal for a page that provides many more filters to narrow down the results. In addition, it provides more data in the results and allows the user to adjust the priority of To Do entries.

NOTE: Restricted by role. Only those entries assigned to a role to which you belong may be seen on this page. Refer to [To Do Entries Reference A Role](#) for more information.

FASTPATH: We recommend familiarizing yourself with the topics described in [The Big Picture of To Do Lists](#) to fully appreciate how To Do entries are managed by the system.

Use **Menu > To Do > To Do List** to open this page.

Description of Page

This page contains a grid that summarizes the To Do entries of a given type on which you can work.

The **Info** section contains a concatenation of the **To Do Type's** description and the user's name.

The area beneath **Info** provides you with options that control which To Do entries appear and the order in which they are displayed. The following points describe the various options:



- Use **Sort By** to define how you want the To Do entries sorted. The following options are available:
 - **Assigned Date - On or After.** This option sorts the To Do entries in chronological order (based on the date the entry was assigned to you) starting from the input **Date**. If you choose this option, the system assumes you want to start from the earliest To Do entry. If you want to start from a date other than the earliest To Do entry, enter the desired **Date** and click the adjacent search button.

- **Assigned Date - On or Before.** This option sorts the To Do entries in reverse chronological order (based on the date the entry was assigned to you) starting from the input **Date**. If you choose this option, the system assumes you want to start from the today's date. If you want to start from a date other than today's date, enter the desired **Date** and click the adjacent search button.
- **Create Date - On or After.** This option sorts the To Do entries in chronological order (based on the date the entry was created) starting from the input **Date**. If you choose this option, the system assumes you want to start from the earliest To Do entry. If you want to start from a date other than the earliest To Do entry, enter the desired **Date** and click the adjacent search button.
- **Create Date - On or Before.** This option sorts the To Do entries in reverse chronological order (based on the date the entry was created) starting from the input **Date**. If you choose this option, the system assumes you want to start from the today's date. If you want to start from a date other than today's date, enter the desired **Date** and click the adjacent search button.
- **Extra Information.** This option sorts the To Do entries by the value of the **Extra Info**.
- Use **Extra Info** to choose data related to the To Do Entry that should be used to sort the To Do entries. This information is displayed in each entry's **Extra Information** column.

NOTE: Extra Info. Each To Do types's available **Extra Info** is defined on the [To Do Type Sort Keys](#). Also note that each To Do type's available **Extra Info** differs based on the configuration of the To Do Type.

- Use **Filter by** to restrict the To Do entries based on their status. The following options are available:
 - **All.** This option shows all entries that are **Open** or **Being Worked On**.
 - **Being Worked On.** This option shows all entries that are **Being Worked On** (and assigned to you).
 - **Open.** This option shows all entries that are **Open**.

The grid that follows contains the To Do entries that match your criteria. The following information appears:

- Click the **Select All** button to select all entries displayed in the grid. Click the **Clear All** button to remove the selections in the grid.
- Use the checkbox if you want to mark specific entries as **Completed** or **Being Worked On** using the **Complete** or **Assign** buttons at the bottom.
- A Comment icon  appears if there are comments on the To Do item and a Log Entry icon  appears if there is a log entry of type **Forwarded**, **Sent Back** or **User Details** on the To Do item. This column appears if at least one To Do has either comments or a special log entry.
- Entries are further sorted by their **Priority**. Highest in priority are listed first.
- **Create Date/Time** contains the date and time the To Do entry was created by the system.
- Click the Create Date/Time hyperlink or either of the special icons to navigate to the [To Do entry](#) page.
- Click the number of **Related** entries hyperlink to navigate to the [To Do Search](#) query to list potentially related entries to the current entry.
- **Assigned Date/Time** contains the date and time the To Do entry was assigned to a user. This field is only populated on entries that are **Being Worked On**. If a user assigned the To Do entry, the user ID and name of the user who assigned the To Do entry are displayed beneath the Date/Time.
- **Message** contains a description of the To Do entry. Use the message description hyperlink to drill into the page on which the entry's respective object is displayed. If the To Do type has been configured to associate a script with this To Do entry's message number, then instead of being taken to the respective page, the associated script is launched. Refer to [Working On A To Do Entry](#) for more information.

NOTE: Current To Do. Drilling into an entry from this page sets it to be the user's current To Do. Refer to [A User's Current To Do](#) for more information on how to efficiently scroll through additional entries in your current work list.

CAUTION: Automatic Assignment! When you drill into an entry using the message description hyperlink from this page, the system changes the status of the entry to **Being Worked On** and assigns it to you.

- **To Do Entry Status** is the current status of the To Do entry.
- **Extra Information** contains the data for the entry that corresponds to the Extra Info chosen in the sort options above.
- **To Do ID** contains the unique identifier of the To Do entry.
- If you want to mark one or more entries as **Being Worked On**, check the desired entries and then click the **Assign** button.
- If you want to mark one or more entries as **Completed**, check the desired entries and then click the **Complete** button. Completed entries do not appear on the To Do list (but they are retained on the database for audit purposes).

To Do Entry

Refer to [Understanding To Do Processing](#) for detailed explanations about many To Do entry features.

To view a To Do entry. Use **Menu > To Do > To Do Entry > Search** . You are brought to a query portal with options for searching for a To Do Entry. Note that some query options limit the results. Refer to the inline help for more information.

NOTE: Restricted by role. Any user may change Comments for any To Do Entry. In addition, any user may add user Log details. However, for all other changes and actions, you may only change To Do entries that are assigned to you or ones that are **Open** and assigned to a role to which you belong. Refer to [To Do Entries Reference A Role](#) for more information about roles.

Once a To Do Entry has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The inline help provides information about several fields. The following points highlight information about some of the fields that may be visible.

If the To Do entry has non-complete [related entries](#) an insight appears indicating the number of related entries. In addition, details about the related entries are shown in the Related To Do Entries zone.

If the To Do entry was **Created By** a background process, information about that background process appears with a link allowing the user to drill into the batch run, if desired. Refer to [To Dos Created for Object-Specific Error Conditions](#) for more information.

If the To Do type is configured to [To Do Entries May Be Routed Out Of The System](#), the **Extract Info** appears. If the extract process is pending, the information indicates that. Otherwise, if it has run, information about the background process appears with a link allowing the user to drill into the batch run, if desired.

The **Characteristics** collection shows existing characteristics. This collection is maintained using the Edit action for users that are allowed to update this To Do entry. Refer to [To Do Entry Characteristics](#) for more information.

Actions

This page has two special actions.

- Use the **Forward** to assign a To Do to another user or role. When a To Do is forwarded, a log entry indicates who forwarded the To Do, to whom it was forwarded, and the specified reason. The dialogue prompts for whether to send to a User or a Role. The search for User includes information about the number of To Do entries already assigned to this user. When choosing a User, confirm the correct Role. The Details provided are captured in the log entry.

- **Send Back** is only applicable if the **Sent By** user is populated. When a To Do is sent back, a log entry is created that indicates when the To Do was sent back and by whom. If the To Do is **Open**, this dialogue allows you to send it back to a different role. If the To Do is assigned to you, the dialogue allows you to re-assign the To Do to the person who assigned it to you. The Details provided are captured in the log entry.

Related To Do Entries

If the To Do has [related To Do entries](#), this zone appears with information about the related entries. Use this **Edit** hyperlink to navigate to the [To Do Search](#) query to perform actions on the related entries.

Adding a Manual To Do Entry

If your system is configured with [manual To Do types](#), you can add a To Do entry online.

Open the **Menu > To Do > To Do Entry > Add**.

You are prompted for a **To Do Type**. Only To Do types that are set up for manual usage appear in the list.

The default **To Do Type Priority** is displayed adjacent. Use **Override Priority** to override the default [priority](#) assigned by the system.

Type a **Subject** for your To Do entry. Depending on how the To Do type is set up, the subject may appear as the message in the To Do entry and To Do lists.

Enter a **Comment** to provide a longer description of and more details about the To Do entry.

Send To indicates whether the new To Do will be assigned to a **Role** (the default) or **User**.

The **Role** indicates the role to which this To Do entry will be assigned. Only users of the specified role will be able to view and work on the To Do entry. The role is defaulted from the To Do type. If you want to assign the To Do to a different role, change the specified role. You can only assign the To Do entry to one of the roles specified on the To Do type.

User appears if **Send To** is set to **User**. Use this option if you want to assign the To Do to a specific user. The user must be a member of the selected role.

If the [To Do type](#) defines valid characteristic types, the **Characteristics** collection is visible. If any of the characteristic types are flagged as Default on the To Do type, the characteristic type with its default value (if applicable) are displayed.

To Do Search

This page allows you to search for To Do entries using various search criteria, and perform various actions on selected ones. This page includes completed To Do entries in its search results, when applicable with respect to the filter criteria.

Use **Menu > To Do > To Do Search** to open this page.

NOTE: Refer to the [To Do Management](#) portal for a page that provides many more filters to narrow down the results. In addition, it provides more data in the results and allows the user to adjust the priority of To Do entries.

NOTE: No restriction by role. This page returns all To Do entries that meet the search criteria, regardless of the To Do entry's role. Using the search, a user may drill into the details of a To Do entry even if the To Do is not assigned to the user. Refer to [To Do Entries Reference A Role](#) for more information about roles.

The top half of the page is where you enter the criteria used to search for To Do entries.

CAUTION: Try to be as specific as possible when entering search criteria. Why? Because entering open-ended search criteria may have a severe impact on response times.



The following table describes each of the different search filters.

Search Filter	Description
Related To Do	<p>If you're searching for potentially related entries for a given To Do entry, choose Specific To Do for the Related To Do Filter. Enter the To Do ID.</p> <p>A filter value of Not Applicable defaults.</p>
To Do Type	<p>If you're searching for a To Do entry and you know the value of a To Do Type linked to the To Do entry, choose Specific To Do Type for the To Do Type Filter. Enter the To Do Type.</p> <p>A filter value of All defaults.</p>
Char Type	<p>If you're searching for a To Do entry and you know the value of a characteristic linked to the To Do entry, choose Specific Char Type for the Char Type Filter. Enter the Characteristic Type and Characteristic Value.</p> <p>A filter value of Not Applicable defaults.</p> <p>Only characteristic types that include a characteristic entity of To Do Entry and that have been marked to allow searching by characteristic value are available.</p>
User	<p>Use this search method if you know the Created By User, Assigned To User or Completed By User for the To Do entry. Enter the appropriate User Id.</p> <p>A filter value of Not Applicable defaults.</p>
Date Range	<p>Use this field if you're searching for a To Do entry and know the Date Created or Date Completed.</p> <p>A filter value of Date Created defaults. The Start Date/Time defaults to the current date, 12:00AM and the End Date/Time defaults to the current date, 11:59PM.</p> <p>If you enter a Date Range Filter of All, the system ignores this search method.</p>
Status	<p>Use this search method to restrict the search to To Do entries with a specific status value. You may search for To Do entries in the status of Open, Being Worked On or Completed. You may also perform a combination search for To Do entries that have a status value of either Open or Being Worked On.</p> <p>A filter value of All defaults.</p>

NOTE: The system allows any combination of values. The system does not try to validate your search combinations, even if they do not make sense. For example, if you enter a **User Filter** value of **Completed By User** and a **Status Filter** value of **Open**, you will not get an error. In this case, the system will find no records that match your criteria (because To Do entries that are Open do not have a Completed By User).

The system shows the total number of To Do entries that satisfy your search results immediately below the grid.

The first group of To Do entries is displayed in the grid at the bottom of the page.

- Click the **Select All** button to select all entries displayed in the grid. Click the **Clear All** to remove the selections in the grid.
- Use the checkbox if you want to perform actions (see below) on specific entries. Only **Open** or **Being Worked On** entries may be selected.
- A Comment icon  appears if there are comments on the To Do item and a Log Entry icon  appears if there is a log entry of type **Forwarded**, **Sent Back** or **User Details** on the To Do item. This column appears if at least one To Do item has either comments or a special log entry. Use either icon to navigate to the To Do entry page.
- Entries are sorted by their **Priority**. Highest in priority are listed first.
- **Create Date/Time** contains the date and time the To Do entry was created by the system. Use the hyperlink to navigate to the To Do entry page.
- Click the number of **Related** entries hyperlink to redisplay this page listing potentially related entries to the current entry.

- **Last Assigned Date/Time** contains the date and time the To Do entry was most recently assigned to a user, if applicable, along with the Assigned To user.
- **Message** contains a description of the To Do entry. If the To Do entry is in the status of **Open** or **Being Worked On**, the message description allows you to drill into the page on which the entry's respective object is displayed. If the To Do type has been configured to associate a script with this To Do entry's message number, then instead of being taken to the respective page, the associated script is launched. Refer to [Working On A To Do Entry](#) for more information.

NOTE: Current To Do. Drilling into the page by clicking the Message hyperlink does not automatically assign **Open** To Do entries to the user (as the To Do List page does when clicking on the Message). But it does treat this as the user's current to do, enabling the functionality in the Current To Do dashboard zone. Refer to [A User's Current To Do](#) for more information.

- **Status** contains the current status of the To Do entry.
- **Completion Date/Time** contains the date and time the To Do entry was completed and the user who completed the entry, if applicable.
- **To Do Type** contains a description of the To Do type for the To Do entry.

Actions

- Use the **Assign** button to assign or reassign the selected entries to a user. Enter a user ID or search for a user using the **Assign To User** field. By default this field is populated with the current user's ID. Note that the user's current number of assignments is displayed adjacent. Upon successful processing, the selected entries are assigned to the specified user and the status of **Open** entries becomes **Being Worked On**. Additionally, a corresponding [log entry](#) is added to each To Do.

NOTE: Only users with **Supervisor Assignment** access rights may reassign entries that are not currently assigned to them.

- Use the **Reopen** button to unassign the selected entries from their users. Upon successful processing, the selected entries become **Open** and a corresponding [log entry](#) is added to each To Do.

NOTE: Only users with **Supervisor Assignment** access rights may unassign entries that are not currently assigned to them.

- Use the **Complete** button to [complete](#) the selected entries.

To Do Dashboard

This portal provides a high level summary of open and being worked on To Do entries by various criteria. It is intended to be used by both supervisors and non-supervisor To Do users. An implementation may introduce additional zones to this portal as needed.

NOTE: Results are restricted by role. For supervisors, the charts consider To Do entries that are in a role of which the supervisor is a member. For non supervisors, the charts consider open To Do entries that are in a role of which the user is a member or entries currently assigned to the user. Refer to [To Do Entries Reference A Role](#) for more information about roles.

Use **Menu > To Do > To Do Dashboard** to navigate to the portal.

The following zones may appear as part of the portal's **Main** tab page

- **To Do Entries By Role.** This zone summarizes To Do entries by their currently assigned role.
- **To Do Entries By Status.** This zone summarizes To Do entries by their current status.

- **To Do Entries By Priority.** This zone summarizes To Do entries by their priority.
- **To Do Entries By Age.** This zone summarizes To Do entries by age based on age ranges defined for the current user.
- **To Do Entries By Type.** This zone summarizes To Do entries by their to do type.

Selecting a specific slice of data on any of these zones broadcasts its value to the other zones on the portal allowing them to filter their data accordingly. Use the **Reset** action on each zone to reset the filter set by this zone.

Chapter 18

Reports

The following section describes the functionality provided for submitting reports on-line and for viewing reports submitted through a third party reporting tool.

FASTPATH: For information about this functionality, see [The Big Picture of Reports](#).

On-line Report Submission

The on-line report submission page enables you to request a specific report to be run. This is only available if your system is able to integrate with your third party reporting system on-line. The system comes configured to integrate with BI Publisher.

Report Submission - Main

This page allows you to submit a report on-line. Navigate to this page using **Menu > Tools > Report Submission..**

Description of Page

To submit a report, choose the **Report Code** for the process you wish to submit. Refer to [Defining Report Definitions](#) for information about defining reports in the system.

A description of the report is displayed.

The parameters defined for the report on the [report definition](#) page are displayed in the grid. To see a description of a parameter, use the search button adjacent to the parameter label.

NOTE: Default note. The default values displayed for the parameters are defined on the [report definition](#) page.

When you have entered the appropriate parameter values, click the **Submit** button to submit the report. Refer to [How To Request Reports](#) for more information.

NOTE: Reports submitted via the on-line report submission page are not logged in the report history repository and are therefore not visible from the [report history](#) page. Only reports submitted through your third party reporting tool's batch scheduler are logged in the report history, if supported by the tool.

Viewing Historic Reports

If your third party reporting tool stores historic copies of reports in a repository, you may configure the system to allow a user to view these historic reports on-line.

NOTE: This page currently does not display historical reports for BI Publisher.

Report History - Main

This page allows you to view a report stored in your third party reporting tool's report repository, if supported. Navigate to this page using **Menu > Tools > Report History**.

Description of Page

The top part of the page shows the **Report Code**, **Description** and **Long Description** for the selected report.

To select a report, search by either **Report Code** or **Description** and click the adjacent search button. The reports matching your search criteria are displayed. Choose the desired report from the selection list.

NOTE: Displaying Report History. When you choose a report, the system needs to interface with your reporting software to display a list of historical occurrences of this report.

**Oracle Utilities
Work and Asset Management and
Operational Device Management**

Administrative User Guide

Release 2.4.0.0.0

F52875-01

April 2022

Also used with:

- Oracle Utilities Operational Device Cloud Service 22A

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

Framework Administrative User Guide.....	17
General Options.....	17
Defining Installation Options.....	17
Installation Options - Main.....	17
Installation Options - Messages.....	18
Installation Options - Algorithms.....	18
Installation Options - Accessible Modules.....	20
Installation Options - Installed Products.....	20
Support For Different Languages.....	21
User Language.....	21
Customer Language.....	22
Defining Languages.....	22
Defining Countries.....	23
Country - Main.....	23
Country - States.....	24
Defining Currency Codes.....	24
Defining Time Zones.....	24
Designing Time Zones.....	24
Setting Up Time Zones.....	26
Setting Up Seasonal Time Shift.....	27
Defining Geographic Types.....	27
Defining Work Calendar.....	27
Defining Display Profiles.....	28
Additional Hijri Date Configuration.....	30
Defining Phone Types.....	30
Defining Feature Configurations.....	30
Feature Configuration - Main.....	30
Feature Configuration - Messages.....	31
Defining Master Configurations.....	31
Security & User Options.....	32
The Big Picture of Application Security.....	33
Application Security.....	33
Action Level Security.....	35
Field Level Security.....	35
Encryption and Masking.....	36
System Encryption.....	36
User Interface Masking.....	36
Application Encryption.....	40
The Base Package Controls One User, One User Group, And Many Application Services.....	42
Importing Security Configuration from an External Source.....	43
The Big Picture of Row Security.....	43
Defining Application Services.....	44
Defining Security Types.....	44
Security Type - Main.....	45
Defining User Groups.....	45
User Group - Main.....	45
User Group - Application Services.....	46
User Group - Users.....	46
User Group Services Management.....	47
Defining Access Groups.....	47
Defining Data Access Roles.....	48
Data Access Role - Main.....	48
Data Access Role - Access Group.....	48
Defining Users.....	48
Data Privacy.....	49
The Approach to Implementing Object Erasure.....	49
Configuring a Maintenance Object for Erasure.....	49
Manage Erasure Schedule Algorithm.....	50

Monitoring the Schedule and Performing Erasure.....	50
Erasing User Information By Obfuscation.....	51
Viewing an Object's Erasure Status.....	51
Viewing Erasure Configuration.....	51
Archiving the Object Erasure Schedule.....	51
Cryptography Keys.....	51
Understanding Key Rings.....	52
Signature Keys.....	52
Defining Key Rings.....	52
User Interface Tools.....	52
Understanding Portals and Zones.....	53
Setting Up Portals and Zones.....	56
Defining Zone Types.....	56
Defining Zones.....	57
Overriding Parameter Values.....	58
Overriding Zone Parameters.....	58
Zone Parameter Details.....	58
Data Explorer Graph Configuration.....	78
Pagination Configuration.....	85
Use Data Explorer for Derived Data.....	86
Configuring Timeline Zones.....	87
Defining Context-Sensitive Zones.....	88
Defining Portals.....	88
Defining Menu Options.....	89
Menu - Main.....	89
Menu - Menu Items.....	90
Defining Navigation Keys.....	91
Navigation Key Types.....	91
Navigation Key vs. Navigation Option.....	92
The Flexibility of Navigation Keys.....	92
Linking to External Locations.....	92
Overriding Navigation Keys.....	93
Maintaining Navigation Keys.....	93
Defining Navigation Options.....	94
Navigation Option - Main.....	95
Navigation Option - Tree.....	96
Understanding System Messages.....	97
Defining System Messages.....	97
Defining Message Categories.....	97
Defining Messages.....	98
Understanding Unified Search.....	98
Trees.....	100
Defining Trees.....	102
Contextual Insights.....	103
Insight Types.....	103
Defining Insight Types.....	106
Insight Groups.....	106
Defining Insight Groups.....	106
Color Configuration.....	107
Defining Colors.....	107
Color Options.....	107
Defining Display Icons.....	107
Custom Look and Feel Options.....	108
User Interface.....	108
UI Map Help.....	108
Database Options.....	108
Understanding The Data Model.....	108
Fields.....	109
Tables.....	111
Maintenance Objects.....	114
Defining Valid Values.....	115
Characteristic Types.....	116
Foreign Key References.....	118
Defining Tables.....	120

Defining Fields.....	121
Defining Maintenance Objects.....	121
Maintenance Object - Main.....	121
Maintenance Object - Options.....	122
Maintenance Object - Algorithms.....	122
Maintenance Object - Maintenance Object Tree.....	124
Defining Lookups.....	124
Lookup - Main.....	125
Defining Extendable Lookups.....	126
Extendable Lookup Advanced Topics.....	126
Defining Characteristic Types.....	127
Defining Foreign Key References.....	128
Audit Trails.....	129
Captured Information.....	129
How Auditing Works.....	129
The Audit Trail File.....	130
How To Enable Auditing.....	130
Turn On Auditing For a Table.....	130
Specify The Fields and Actions To Be Audited.....	131
Audit Queries.....	131
Audit Query by User.....	131
Audit Query by Table / Field / Key.....	132
Bundling.....	133
About Bundling.....	133
Sequencing of Objects in a Bundle.....	133
Recursive Key References.....	134
Owner Flags on Bundled Entities.....	134
Configuring Maintenance Objects for Bundling.....	134
Working with Bundles.....	136
Creating Export Bundles.....	136
Creating and Applying Import Bundles.....	137
Editing Export Bundles.....	137
Editing Import Bundles.....	138
Revision Control.....	138
About Revision Control.....	138
Turning On Revision Control.....	138
Configuring Maintenance Objects for Revision Control.....	139
Working with the Revision Control Zones.....	139
Checking Out an Object.....	140
Checking In an Object.....	140
Reverting Changes.....	140
Forcing a Check In or Restore.....	140
Deleting an Object.....	141
Restoring an Object.....	141
Working with the Revision Control Portal.....	141
Revision Control Search.....	141
Information Lifecycle Management	142
The Approach to Implementing Information Lifecycle Management.....	142
Batch Processes.....	143
Eligibility Algorithm.....	145
Enabling ILM for Supported Maintenance Objects.....	145
Ongoing ILM Tasks.....	146
Archived Foreign Keys.....	146
Configuration Tools.....	147
Business Objects.....	147
The Big Picture of Business Objects.....	147
What Is A Business Object?.....	147
Business Object Inheritance.....	153
Each Business Object Can Have A Different Lifecycle.....	154
BO Algorithm Execution Summary.....	161
Granting Access To Business Objects.....	162
Defining Business Objects.....	162
Business Object - Main.....	162
Business Object - Schema.....	163

Business Object - Algorithms.....	164
Business Object - Lifecycle.....	166
Business Object - Summary.....	168
Advanced BO Tips and Techniques.....	168
Creating a Physical Business Object.....	169
Managing To Do Entries.....	169
Submitting a Batch Job.....	170
Defining Status Reasons.....	170
Data Areas.....	170
Defining Data Areas.....	171
Data Area - Main.....	171
Data Area - Schema.....	171
Business Services.....	172
Service Program.....	172
Defining Business Services.....	173
Business Service - Main.....	173
Business Service - Schema.....	173
Useful Services and Business Services.....	174
User Interface (UI) Maps.....	177
Defining UI Maps.....	179
UI Map - Main.....	179
UI Map - Schema.....	180
UI Map Attributes and Functions.....	180
UI Map Standards.....	237
Ensuring Unique Element IDs for UI Maps.....	248
Process Flows.....	249
Understanding Process Flows.....	249
Designing Process Flows.....	251
Defining Process Flow Types.....	253
Action Method.....	253
Defining Action Methods.....	255
Algorithms.....	255
The Big Picture Of Algorithms.....	255
How To Add A New Algorithm.....	257
Minimizing The Impact Of Future Upgrades.....	258
Defining Algorithm Types.....	258
Defining Algorithms.....	259
Advanced Algorithm Tips.....	260
Script Options.....	260
The Big Picture Of Scripts.....	260
Scripts Are Business Process-Oriented.....	260
A Script Is Composed Of Steps.....	261
A Script May Declare Data Areas.....	261
Securing Script Execution.....	261
The Big Picture Of BPA Scripts.....	261
How To Invoke Scripts.....	262
Developing and Debugging Your BPA Scripts.....	262
Launching A Script From A Menu.....	263
Launching A Script When Starting The System.....	263
Executing A Script When A To Do Entry Is Selected.....	263
The Big Picture Of Script Eligibility Rules.....	264
The Big Picture Of Server-Based Scripts.....	267
Additional Coding Options For Server Scripts.....	268
Plug-In Scripts.....	269
Service Scripts.....	271
Groovy Library Scripts.....	271
Debugging Server-Based Scripts.....	272
Maintaining Scripts.....	272
Script - Main.....	272
Script - Step.....	273
Script - Data Area.....	315
Script - Schema.....	315
Script - Eligibility.....	316
Merging Scripts.....	317

Script Merge.....	318
Maintaining Functions.....	320
Function - Main.....	321
Function - Send Fields.....	321
Function - Receive Fields.....	322
Maintaining Managed Content.....	322
Defining Entity Tags.....	323
Advanced Schema Topics.....	323
Schema Nodes and Attributes.....	323
UI Hint Syntax.....	344
Schema Designer.....	352
Schema Viewer.....	354
Business Event Log.....	354
Miscellaneous Topics.....	355
Module Configuration.....	355
Menu Item Suppression.....	355
Menu Suppression.....	355
Turn Off A Function Module.....	356
Global Context Overview.....	356
System Data Naming Convention.....	356
Base Product System Data.....	356
Implementation System Data.....	357
Accessibility Considerations.....	357
Referencing URIs.....	357
Validation Against an Allowlist.....	358
URI Substitution.....	358
External File Storage.....	359
Caching Overview.....	360
Expression Parser.....	361
Debug Mode.....	364
Prepare Issue Details.....	365
System Override Date.....	365
Additional Search Options.....	365
To Do Lists.....	366
The Big Picture of To Do Lists.....	366
To Do Entries Reference A To Do Type.....	366
To Do Entries Reference A Role.....	366
The Lifecycle Of A To Do Entry.....	367
How Are To Do Entries Created?.....	368
Suppress the Creation of the To Do Entry.....	368
To Do Entries Created By Background Processes.....	369
To Do Entries Created By Algorithms.....	370
To Do Entries Created Manually.....	371
Refining a To Do Entry's Details at Creation Time.....	371
The Priority Of A To Do Entry.....	373
Working On A To Do Entry.....	374
Monitoring A To Do Entry.....	374
To Do Entries Have Logs.....	374
Implementing Additional To Do Entry Business Rules.....	375
To Do Entries May Be Routed Out Of The System.....	375
To Do Information May Be Formatted By An Algorithm.....	375
Periodically Purging To Do Entries.....	375
Setting Up To Do Options.....	376
Installation Options.....	376
Messages.....	376
Feature Configuration.....	376
Defining To Do Roles.....	377
To Do Role - Main.....	377
To Do Role - To Do Types.....	377
Defining To Do Types.....	378
To Do Type - Main.....	378
To Do Type - Roles.....	378
To Do Type - Sort Keys.....	379
To Do Type - Drill Keys.....	379

To Do Type - Message Overrides.....	380
To Do Type - To Do Characteristics.....	380
To Do Type - Algorithms.....	380
Implementing The To Do Entries.....	381
Background Processes.....	382
Understanding Background Processes.....	382
Background Processing Overview.....	382
Parallel Background Processes.....	383
Optimal Thread Count.....	384
Parameters Supplied To Background Processes.....	385
Indicating a File Path or File Name.....	387
Manifest File.....	387
Sharing Customer Exported Data.....	388
Processing Errors.....	389
Error Post-Processing Logic.....	389
Post-Processing Logic.....	389
Timed Batch Processes.....	390
Monitor Background Processes.....	390
Plug-in Driven Background Processes.....	392
Processing System Records.....	392
Uploading Records.....	397
How to Re-extract Information.....	398
How to Submit Batch Jobs.....	399
How to Track Batch Jobs.....	399
How to Restart Failed Jobs and Processes.....	399
Assessing Level of Service.....	399
Defining Batch Controls.....	399
Batch Control - Algorithms.....	402
Defining File Integration Types.....	403
On-Line Batch Submission.....	403
Batch Submission Creates a Batch Run.....	403
Jobs Submitted in the Background.....	404
Email Notification.....	404
Running Multi-Threaded Processes.....	404
Batch Jobs May End in Error.....	405
Submitting Jobs in the Future.....	405
Lifecycle of a Batch Job Submission.....	405
Granting Access To Batch Submission.....	406
Batch Job Submission - Main.....	406
Tracking Batch Processes.....	408
Batch Run Tree.....	408
Batch Run Tree - Main.....	408
Batch Run Tree - Run Control.....	409
Batch Day Tracking.....	409
Batch Day Dashboard.....	410
Batch Run Threads.....	411
The Big Picture of Requests.....	412
Request Type Defines Parameters.....	412
Previewing and Submitting a Request	412
To Do Summary Email.....	412
Defining a New Request.....	413
Setting Up Request Types.....	413
Maintaining Requests.....	413
Attachments.....	414
Attachment Overview.....	414
Configuring Your System for Attachments.....	415
Maintaining Attachments.....	416
Adding Attachments.....	416
Application Viewer.....	417
Application Viewer Toolbar.....	417
Data Dictionary Button.....	417
Physical and Logical Buttons.....	417
Collapse Button.....	417
Attributes and Schema Button.....	418

Maintenance Object Button.....	418
Algorithm Button.....	418
Batch Control Button.....	418
To Do Type Button.....	418
Description and Code Buttons.....	419
Service XML Button.....	419
Select Service Button.....	419
Java Docs Button.....	419
Groovy Java Docs Button.....	420
Classic Button.....	420
Preferences Button.....	420
Help Button.....	420
About Button.....	420
Slider Icon.....	420
Data Dictionary.....	421
Using the Data Dictionary List Panel.....	421
Primary And Foreign Keys.....	421
Field Descriptions Shown.....	422
Using the Data Dictionary Detail Panel.....	422
Related Tables View.....	422
Table Detail View.....	422
Column Detail View.....	423
Maintenance Object Viewer.....	423
Using the Maintenance Object List Panel.....	423
Using the Maintenance Object Detail Panel.....	423
Algorithm Viewer.....	424
Using the Algorithm Viewer List Panel.....	424
Using the Algorithm Plug-In Spot Detail Panel.....	424
Using the Algorithm Type Detail Panel.....	424
Using the Algorithm Detail Panel.....	424
Batch Control Viewer.....	424
Using the Batch Control Viewer List Panel.....	425
Using the Batch Control Detail Panel.....	425
To Do Type Viewer.....	425
Using the To Do Type Viewer List Panel.....	425
Using the To Do Type Detail Panel.....	425
Service XML Viewer.....	426
Using the Service XML Viewer Overview Panel.....	426
Using the Service XML Viewer Detail Panel.....	426
Java Docs Viewer.....	426
Using the Java Docs Viewer List Panel.....	427
Using the Java Package Detail Panel.....	427
Using the Java Interface / Class Detail Panel.....	427
Groovy Java Docs Viewer.....	427
Application Viewer Preferences.....	427
Application Viewer Stand-Alone Operation.....	428
Stand-Alone Configuration Options.....	428
Example Application Viewer Configuration.....	428
Application Viewer Generation.....	429
Monitoring and Analytics.....	429
Reporting Tool Integration.....	430
The Big Picture Of Reports.....	430
Integration with BI Publisher.....	430
How To Request Reports.....	431
Viewing Reports.....	431
Configuring The System To Enable Reports.....	432
Configuring BI Publisher Reports.....	432
Defining Reporting Options.....	432
Defining Report Definitions.....	433
Sample Reports Supplied with the Product.....	434
How to Use a Sample Report Provided with the System.....	434
How To Define A New Report.....	434
Use a Sample Report as a Starting Point.....	435
Publishing Reports in BI Publisher.....	435

Designing Your Report Definition.....	436
Service Health Check.....	438
Health Check Configuration.....	439
Viewing Health Check Results.....	439
Introducing Health Check Conditions.....	440
Measuring Performance.....	440
Understanding Performance Targets.....	440
Performance Target Objects Overview.....	441
Calculating and Displaying Performance Targets.....	441
Performance Target Metrics and Metric Types.....	441
Performance Target Categories and Types.....	442
Performance Targets Define Specific Metrics.....	442
Objects Linked to a Performance Target.....	442
Creating Performance Target Zones.....	443
Setting Up Performance Target Configuration.....	443
Performance Target Category Lookup.....	443
Defining Performance Target Types.....	443
Maintaining Performance Targets.....	444
Capturing Statistics.....	444
Understanding Statistics.....	444
Configuring Your System for Statistics.....	445
Defining and Monitoring Statistics.....	445
Creating Cube Views.....	445
Understanding Cube Viewer.....	446
Cube Viewer Components.....	446
Cube Configuration Components.....	450
Configuring Cube Types.....	450
Maintaining Cube Types.....	451
Cube Type Advanced Topics.....	451
Product Use Metrics.....	452
External Messages.....	452
Incoming Messages.....	452
Inbound Web Services.....	452
Understanding Inbound Web Services.....	452
Configuring Inbound Web Service Options.....	458
Deploying Inbound SOAP Web Services.....	467
Guaranteed Delivery.....	468
Outgoing Messages.....	469
Outbound Messages.....	469
Polling Outbound Messages Using OSB.....	470
Batch Message Processing.....	470
Real Time Messages.....	471
Designing the System for Outbound Messages.....	471
Configuring the System for Outbound Messages.....	475
Managing Outbound Messages.....	485
Web Service Adapters.....	486
Understanding Web Service Adapters.....	486
Setting Up Web Service Adapters.....	487
Sending Email.....	488
Web Service Category.....	488
Defining Web Service Categories.....	489
JMS Message Browser.....	489
Oracle Integration Cloud Catalog.....	489
Web Service Catalog Configuration.....	490
Web Service Catalog Master Configuration.....	490
Maintaining the Web Service Catalog.....	490
XAI Documentation Note.....	491
Integrations.....	491
Creating Application Links in External Applications.....	491
LDAP Integration.....	492
LDAP Integration Overview.....	493
Configuring LDAP Integration.....	494
LDAP Mapping.....	495
Oracle Identity Manager Integration.....	497

User Redirection.....	499
Batch Scheduler Integration.....	500
Data Synchronization.....	500
Understanding Data Synchronization.....	500
Accumulative Data Synchronization.....	501
Generic Data Synchronization.....	502
Maintaining Sync Requests.....	504
Operational Analytics.....	504
Understanding Operational Analytics.....	504
Calendar and Time Dimensions.....	505
Maintaining Bucket Configurations.....	506
Defining Analytics Options.....	507
Defining Analytics Tables.....	507
Defining Analytics Dimensions.....	507
Defining Analytics Facts.....	507
Defining Characteristic Mapping.....	508
Analytics Integration.....	508
Understanding Analytics Integration.....	508
Business Flags.....	509
Understanding Business Flags.....	509
Standard Name.....	510
Business Flag Type Defines Behavior for a Standard Name.....	510
Business Flag Type Algorithms.....	510
Objects Linked to a Business Flag.....	510
Impacted Business Process.....	511
Dates.....	511
Creating Business Flags.....	511
Confidence.....	512
Setting Up Business Flag Configuration.....	513
Standard Name Category Characteristic Type.....	513
Business Flag Standard Name Lookup.....	513
Business Process Lookup.....	513
Integration Configuration.....	513
Defining Business Flag Types.....	514
Maintaining Business Flags.....	514
Market Transaction Management.....	514
Understanding Market Transaction Management.....	514
Configuring Market Transaction Management.....	515
Defining Market Configurations.....	515
Defining Market Message Types.....	515
Defining Market Process Types.....	515
Maintaining Market Transactions.....	516
Maintaining Market Processes.....	516
Maintaining Market Process Events.....	516
Maintaining Market Messages.....	516
Content Migration Assistant (CMA).....	517
Understanding CMA.....	517
Types of Migrations.....	518
Migrating Configuration Data.....	519
Migrating Business Data.....	520
Data with System Generated Primary Keys.....	522
Importing Data that References a User	523
Additional Considerations.....	523
CMA Configuration.....	524
Master Configuration - Migration Assistant.....	524
Migration Plans.....	525
Defining a Migration Plan.....	525
Understanding the BO Filtering Process.....	526
Migration Plans for Objects with XML-Embedded Links.....	527
Defining a Migration Request.....	528
Identifying Tables to Exclude From Migrations.....	529
Configuring Custom Objects for Migration.....	529
The CMA Execution Process.....	530
Exporting a Migration.....	531

Migration Data Set Export.....	532
Export Lifecycle.....	532
Importing and Applying a Migration.....	533
Import Step.....	533
Compare Step.....	535
Approval Step.....	537
Apply Step.....	538
Adjusting Data Prior to Comparing.....	542
Import Process Summary	543
Cancelling a Data Set.....	547
Additional Note Regarding Imports.....	548
Caching Considerations.....	548
Maintaining Import Data.....	548
Running Batch Jobs.....	550
CMA Reference.....	551
Framework-Provided Migration Configuration.....	551
Facts.....	553
Fact Is A Generic Entity.....	553
Fact's Business Object Controls Everything.....	553
Fact Supports A Log.....	554
Conversion.....	554
Understanding The Conversion Process.....	554
Conversion Entities.....	556
Conversion Steps.....	556
Load Legacy Data Into Staging Tables.....	556
Validate Information In The Staging Tables.....	557
Allocate Production Keys.....	558
XML Resolution.....	560
Insert Production Data.....	561
Validate Production.....	562
A Note About Keys.....	562
Multiple Owners In A Single Database.....	562
Conversion Entity Dashboard.....	564
Data Export.....	564
Understanding Data Export.....	564
Generalized Data Export.....	564
Specialized Data Export.....	569
Defining Data Export Control.....	570
Generalized Export Dashboard.....	570
Work and Asset Management Administrative User Guide.....	571
General Options.....	571
Defining Installation Options.....	571
Formatted Address.....	571
Address Geocoding.....	571
Access Control.....	572
Three Decimals Places for Currency.....	575
Defining Colors.....	576
Documents.....	576
Understanding Documents.....	576
Configuring Document Options.....	577
Defining Document Types.....	577
Defining Documents.....	577
Defining Standard Notes.....	577
Organizations.....	578
Understanding Organizations.....	578
Understanding Contact Types.....	578
Configuring Organization Options.....	578
Defining Organization Types.....	578
Defining Organizations.....	579
Configuring Contact Options.....	579
Defining Contact Types.....	579
Compliance.....	579
Understanding Compliance.....	579
Preventive.....	580

Corrective.....	580
Configuring Compliance Options.....	580
Defining Compliance Categories.....	580
Defining Compliance Types.....	580
Service History.....	581
Understanding Service History Types.....	581
Failure Profiles.....	581
Questionnaires.....	582
Follow Up Work.....	582
Understanding Questionnaires.....	583
Configuring Service History Options.....	583
Defining Questions.....	583
Defining Failure Causes.....	583
Defining Failure Modes.....	583
Defining Failure Components.....	584
Defining Failure Repairs.....	584
Defining Failure Profiles.....	584
Defining Service History Types.....	584
Assets.....	585
Understanding Asset Location Types.....	585
Location Hierarchy.....	585
Understanding Asset Types.....	586
Valid Service History Types.....	586
Valid Attributes.....	586
Valid Measurement Types.....	587
Condition Assessment Class.....	588
Understanding Functional Process Types.....	588
Defining Functional Process Types.....	588
Understanding Configuration Types.....	588
Configuring Asset Options.....	589
Defining Measurement Identifiers.....	589
Defining Measurement Types.....	589
Defining Asset Location Types.....	589
Defining Out of Service Location Types.....	590
Defining Out of Service Locations.....	590
Defining Condition Assessment Groups.....	590
Defining Condition Assessment Class.....	590
Defining Condition Assessments.....	591
Defining Asset Types.....	591
Configuring Activity Generation Configurations.....	591
Defining Configuration Types.....	591
Defining Configurations.....	592
Defining Configuration Reports.....	592
Operational Device Management.....	592
Understanding Operational Device Management.....	592
Asset and Component Configuration Types.....	593
Configuring Operational Device Management Options.....	593
Defining Asset Configuration Types.....	593
Inventory.....	593
Understanding Inventory.....	593
Storerooms.....	594
Vehicles as Storerooms.....	594
Shipment Deficiency.....	594
Configuring Inventory Options.....	594
Defining Storeroom Types.....	595
Defining Storerooms.....	595
Defining Shipment Deficiency Types.....	595
Purchasing.....	595
Understanding Purchasing.....	596
Buyers.....	596
Commodities.....	596
Vendors.....	596
Capability Types.....	597
Freight on Board (FOB).....	597

Payment Terms.....	597
Understanding Invoices.....	597
Configuring Purchasing Options.....	599
Defining Blanket Contract Access Lists.....	599
Defining Buyers.....	599
Defining Commodity Codes.....	600
Defining Purchase Commodities.....	600
Defining Capability Type.....	600
Defining Free on Board.....	600
Defining Payment Terms.....	601
Defining Exchange Rates.....	601
Work Options.....	602
Understanding Activity Types.....	602
Default Labor Requirements.....	602
Checklist Types.....	602
Planned Service History Types.....	603
Work Categories.....	603
Work Classes.....	603
Service Class.....	603
Completion Event Types.....	604
Understanding Resource Requirements.....	605
Crafts.....	605
Equipment.....	605
Miscellaneous Resources.....	605
Materials.....	605
Understanding Resource Charges.....	605
Labor Earning Types.....	606
Time Periods.....	606
Timekeepers.....	606
Understanding Projects.....	606
Understanding Construction Work.....	607
Compatible Units.....	607
Understanding Permit Templates.....	608
Understanding Completion Events.....	609
Processing an Activity Completion Message.....	609
Understanding Service Call.....	609
Service Codes.....	609
Service Categories.....	610
Configuring Resource Related Options.....	610
Defining Unit of Measure Resources.....	610
Defining Crafts.....	610
Defining Equipment.....	610
Defining Miscellaneous Resource.....	611
Configuring Resource Charges Options.....	611
Defining Labor Earning Types.....	611
Defining Overtime Types.....	611
Defining Timekeepers.....	612
Configuring Work Options.....	612
Total Risk Lookups.....	612
Defining Project Categories.....	612
Defining Checklist Types.....	613
Defining Permit Classes.....	613
Defining Permit Templates.....	613
Defining Service Classes.....	613
Defining Work Categories.....	614
Defining Work Classes.....	614
Defining Completion Event Types.....	614
Defining Completion Events.....	614
Defining Service Codes.....	615
Defining Service Categories.....	615
Defining Activity Types.....	615
Configuring Construction Work Options.....	616
Defining Property Units.....	616
Defining Compatible Unit Categories.....	616

Defining Compatible Units.....	616
Defining Compatible Unit Sets.....	617
Defining Compatible Unit Usages.....	617
Defining Compatible Unit Contractor Bids.....	617
Defining Equipment Groups.....	618
Defining Overheads.....	618
Preventive Maintenance.....	618
Understanding Preventive Maintenance.....	618
Preventive Maintenance Calendars.....	620
Configuring Preventive Maintenance Options.....	620
Defining Maintenance Managers.....	621
Defining Preventive Maintenance Calendar.....	621
Scheduling.....	621
Understanding Scheduling.....	621
Employees.....	621
Crews.....	623
Planners.....	623
Service Area Criteria.....	623
Service Class Criteria.....	623
Understanding Shift Plan Templates.....	624
A Shift Plan Template is Made of Shift Templates.....	624
Crew Specific Plan Template.....	625
Common Plan Templates.....	625
One Effective Plan Template for a Crew.....	626
Shift Generation.....	626
Overridden Shift Details.....	627
Draft and Original Templates.....	627
Configuring Scheduling Options.....	628
Defining Employee Types.....	628
Defining Employees.....	628
Defining Employee Unavailability Types.....	629
Defining Crew Shift Types.....	629
Defining Crew Types.....	629
Defining Crews.....	629
Defining Planners.....	630
Defining Service Areas.....	630
Defining Shift Plan Templates.....	630
Defining Shift Templates.....	630
Approvals.....	631
Understanding Approval Processing.....	631
Approval Hierarchy.....	631
No Approval Needed.....	632
Example.....	633
Defining Approval Profiles.....	634
Understanding Change Request Types.....	634
Entity Specific Versus General Change Requests.....	634
Initiating Change Requests.....	634
Enabling Change Request For Another Entity.....	635
Defining Change Request Types.....	636
Financials.....	636
Understanding Financials.....	636
Cost Category.....	636
Cost Centers.....	637
Business Unit.....	637
Rate Calculation.....	637
ERP Integration.....	639
Understanding Cost Adjustment Types.....	639
Configuring Financial Options.....	639
Defining Accounting Calendars.....	639
Defining Business Units.....	640
Defining Cost Adjustment Types.....	640
Defining Cost Categories.....	640
Defining Cost Centers.....	640
Defining Distribution Codes.....	641

Defining Expense Codes.....	641
Defining Factors.....	641
Defining Tax Rate Schedules.....	641
Integrations.....	642
Understanding Integrations.....	642
Understanding Web Services.....	642
Work Order API.....	642
Defining External Systems.....	643
Understanding GIS Integration.....	643
GIS Owns Non-Tracked Assets.....	643
Which System Owns Asset Related Details?.....	644
A Feature's Global Identifier Is Key To Synchronization.....	644
Asset Synchronization from GIS is Version Based.....	644
Asset Synchronization Mode.....	645
Asset Removal.....	645
Asset Abandonment.....	645
Reference The Original Asset.....	646
GIS Map Viewer.....	646
Configuring GIS Integration.....	646
Asset Details Mapping.....	646
Map Viewer Setup.....	647
Configuring Outbound Messages.....	647
Master Configuration Setup.....	648
Asset Synchronization Batch Processes.....	648
Understanding Data Synchronization.....	650
Configuring Inbound Communication Options.....	651
Defining Communication Types.....	652
Maintaining Inbound Communications.....	652
Enterprise Resource Planning Integration.....	652
Creating Application Links in External Applications.....	652
Configuration Options.....	653
Content Migration Assistant (CMA).....	653
Data that Cannot be Migrated Using CMA.....	653
Global Configuration Setup.....	654
Analytics Configuration.....	654
Contextual Insights Configuration.....	654
Home Page Configuration.....	656
Understanding Home Page Configuration.....	656
Defining Statistics.....	657
Understanding Online / Batch Loaders.....	657
Creating CSV Files.....	657
Initiating Requests.....	658
Table Setup Sequence.....	659
OUAF Admin Setup.....	659
General Data Setup.....	660
Approval Setup.....	661
Asset Management Setup.....	661
Construction Work Management Setup.....	662
Financial Setup.....	663
Inventory Setup.....	663
Purchasing Setup.....	664
Work Management Setup.....	664
Batch Controls.....	665

Chapter 1

Framework Administrative User Guide

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

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 **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**.
- 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.

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, zones, 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.

CAUTION: In most implementations, the time zone defined here matches the database time zone. However, if there is some reason that the database time zone does not match the installation time zone, an implementation may configure a setting in the properties file to automatically convert data from the database time zone to the time zone defined here when displaying dates. Note that when this property setting is defined, changes to the installation time zone will require the server and the thread pool workers to be restarted in order for the changes to take effect.

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 sample reports 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).
Domain Name	The text entered for this message type is meant to identify the environment and can be used for logic where it is useful to indicate the environment for a user. For example, the email sent when a background process finishes includes this information, if populated.

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**.

System Event	Optional / Required	Description
Validate Email Attachment	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 Sending Email for more information
Address Geocoding	Optional	Algorithms of this type use Oracle Locator to retrieve latitude and longitude coordinates using address information.
Global Context	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 Global Context Overview for more information.
Guaranteed Delivery	Optional	Algorithms of this type may be called by processes that receive incoming messages that should 'guarantee delivery'. Refer to Guaranteed Delivery for more information. The business service F1-GuaranteedDelivery may be used to invoke this plug-in spot.
Ldap Import	Optional	Algorithms of this type are called for operations on users, groups, and group memberships after they have been processed.
Ldap Import Preprocess	Optional	Algorithms of this type are called to preprocess data retrieved from LDAP.
Next To Do Assignment	Optional	This type of algorithm is used to find the next To Do entry a user should work on. It is called from the Current To Do dashboard zone when the user ask for the next assignment.
Reporting Tool	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 report submission or to review report history online. This algorithm is used by the two on-line reporting pages to properly invoke the reporting tool from within the system.
SMS Receive	Optional	This type of algorithm is used to provide SMS receive service. Only one algorithm of this type should be plugged in.
SMS Send	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

System Event	Optional / Required	Description
		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.
To Do Information	Optional	We use the term To Do information to describe the basic information that appears throughout the system to describe a To Do entry . Plug an algorithm into this spot to override the system default "To Do information".
To Do Pre-creation	Optional	These types of algorithms are called when a To Do entry is being added. Refer to Refining a To Do Entry's Details at Creation Time for more information.

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.

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

NOTE: If you have developed implementation-specific transactions, you can populate the release ID, release ID suffix, build number and patch number for the **Customer Release** entry to define the latest version of implementation-specific logic that has been applied to this environment. The framework provides a script (**F1CustRelM**) to allow these fields to be updated manually. In addition, the **F1-InstallationOptions** inbound web service provides operations to view and update those fields via a REST call. Alternatively, your implementation team may populate these fields as part of their upgrade scripts.

The **Product Status** is used to indicate if a product installed by default in the version you are logged into is **Active** or **Inactive**.

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](#) window for your product.

Support For Different Languages

User Language

The language English (with the locale en-US) is provided by the system and all system metadata are delivered with an English description or label. The system provides support for defining other languages and supports 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.

Supporting a New Language

Normally, setting up the system for another language (besides the delivered English language entry) 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.

- 1. Define a language code and indicate that it is enabled.** For details on this procedure, see [Defining Languages](#).
- 2. Install or confirm the existence of a language pack.** For an on-premise installation, if the product supplies a language pack with translations for the system metadata descriptions, follow the instructions for downloading the language pack and installing it into your environment. For a cloud installation, if the product supplies a language pack for a given language, no additional step is needed here. The language pack is already installed.
- 3. Copy descriptions of all language-enabled tables from an existing translation (e.g., English).** The copied values act 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, **F1-LANG**. The batch copies all English labels in the system.

The batch process will also update the new language rows with the translated metadata descriptions from the language pack, if installed.

- 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.

NOTE: The language pack updates all language entries for base owned system data. If your implementation updates base owned labels and descriptions after applying a language pack, they will be overwritten the next time an updated language pack is applied. Note that most user facing labels and messages support defining an Override Label or

Override Description. This information is not updated by the base product and should be utilized if your implementation desires a specific label or description.

Managing Multiple Languages

Once a non-English language has been set up, there are several points to note when adding or updating records that include translatable strings.

- When a new record is added through the user interface or through a REST service call or any other means that goes through the standard maintenance service, the system automatically generates entries for each supported language and copies the text of the entered description to all the generated languages.
- When the translatable string for a record is updated, only the user's language row is updated. The system does not ripple the change to other supported language entries (because it could be overriding a valid translation).
- When your system is upgraded to a new version, the new version typically includes new system data. The upgrade only includes the English language rows. Implementers should run the **F1-LANG** batch program to generate language rows for the your supported languages for this new data. If a new translation pack for your supported language is available at the time of upgrade, then running this process also applies the translated strings. (As mentioned above, for on-premise installations, you need to download and install the language pack before this step). If the language pack is not available at the time of upgrade, then the batch program should be run again when the language pack has been installed.
- New translatable objects may also be included in a patch / hot fix. In this situation, the **F1-LANG** batch program should be run after installing the patch in order to ensure there are language rows for your supported languages. Updated language packs are only provided for full release upgrades and not for hot fixes. So implementations may need to temporarily supply translations for the new strings until you update to the next release with the next updated translation pack.

Customer Language

Your specific product may also support capturing the language of a customer. Such that correspondence sent from the product may be produced in a language set on a customer record. Refer to your specific product's documentation for more information about additional language support.

Defining Languages

Your product may support multiple languages. For example, the field labels, input text, and even outputs and reports can be configured to appear in a localized language. A language code for every potential language exists in the system to supply this information in various languages.

Select **Admin > General > Language** to define a language.

Description of Page

Enter a unique **Language**. 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.

NOTE: For on premise applications, the login page displays all the languages that are enabled, allowing the user to toggle the login instructions in that language. 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:

Locale	Comments
en_US	American English
en_AU	Australian English
pt_BR	Brazilian Portuguese
fr_FR_EURO	European French
ja_JP	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 <https://docs.oracle.com/javase/7/docs/api/java/text/Collator.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.

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**.

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 indicating that an address field is **Optional**, you make that field available everywhere addresses for this country are used in the system. You can enter your own descriptions for the labels. These labels will appear wherever addresses are maintained in the system.

NOTE: Your specific product may also add the ability to mark an address field as **Required**. If that is available, then the product is also supplying appropriate validation in all places where a user defines an address.

NOTE: For any country where the **State** is enabled, 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.

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 > General > 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.

Defining Time Zones

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

Designing 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.

The following sections describe concepts and configuration topics related to managing 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.

The Base Time Zone

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.

NOTE: An attribute in the system properties file may be configured to indicate that the DB session time zone should be synchronized with the value defined on the Installation Options. Refer to the *Server Administration Guide* 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.

Standard vs. Legal Time

The term **Legal Time** refers to the actual clock time as may be impacted by daylight saving (also called 'local time'). This time line has a gap hour on entry to the daylight saving period and a duplicate hour on exit from it.

The term **Standard Time** refers to a time line without any daylight saving shifts applied to it. This virtual time line has no gap hour on entry to the daylight saving period nor a duplicate hour on exit, i.e. it is continuous and non-ambiguous. As a general rule, it is recommended that all time sensitive data is stored in the standard 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 time zone or daylight saving shifting of data that may be stored in different time zones.

Depending on your specific product, entities may have their date/time information stored in either of the following options:

- Standard time of the base time zone (also called 'physical time').
- Standard time of another time zone related to the entity's geographical location (also called 'logical time').
- Legal time of the base time zone.

The metadata definition of a [table](#) specifies whether any of its date/time fields are stored in standard time or not. While typically all date/time fields of a table that supports standard time would be stored in the same way the system allows each [field](#) to specify its own option. The metadata definition of such field specifies whether it is base time zone standard time (**Physical Standard Time**), standard time of some other time zone associated with the entity (**Legal Standard Time**) or stored in base time zone legal time (**Referenced Time Zone**).

Storage vs. Display

Regardless of the time zone option used for storing the data in standard time, all date/time information is entered and displayed on the user interface in the legal time of its respective time zone. If a field is defined to be stored in the standard time of the base or other time zone, the user interface is designed to accept the data in the legal time of the respective time zone and shift it to the standard time of the same time zone before saving the data and performs the reverse shift from standard time back to legal time before displaying the data to the user.

It is important to understand that time zone and daylight saving time conversions to and from standard time are performed behind the scenes as part of the user interface layer. Once data reaches the server it is already in its storage time option. The user should not be aware of these time conversions.

Time shifting between standard and legal times does not happen automatically. When dealing with fields stored in standard time, their data entry and display need to explicitly include time shifting features as part of their user interface and zone configuration.

Date / Time Schema Elements

When defining date / time fields in a business object 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).

By default data stored in standard time is displayed in the corresponding time zone's legal time. Additional schema attributes can be used to indicate if the display of the time should be shifted to the legal time of a different time zone. For example, 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 Nodes and Attributes- Standard Time Considerations](#) for more information.

Exchanging Date / Time Information With External Systems

Date/time information should be exchanged between systems in standard XSD format as it includes a reference to the time zone of the specified time as an offset from Coordinated Universal Time (UTC). The system automatically converts date/time elements in an inbound message from XSD to internal format and vice versa for an outbound message. The latter is controlled by an explicit date/time format setting on the External System record for the outbound message type.

The offset is determined as followed and it is based on the time zone associated with the element as explicitly defined by the respective inbound or outbound schema:

- Elements defined as stored in the standard time of the base or other time zone would always have the same offset throughout the year as this time line is never shifted for daylight saving. The offset would be the standard offset of that time zone from UTC, i.e. not during daylight saving time.
- Elements stored in the legal time of the base time zone would have the standard offset of the base time zone when the date/time value is outside the daylight saving period and the shifted offset when it is inside that period of time.

User Time Zone

If your company does business in multiple time zones, the user record of each user may reference the time zone of their location. This time zone provides additional information about the location of the user and as such may be used by specific business rules that involve time zone settings.

NOTE: The time zone defined on the user record is not used to automatically display date/time information in the user's time zone. By default, the system displays date/time fields that are stored in standard time in the legal time of their respective time zone and not the user's time zone.

Setting Up Time Zones

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

Open this page using **Admin > General > Time Zone > Search** .

This is a standard [All-in-One](#) portal.

Refer to the embedded help text for more information on the Time Zone attributes.

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**.

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: Product specific. There is no framework functionality that uses this information. Refer to your specific product documentation to verify how this table is used in your specific product. In addition, use the data dictionary link below to determine if this object is a foreign key on any tables specific to your product.

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.

Defining Work Calendar

This portal is used to view and maintain information about a work calendar. Work calendars are used to ensure system-calculated dates fall on a workday.

You can access the portal from **Admin > General > Work Calendar**. You are brought to a query portal with options for searching for a specific calendar. Once a calendar has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones will appear as part of the portal's **Main** tab page

- **Work Calendar.** This zone provides information about the selected calendar.
- **Work Calendar Holidays.** This zone lists the holidays observed by your organization for this calendar. Use the zone actions to add, maintain or duplicate a work holiday entry.

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** and **Description** to identify the profile.

Enter a **Date Format**. This affects how users view dates and how entered dates are parsed. The following table highlights standard supported date mnemonics and what is displayed at runtime.

Mnemonic	Comments
dd	Day of the month.
d	Day of the month, suppressing the leading 0.
MM	Month number.
M	Month number, suppressing the leading 0.
yyyy	The 4-digit year.
yy	The 2-digit year.
y	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.

Examples:

Configuration Format	Sample Output
MM-dd-yyyy	04-09-2001
d/M/yyyy	9/4/2001
yy.MM.dd	01.04.09
MM-dd-y	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 [master configuration](#) 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. The letter 'H' is used for dates that fall within the *Heisei* era. The letter 'R' is used for dates that fall within the *Reiwa* 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**. The following table highlights standard supported date mnemonics.

Mnemonic	Comments
hh	The hour 1-12.
h	The hour 1-12, suppressing the leading 0.
HH	The hour 0-23.
H	The hour 0-23, suppressing the leading 0.
KK	The hour 0-11.
K	The hour 0-11, suppressing the leading 0.
kk	The hour 1-24.
k	The hour 1-24, suppressing the leading 0.
mm	Minutes.
m	Minutes, suppressing the leading 0.
ss	Seconds.
s	Seconds, suppressing the leading 0.
a	Indicates to include am or pm . This is only needed for 12 hour formats, not 24 hour formats. (hh, h, KK, K). If an am or pm is not entered, it defaults to am .

Examples:

Configuration Format	Sample Output
hh:mma	09:34PM (can be entered as 09:34p)
hh:mm:ss	21:34:00
h:m:s	9:34:0

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 as follows:

- A comma (","). Large numbers group by threes separated by a comma, for example 1,000,000.
- A 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.

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 the [Hijri to Gregorian Date Mappingmaster configuration](#) 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.

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.

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 > General > Feature Configuration**.

Use the **Feature Configuration Query** portal to search for an existing feature configuration. Once a feature configuration is selected, you are brought to the maintenance portal to view and maintain the selected record.

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 external messages, you'd select **External Messages**.

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. Use the index to look for 'feature configuration' to find the various types of feature options.

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 > General > Feature Configuration** 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.

Overview / Initial Setup

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 content 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. To find help topics related to functionality controlled by the master configuration records, use the keyword 'master configuration' in the index.

Implementations may configure the business objects to limit the ones that are visible to users if desired.

- If the master configuration does not apply to your implementation, navigate to the business object for this master configuration and update the Instance Control to **Do not allow new instances**. This ensures that the business object will not be visible to any user for the implementation.
- If a given master configuration record is only relevant for certain users in the system, application security may be used to limit the records visible by a given user. By default the base delivered business objects typically reference the Master Configuration MO application service. This may be overridden to link custom application services to the master configuration BOs to provide more granular security. Users will only see the master configurations that they have security for.

Configuration

To set up a master configuration, open **Admin > General > 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:

- If a master configuration record exists for a given master configuration business object, the broadcast icon may be used to view details information about the adjacent master configuration. In addition, an edit icon is visible to allow a user to update the record.
- If a master configuration record does not exist for a given master configuration business object, the add icon is visible to allow a user to define the record.

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.

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 transactions or explicit services using an [application service](#). The following points highlight what may be secured.

- The following points highlight security related to viewing and modifying individual records 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. Note that the application service for the maintenance object is defined on its related [service program](#).
 - 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 (stand-alone) [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. Note that the application service for the portal is defined on its related [service program](#), which is derived via its navigation option and navigation key.
 - [Menu items](#) may define an application service / access mode. Typically the security supplied for portals and maintenance objects provides enough granularity to suppress menu items that a user does not have access to. Linking an explicit application service / access mode will further suppress the menu item under one of 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 menu option, link the object's application service and the Add access mode to the Add menu item.

NOTE: The base product does not typically provide menu items with application services configured. Implementations may add this configuration if one of the above scenarios exist.

- [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 is delivered with 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.
- For portal based pages, individual elements may be shown or hidden based on security using the **oraHasSecurity** function in a UI map's HTML or in UI Hints in a schema. Refer to [Check User's Security Access](#) for more information.
- **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. In addition, implementations may override the configured application service if desired.
- **Batch controls** define an application service which provides the ability to secure submission of individual batch processes. The application service must include an access mode of **Execute**. The base product batch controls are supplied with appropriate application services. These services will typically have an ID that matches the batch control ID.
- **Report Definition** records define an application service. The application service must include an access mode of **Submit / View Report**.
- The following objects are securable but are typically executed via internal processes. The security is provided to ensure that any access to the objects from an external source is secured.
 - **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 that are linked to an inbound web service are configured with special application service. All other business services and service scripts are delivered with a default application service, which may be overridden by an implementation.
 - Scripts may also check security within an Edit Data step type if there is some functionality that should or should not execute based on a user's security access. The Edit Data step would include a call to the business service **F1-CheckApplicationSecurity**
 - **Service Programs** define an application service. As mentioned above, for Portals and Maintenance Objects, their application service is taken from the related service program. In base, specific application services are released for each of these types of service programs. All other service programs are typically delivered with a default application service, which may be overridden by an implementation. Note that for service programs linked to a Business Service, the application service on the business service takes precedence when invoking the business service.

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.

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). The recommendation is to use a business object validation algorithm for checking field level security.

NOTE:

The check for field level security must exist in validation logic to ensure that data is not added or updated if it does not pass the field level security rules.

It is not straight forward to implement user interface changes related to field level security, such as protecting elements under certain conditions. This is because overriding the base product delivered user interface to check for implementation specific conditions is not always supported, depending on how the user interface is built, or requires a full duplication of product artifacts, which is not recommended.

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 the database using an encryption key. There are two different types of encryption described in the sections below. System encryption refers to columns in the system identified by the product to use encryption. Application encryption refers to the ability for an implementation to configure fields and elements that should be encrypted in the database.

"Masking" refers to overwriting all or part of an un-encrypted field value with a masking character. For example, perhaps only the last 4 digits of a tax ID number are visible with the other digits changed to an asterisk. The system provides support for masking fields on the user interface that may be stored as plain text in the database. In addition, there are cases where encrypted fields are shown to the user interface using masked values rather than the encrypted value.

The following sections provide more information about each feature.

System Encryption

The system automatically encrypts certain fields captured in various option tables or context tables. This is mainly used for passwords. For example, passwords captured in Message Sender context or password

In addition, batch control supports configuring a security option for parameters that capture sensitive information, such as a password. Refer to [Defining Batch Controls](#) for more information.

It is also possible to enable system encryption using the characteristic type **F1-PWD**. However, the maintenance object must include specific code to enable system encryption for characteristics of this type. In Oracle Utilities Application Framework, the only maintenance object that supports this is extendable lookup. Refer to [Extendable Lookup Advanced Topics](#) for more information.

User Interface Masking

The functionality described in this section is used to take data that is stored in plain text in the database and mask the value before it is presented to a user (or an external system). This feature includes the ability to allow some users to view the data unmasked using security configuration. 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.

The following topics describe how to mask field values.

Identify the Data to be Masked

Identify the data that is stored as plain text, but should be masked for display to users. For example, imagine that you have identified that Credit Card Numbers and a person's federal ID number (for example, in the United States, the Social Security Number or SSN). Each field identified may be displayed and maintained in different user interfaces throughout the system, but the masking rules for a given field are probably uniform regardless of where the data is displayed.

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.

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

For each field, if there are some users that may see the data unmasked on one or more of the user interfaces, then security configuration is required. If the value of a field should be masked for all users across all pages in the application, then the security configuration is not needed.

Security Configuration

Define a [security type](#) for each field with two authorization levels:

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

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.

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.

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.

Configure a Masking Algorithm

A data masking algorithm (using the algorithm entity value of **Feature Configuration - Data Masking**) must be created for each combination of masking rules and 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. If certain users may see the data unmasked, parameters capture the application service, security type and authorization level defined above used to evaluate this. In addition, parameters allow you to configure how much of the data to mask, what masking character to use. Refer to the algorithm type description for more information.

Determine How the Fields are Displayed

The masking configuration differs based on how a field is retrieved for access to the user interface. So for the masking of one "logical" field (like a person's SSN), there may be multiple configuration entries required to cover all the access methods. Review each user interface where a given field is displayed and create the following categories:

- The field is an element that is retrieved by invoking a business object, a business service, or a service script
- The field is displayed on a fixed maintenance page (and is therefore retrieved by invoking a page service)
- The field is displayed on a fixed search page (and is therefore retrieved by invoking a search service)
- The field is stored as an ad hoc characteristic

Create a Feature Configuration for Each Masked Element

Create a feature configuration with a Feature Type of **Data Masking**. An option entry with option type of **Field Masking** is needed for every combination of field to mask and the method used to display the data. The value will contain mnemonics

that reference the appropriate data masking algorithm along with configuration that differs depending on how the field is retrieved for display as described below.

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 and XML column in the table as a single element, the element in the 'where' clause must map to the same XML column 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 on a 'fixed' search page, it is retrieved 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 **Inspect**. Search for "serviceName". The service name is listed there. To find the field name to mask, search for "Widget Info". There should be two results found, one for the filter area and one for the results area. This results area has the text "SEARCH_RESULTS" as prefix for each field. The field names are the ones after the x\$. Do not reference the x\$ when defining the field name. Note, the "where" statement can only apply to fields that are also part of the search results.

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.

Application Encryption

The functionality described in this section allows implementations to configure fields to encrypt when storing it in the database. This functionality is mutually exclusive from the User Interface Masking functionality described in the previous section. This feature supports encrypting specific elements stored within a CLOB or XML column.

The following points highlight the features of the encryption functionality:

- The encryption key is defined using a keystore, which must be set up in order to use this functionality. For details about setting up the keystore in the system, see the Installation Guide.
- When a field is configured to be encrypted, the encrypted data is stored in a special encryption field that is not the source field (the one exposed to the user on the user interface). The source field captures the data as masked. Because a special field is required to support encryption, the product must provide support for that field to be encrypted.
- For encrypted data that must allow searching, the system supports capturing a hash value in a special field. The product must provide support for this functionality. Besides providing a special field to capture the hash value, base search functionality for that data must also cater for this configuration.
- The system supports encrypting data that is captured as an element within an XML field. If the XML field is provided in a schema owned by the product, then the product must provide specific support for the capture of the encrypted data.

The following sections provide additional information about the support for encryption provided by the framework. Refer to the security chapter of the administration guide for your particular product for more information.

Encrypting and Masking the Data

When a product enables encrypting for a given type of data, a special encryption field should be created to capture the encrypted value. Because encrypting is optional, the source field (the one exposed to the user) should not be this special encrypted field. If encryption is configured, the system will internally populate the encrypted field. The source field will be populated with asterisks by default. That way the masked data is what is shown to the user on page rather than the encrypted value.

The following points highlight how the system behaves when encryption is configured and when it is not. Assume as an example, the field is a credit card number. The user views and populates a field with the field name CC_NBR. The table also has a second field ENCR_CC_NBR. A user populates the credit card number:

- If encryption is not configured, CC_NBR will be updated with the entered credit card number and ENCR_CC_NBR will be empty. Note that in this case, an implementation may choose to configure [user interface masking](#).
- If encryption is configured, CC_NBR will be updated with ‘*****’ and ENCR_CC_NBR will contain the encrypted value. The asterisks for the standard field will fill the full field size up to 50 characters.

If for some reason the standard masking using all asterisks is not desired, the system supports supplying an explicit masking algorithm using the same Feature Configuration - Data Masking plug-in spot used for [User Interface Masking](#).

WARNING: Unlike user interface masking, the masking of encrypted fields is not driven by security. The data stored in the source field for all encrypted data should be masked. Be sure not to configure security authorization logic in algorithms used for this type of masking.

Feature Option Configuration

Create a feature configuration with a Feature Type of **Encryption**. For each source field you are encrypting, enter an option with option type of **Field Encryption**. The value will contain mnemonics that reference the appropriate encryption key alias defined in the keystore along with configuration related to the field and its table location. Unlike the user interface data masking, the configuration for data encryption is related to how the data is stored rather than how it is displayed. In addition, each entry may define an explicit masking algorithm to override the default and if supported, may also define a hash field and hash alias.

For data that is stored in a specific column on a table, an explicit field to capture the encrypted value must exist. Indicate the table name, source field name and encrypted field name along with the alias: **table='table_name', field='fld_name', encryptedField='encr_fld_name', alias='alias key'**

A "where" clause may also be specified when data resides in a child table and only data of a certain type needs to be encrypted.

Example, **table='CI_PER_ID', field='PER_ID_NBR', encryptedField='ENCR_PER_ID_NBR', alias='key alias', where='ID_TYPE_CD='SSN''**

For data that is stored in an XML column in a record, the source field to be encrypted must reference a meta-data field name in its schema definition along with the element that captures the encrypted data and the alias: **field='field_name', encryptedField='encr_field_name', alias='key alias'**

The syntax for adding a reference to a masking algorithm is **maskAlg='algorithm name'** .

The syntax for adding configuration for capturing a hash value for searching purposes is **hashAlias='hashAliasKey' hashField='HASH_FLD_NAME'**.

The following is an example of configuration that uses all the possible options (specific masking algorithm, where clause and hash field support):

```
table='CI_PER_ID', field='PER_ID_NBR', alias='aliasKey', encryptedField='ENCR_PER_ID_NBR',  
hashAlias='hashAliasKey' hashField='HASH_PER_ID_NBR', where='ID_TYPE_CD=SSN', maskAlg='CM-  
PERIDMASK'
```

Searching by an Encrypted Value

If the product supports a hashed value for an encrypted field for searching purposes, the following points highlight explorer zone configuration for this purpose

- The user filter value should reference the source field and should include an additional **encrypt=** mnemonic. For example

```
type=STRING  
label=PER_ID_NBR  
encrypt=[CI_PER_ID,PER_ID_NBR, ID_TYPE_CD, F1]
```

Refer to [User Filters](#) for more information.

- The SQL should include the hashed value in the WHERE clause. Note that because encryption is optional, a product zone that includes searching by a field eligible for encryption will include finding a match for the filter in the source field (as plain text) or in the hashed field. For example:

```
WHERE  
  [(F2) (ID.PER_ID_NBR =:F2 OR ID.HASH_PER_ID_NBR = :F2)]
```

Customizing Encryption Algorithm

Although the encryption algorithm to use with a given key can be gleaned from the key in the keystore, there is sometimes extra information associated with an algorithm that might need to be used to encrypt or decrypt data.

The system provides a feature configuration option for the **Encryption** feature type using the option type **Algorithm Info** that can be used to adjust the behavior of the encryption.

- You can modify the default mode and padding of the encryption algorithm.
- If a key will be used to digitally sign anything, the signing algorithm can also be specified for the key.

For details about the syntax, refer to the feature option type's detailed description.

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.

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.

In addition, **SYSUSER** is provided to allow for an initial user to define appropriate users in your implementation. Once proper administrative users have been defined, it is recommended that **SYSUSER** is updated to set the User Enable setting to Disabled.

When you receive an upgrade:

- New application services are delivered for the new transactions, business objects, zones introduced in the release. The release notes highlights the additions / changes.
- 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.
- Implementations should review the release notes and determine which user groups created for your implementation should be updated with the additions, if applicable.

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.

A combination of framework configuration and configuration in your edge product is required for row level security. The following points describe the configuration:

- For each record that should be secured, associate it with an **Access Group**. Note that if your edge product supports row level security, that product is providing a link between the secure-able record and Access Group. Your access groups

may be granular and only referenced by one secured record or they may be more broad and be referenced by multiple secured records that require the same type of security restriction.

- To define which users have access to the secured records, you define a **Data Access Role**. For each data access role, define which Access Groups the role has security clearance for. An access group may be linked to one or more data access roles. In addition, define the **Users** that have access rights to these secured records. 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 secured records in all the referenced access groups. A user may belong to many data access roles.

If your edge product supports row level security, it will include logic in the appropriate areas of the system to limit the secured rows that a user may view or maintain based on this configuration. For example, in Oracle Utilities Customer Care and Billing, throughout the system users are only able to view and maintain information about an account and any of its detail if the user is in a Data Access Role for the account's Access Group (or the account is not linked to an Access Group).

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

Defining Application Services

Please refer to [Application Security](#) for a description of how application services are used when you grant user groups access rights transactions. Refer to [Action Level Security](#) for more information about the significance of an application service's access modes.

You can access the portal using **Admin > Security > Application Service**. You are brought to a query portal with options for searching for a specific application service. Once a record has been selected you are brought to the maintenance portal to view and maintain the selected record.

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

The **Application Service** zone provides the information about the application service and its access modes.

The **Secured Objects** zone displays the object (or objects) that are secured by this application service. Refer to [Application Security](#) for details about the types of objects that may be secured.

Application Security

The Application Security tab includes zones to view and manage which user groups have access to the application service and which ones do not have access.

The **User Groups Linked** zone lists the user groups that currently have a link to the application service. Note that expired links are also included.

- Click the **Description** link to navigate to the [User Group - Users](#) page for the adjacent user group. This allows you to add or remove users linked to the user group.
- Click **Deny Access** to remove the selected Application Service's link to this user group.

The **User Groups not Linked** lists the user groups that do not have a link 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 **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 allowing you to set the access modes and the expiration date.

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 programming code. 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](#).

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. Use the **App Service Description** search to restrict the application services displayed in the grid.

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.

Please note the following with respect to maintaining application services linked to the user group:

- 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 delete icon to remove the application service from the user group.

FASTPATH: Refer to [User Group Services Management](#) for a more convenient, faster method to add or remove application services to or from a user group in bulk.

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.

FASTPATH: Refer to [User Group Services Management](#) for a more convenient, faster method to add or remove application services to or from a user group in bulk.

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** 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 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](#).

User Group Services Management

This portal allows a security administrator to more quickly manage the application services and access modes linked to a user group.

Open this page using **Admin > Security > User Group Services Management**.

This section describes the available zones on this page.

The **User Groups** zone provides a list of user groups in the system. Broadcast the desired user group to enable the subsequent zones.

The **Application Services Linked** zone displays a row for each application service and access mode combination currently linked to the user group. The zone includes a filter area to limit the list based on criteria. A user can select one or more rows to remove the selection from the user group.

The **Application Services Not Linked** zone displays a row for each application service and access mode combination not currently linked to the user group. The zone includes a filter area to limit the list based on criteria. A user can select one or more rows to add the selection to the user group and is prompted for the expiration date.

Defining Access Groups

Refer to [The Big Picture of Row Security](#) for a description of how access groups are used 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. You can access the portal using **Admin > Security > Access Group**. You are brought to a query portal with options for searching for a specific access group. Once a record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The **Access Mode** zone provides the information about the access group and its data access roles.

- The **Data Access Role** collection defines the data access roles whose users have access to the access group's secured 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 objects linked to the access group.

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

- The **To Do Role Assignment** collection may be used to route To Do Entries of a given To Do Type to a specific To Do Role if the objects related to the To Do Entry are associated with the access group. Refer to [Adjusting the To Do Role](#) for more information.

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.

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.

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.

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](#) from the user menu or shortcut key.

Data Privacy

Personally Identifiable Information (PII) is any information that identifies or could be used in combination with other information to identify an individual. Many countries have data privacy laws governing the use of such information. A key aspect of these laws is an organization's obligation to erase personal information when there is no compelling reason to retain it. This obligation can be fulfilled either by deleting data or altering the data in such a way that it is no longer possible to associate the data with the individual.

Organizations are likely to have other legal obligations that impact how long they retain personal data. For example, there could be financial audit requirements that oblige the organization to retain relevant information for a fixed number of years after termination of a contract.

Object Erasure is designed to address right to erasure issues, with a combination of configuration and processes that may be extended to implement the appropriate policies for the PII within your data.

The Approach to Implementing Object Erasure

This section describes the product approach to implementing object erasure for its maintenance objects (MOs).

NOTE: The approach outlined is intended for managing erasure of master data objects. The assumption is that Information Lifecycle Management will be used for archiving any related transactional data.

Various events in the system may signal the need to erase data at some future date. For example, closing an account may indicate the need to erase both the account and person data once there is no longer a need to maintain related financial details. There may also be events that re-establish the business relationship with a person and signal that data marked for erasure should now be retained.

The Object Erasure Schedule is used to capture key details of objects with data that needs to be erased. Records in the schedule are monitored periodically to determine if the date for erasure has been reached and if so, execute the erasure logic. The system will create and maintain records in the schedule when specific events occur for an MO that is marked for erasure.

The system provides the ability to mark an MO as eligible for erasure via a number of MO configuration entries. Included in these is an option that defines the business object for the MO's erasure schedule records. In addition to defining the erasure record elements, this BO defines the monitoring process and the specific erasure logic for the MO.

The following sections provide more information about configuring and managing object erasure.

Configuring a Maintenance Object for Erasure

There are three types of maintenance object configuration entries that define how to manage erasure:

- An option to define the business object for erasure schedule records.
- An option to define the period between the date on which the need for erasure is detected and the date on which erasure should occur. The common logic provided by the system to add or update erasure records uses this option to determine the erasure date.
- One or more entries in the MO algorithm collection for the algorithms that determine whether a record should be scheduled for erasure, according to the current state of the record, and create or update the erasure schedule entries.

Some MOs that would commonly be regarded as eligible for erasure will be installed with a base Erasure BO already configured. The expectation is that the corresponding Erasure Period option and the algorithm for the Manage Erasure Schedule plug-in spot will be set up by your implementation with the appropriate entries for your organization's business rules.

NOTE: The presence of an Erasure Period option is the signal to the system that your implementation has enabled an object for erasure.

It may not be necessary for all master data objects that contain personally identifiable information to manage their own erasure schedule. For example, the event which triggers erasure for a person record may also trigger the creation of erasure schedule entries for the person's other master data records. In this case, only the person maintenance object will need to be configured with a Manage Erasure algorithm (or algorithms). Refer to your product specific documentation for more information on the recommended approaches for your product's master data objects.

Manage Erasure Schedule Algorithm

Algorithms of this type are responsible for determining a record's erasure status and creating or maintaining an entry in the Object Erasure Schedule for the record. They are triggered when certain events occur in the system for maintenance objects that are eligible for erasure. Multiple algorithms can be configured for the plug-in spot.

Depending on the determination made by the algorithm (or algorithms), a number of different actions may need to be taken. For example:

- A new schedule record may need to be added
- An existing schedule record may need to be updated with a new erasure date
- A pending record may need to be discarded if erasure is no longer applicable
- A discarded record may need to be reactivated if erasure now applies again

The base business service **F1-ManageErasureSchedule** performs the common logic to handle the possible actions and is recommended for use by these types of algorithms. The service can also optionally create a log entry for information purposes. Refer to the description of the business service for full details.

In certain circumstances, if one record is scheduled for erasure, other related records may need to be evaluated. The base business service **F1-ManageErasureScheduleDriver** performs the logic to execute an MO's Manage Erasure Schedule algorithms. It can be used to manage any related records from within another algorithm.

This type of algorithm is plugged into the [Maintenance Object — Algorithm](#) collection.

Your product may supply algorithms to manage the schedule for base MOs that are regarded as eligible for erasure. If your organization has special business rules that are evaluated to determine the erasure schedule for an MO, a custom algorithm can be created and applied by the implementation team.

Monitoring the Schedule and Performing Erasure

The erasure schedule is managed using the lifecycle of the erasure record's business object. A deferred monitor process is used to periodically check for records that are due for erasure and transition them from the 'pending erasure' state to the 'erased' state. An enter plug in on the 'erased' state is responsible for performing the erasure logic.

The system provides a 'root' business object for the object erasure schedule (**F1-ErasureScheduleRoot**) which defines the lifecycle that erasure schedule business objects should follow. The system also provides a monitor process (**F1-OESMN**) which is configured on the 'pending erasure' state of the 'root' business object. This process is configured to monitor object erasure schedule records whose erasure date is on or before the batch business date.

Maintenance objects that are eligible for erasure should be configured with an erasure schedule business object that uses the 'root' business object as its parent. The algorithm that performs the erasure processing applicable to that maintenance object must be plugged in on the 'erased' state of the 'child' business object.

The system provides an erasure schedule business object (**F1-ErasureScheduleCommon**) which is designed to erase an object by deleting the main record and any child records. Your product may supply additional erasure schedule business objects for certain use cases. Refer to your product specific documentation for more information.

Erasing User Information By Obfuscation

Some implementations consider the base User object to have personally identifiable information. Erasing user information by deleting the records is not advisable as it can cause referential integrity problems. The recommended approach is to obfuscate the data instead.

The system provides a user erasure business object (**F1-ErasureScheduleUser**) which is designed to support obfuscating a user's identifiable information by removing it or replacing it with a non-identifiable value if the field is required.

If the User maintenance object is configured to be eligible for erasure in your implementation, the manage erasure schedule algorithm for the record will be invoked whenever the **Enable** flag is changed for a record on the User maintenance page. The system provides a base algorithm (**F1-OBJERSUSR**) specifically for the User MO. This algorithm will add an entry to the object erasure schedule if the user record is disabled or deactivate an existing erasure schedule record if a disabled user record is enabled again.

Viewing an Object's Erasure Status

The system provides a portal for viewing and editing an erasure schedule record. The portal is accessed from the **Object Erasure Schedule** dashboard zone. This is a context sensitive zone which only appears when accessing a record for a maintenance object that is configured to be eligible for erasure. It displays the information for the erasure schedule entry for the record in context, if applicable.

To add the zone to an eligible maintenance object:

- Navigate to [Context Sensitive Zone](#) and search for the navigation key for the maintenance object.
- Add the Object Erasure Schedule zone **F1-OBJERSRSCD** to that navigation key.

Viewing Erasure Configuration

The Erasure Configuration [All-in-One portal](#) provides the ability to view the maintenance objects that are marked as eligible for erasure and maintain their erasure configuration entries in a single place.

To view a list of the eligible MOs, open **Admin > Security > Erasure Configuration**.

Press the edit icon to open a window in which the configuration values for that row can be changed.

Refer to [The Approach to Implementing Object Erasure](#) for an overview of object erasure configuration.

Archiving the Object Erasure Schedule

The system provides the ability to archive older records in the object erasure schedule using Information Lifecycle Management. The object erasure schedule maintenance object is configured with a base ILM eligibility algorithm and an ILM crawler batch control.

The base object erasure schedule BO is configured to set the ILM archive switch when an erasure schedule record enters a final state and to reset it if the record becomes active again.

Refer to [The Approach to Implementing Information Lifecycle Management](#) for more information on archiving functionality.

Cryptography Keys

Often when communicating information with an external system, cryptography keys are used to exchange encrypted information or confirm that the two communicating parties recognize each other and the information being provided.

The following sections include information about the functionality provided to support this functionality.

Understanding Key Rings

Cryptography keys may be used to provide a signature to a request so that the system recognizes that the request comes from a trusted party. Keys may also be used to encrypt or decrypt files shared between two parties.

In this release, support for a Signature type of key is provided to be able to access files stored in Oracle Cloud Object Storage.

The product supports the ability for the following:

- Maintaining pairs of keys - public and private. For keys generated in the system, the private key is stored in an appropriate "secret store" and the public key is available to copy and share with a third party.
- Key rotation. For increased security, a new key pair should be generated periodically.

The Key Ring object is provided to reference the key pairs that are used over time for a given business use case. Only one key pair may be active at any given time.

The following sections include information about the functionality provided to support this functionality.

Signature Keys

Oracle Cloud Object Storage is used primarily by cloud customers to store any files that are needed by the system. Certain processes in the system may need to write files to or read files from cloud object storage. Cloud object storage has its own method for encrypting files and products using the framework application do not need to define additional configuration for that. However, when communicating with cloud object storage, the application is required to provide a signature key.

Refer to [External File Storage](#) for more information about configuring the system to connect to cloud object storage.

The product provides a Signature Key Ring business object that supports the following functions:

- Generating a key pair, for key rotation
- Viewing the public key, allowing a user to copy it to register it with cloud object storage.
- Activating the new key pair (to be done after registering the new public key)

Defining Key Rings

Refer to [Understanding Key Rings](#) for an overview of key ring functionality.

To maintain the key rings applicable to your product or implementation, open **Admin > Security > Key Ring**.

This is a standard [All-in-One portal](#) and includes the standard List and display zones for a key ring.

The information captured on the key ring depends on the business objects supported by your product or implementation. Refer to the embedded help text for more information.

Key Pairs

The key pair zone provides information about the keys that are associated with the key ring. Actions available on the zone depend on the type of key. Refer to embedded help for more information.

User Interface Tools

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

Understanding 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.

There are three broad classes of portals:

- **Standalone.** 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.** 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.** 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.

The contents of this section describe general information about portals and zones.

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.

Not All Zones Make Up Portals

Zones may or may not be associated with a portal. The following describes additional uses of zones:

- A data explorer type of zone may implement a search for an entity on an input user interface map. These zones may be referenced on a FK reference record or directly referenced on a UI map.
- A data explorer type of zone may implement a Business Service. The latter may be used for an internal process or used to build a user interface dropdown.
- A zone may be defined as a **Context Sensitive Zone** that appears on the Dashboard when a specific page or portal is in context. This type of zone is not linked to the Dashboard portal but associated with the context page or portal’s navigation key.

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

Flexible Layout Control

By default a zone spans the full width of the portal's display area but can be configured in a number of ways using the layout control features on the portal.

These features allow portal designers to have finer-grained control over how zones are laid out, and enables them to achieve a better user experience and a more optimal layout that reduces unused space.

The following highlight key layout control options:

- In addition to the standard half and full width options, 1/4, 3/4, 1/3 and 2/3 width options are also available.
- Better handling of zones that are dynamically hidden by a visibility script. When a zone becomes invisible its space may be reclaimed by a next zone as long as it is of same width or smaller.
- A zone may define the beginning of a new row. This indication allows for a more fixed layout that is not impacted by hidden zones. The new row indication may establish a layout of one, two or three zones displayed vertically.
- Support the following zone height options:
 - **Content (Unlimited Height)** lets the height be determined by content.
 - **Content (Limited Height)** lets the height be determined by content up to a specified maximum limit. This option may be used to achieve a more consistent portal layout, especially when a zone is rendered on long or small devices.
 - **Fixed Height** sets the zone at a specific height regardless of content.

This option too may be used to achieve a more consistent portal layout.

- When a height option is not explicitly selected, zone height is unlimited and determined by content. However, for a data explorer zone, the results grid area itself (not the entire zone) is limited by the zone parameter "height of report" (which has a default value of 50%).

User Preferences

A **Standalone** portal includes configuration to indicate whether or not the portal should appear on a user's portal preferences. The portal will not appear on the user's portal preferences if the **Show on Portal Preferences** setting is set to No. Note that an implementation may change this value for a product delivered portal. This setting 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.

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

Removing Zones From 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.

Granting Access to A Portal

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

When you add a new **Standalone** portal, the system automatically creates its application service behind the scenes and displays it as part of the portal definition. You'll need to use this application service to grant access to the portal. **Tab Page** 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.

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.

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. 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 portal. In other words, if the user has access to the page, then the user 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 (standalone) 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).

Putting Portals on Menus

When you add a new **Standalone** portal, the system automatically creates its navigation option behind the scenes and displays it as part of the portal definition. You'll need to use this navigation option to add the portal to a menu or to allow any other type of navigation method to the portal.

To add a portal to a menu, you must add a [menu item](#) to the desired menu and reference the portal's navigation option on the new menu item. 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).

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.

Portal Options

Various portal features are controlled by configuration options, such as the ability to include additional applicable actions on the portal's actions bar or the ability to include the standard information description of the maintained entity as part of the portal's title.

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**.

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 Types 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.

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

This portal is used to view and maintain the configuration of a zone. Implementations may define custom zones or override the descriptions and some parameter values for base product zones.

You can access the portal from the **Admin Menu > System > Zone**. You are brought to a query portal with options for searching for a specific zone. Once a zone record has been selected you are brought to the maintenance portal to view and maintain the selected record.

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

Refer to [Understanding Portals and Zones](#) for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Zone.** This zone provides general information about the selected zone.
- **Query Options.** This zone maintains the list of query options for a Multi-Query zone. The zone is only visible for Multi-Query zones.
- **Filters.** This zone maintains various zone parameters that handle filtering options for a data explorer zone. The zone is only visible for data explorer zones.
- **General Parameters.** This zone maintains various general purpose zone parameters for a data explorer zone.
- **SQLs.** This zone maintains individual SQLs and their related parameters. The zone is only visible for data explorer zones.
- **SQL Configuration.** This zone maintains all the zone parameters related to a specific data explorer SQL. The zone is only visible when a specific SQL is broadcast from the **SQLs** zone.

The following zones may appear as part of the portal's **References** tab page

- **Zone References.** This zone list various configuration entities referencing the current zone.

Overriding Parameter Values

Implementation teams may override the value of selected types of base provided zone parameters. The override parameter value field for base owned zone parameters is only enabled when the related zone type parameter usage is **Zone - Override Allowed**.

For multi-query zones, you may use the override parameter value to add custom query options to a base product owned zone.

Overriding Zone Parameters

Implementations may override the value of a base owned zone parameter only if the zone type definition allows such override. The override parameter value is enabled for a specific base owned parameter when the related zone type parameter usage is set to **Zone - Override Allowed**.

You may use the override parameter capability to provide alternative or additional custom query options for a base owned multi-query zone.

Zone Parameter Details

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

Zone Visibility Service Script

All zones support a visibility script that is used to determine if the zone should be displayed to the user or not based on conditions. The script may receive input parameters and is expected to return a Boolean value indicating if the zone should be displayed or not. The embedded help for the **Zone Visibility Service Script** parameter provides details related to the syntax.

The following table highlights some service scripts provided by the product that may be used if applicable to your zone's requirements. This is not an exhaustive list of visibility scripts. There may be others that are specific to a given zone.

Script Code	Description	Comments
F1-ShldShwZn	Zone Visibility - Display Zone in Portal	This script simply returns a value of 'true' and is used when the zone should always appear.
F1-CondShwZn	Zone Visibility - Display Zone in Portal Conditionally	This is used when the condition for showing the zone is based on the population of a context value. This is commonly used when one zone in the portal should only appear after a broadcast of a record from another zone in the portal. For an example of a zone that uses this visibility script, refer to F1-BSFTYPE .
F1-RwCtShwZn	Zone Visibility - Based on Row Count	This is used when the condition for showing the zone is based on the existence of one or more rows that can be determined using SQL. This script accepts a zone code, user filters 1 through 25 and hidden filters 1 through 10. The script returns an indication of 'true' if at least one row count is returned by the zone. To use this visibility script, a specific data explorer zone must be developed for the specific use case.

Script Code	Description	Comments
		For an example of a zone the uses this visibility script, refer to F1-MIGRREQEL .

SQL Statement

Data explorer zones are used to select data to display using one or more SQL statements. The SQL parameters are applicable to the following zone types

- Info Data Explorer - Single SQL (**F1-DE-SINGLE**). The parameter has the description **SQL Statement**.
- Info Data Explorer - Multiple SQLs (**F1-DE**). The parameters follow the description pattern of **SQL Statement x**.
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**). The parameters follow the description pattern of **SQL Statement x**.

NOTE: If your implementation has been configured to restrict the functions that may be used when defining an SQL then an error is issued at runtime if there are functions found that are not in the allowlist. The allowlist may be viewed using the **View SQL function allowlist** link in the Tips zone on the zone maintenance page.

The following table provides a list of SQL substituted keywords that may be used in the SQL Statement parameters in explorer zones. At execution time, the system determines the database and substitutes the keyword with the database specific syntax:

Keyword	Description	Examples
@toCharacter()	Converts the input to Character data type.	select @toCharacter (batch_cd) as batchCode from ci_batch_ctrl
@toDate()	Converts the input to Date data type.	select @toDate (last_update_dttm) as lastUpdateDate from ci_batch_ctrl
@toNumber()	Converts the input to Number data type.	select @toNumber (next_batch_nbr) from ci_batch_ctrl
@currentDate	Fetches the current date. CAUTION: The Oracle functions SYSDATE and CURRENT_DATE should not be used because they do not properly cater for adjusting dates from the database time zone to the installation time zone, if needed.	select batch_cd, @currentDate as today from ci_batch_ctrl
@currentTimestamp	Fetches the current date / time. CAUTION: The Oracle functions SYSTEMSTAMP and CURRENT_TIMESTAMP should not be used because they do not properly cater for adjusting the date / time from the database time zone to the installation time zone, if needed.	select batch_cd from ci_batch_ctrl where last_update_dttm > @currentTimestamp
@concat	Combines the result list of two or more columns.	select batch_cd @concat next_batch_nbr concatNbr from ci_batch_ctrl
@substr(string, start)	String is the input String that you are trying to get a substring of. Start is the position of the character for the output results.	select batch_cd batchCode from ci_batch_ctrl Result: TESTCD select @substr (batch_cd,3) batchCode from ci_batch_ctrl Result: STCD
@substr(string, start, end)	String is the input String that you are trying to get a substring of. Start is the position of the character for the output results.	Select batch_cd batchCode from ci_batch_ctrl Result: TESTCD select @substr (batch_cd,3,2) batchCode from ci_batch_ctrl

Keyword	Description	Examples
	End is the number of characters required in the output from starting position.	Result: ST
@trim	Trims the white spaces of the output on both sides.	select @trim (batch_cd) as batchCode from ci_batch_ctrl
The following syntax is related to 'fuzzy' searching. It is only applicable if Oracle DB Text is enabled and a context text index has been created. Refer to Advanced Search Options for more information.		
@fuzzy(string, score, numresult, 'weight')	<p>String is the input value for the search.</p> <p>Score is the degree of 'fuzziness'. Valid values are between 1 - 80. The higher the number the more precise the search. Default is 60.</p> <p>Numresults is the number of variations to consider for the string. Valid values are between 1 and 5000. Default is 100.</p> <p>Indicate 'weight' to signal that the results are returned in order of weight. Leave this setting off to indicate that the results are returned in order of score.</p>	<p>Set score to 70, number results to 6, and specify weight.</p> <pre>select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1,70, 6, 'weight')) > 0</pre>
@fuzzy(string)	This returns a string result from the fuzzy expansion operation where the default value of 60 is assumed for the score and the default value of 100 is assumed for the numresult .	<p>To use default values:</p> <pre>select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1))> 0</pre>
@fuzzy(string, score)	This returns a string result from the fuzzy expansion operation with the score specified and the default value of 100 for the numresult .	<p>Set score to 70.</p> <pre>select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1,70)) > 0</pre>
@fuzzy(string, score, numresult)	This returns a string result from the fuzzy expansion operation with the similarity score and the numresults specified.	<p>Set score to 70, number results to 6.</p> <pre>select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1,70, 6)) > 0</pre>

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.

These parameters are applicable to the zone types

- Info Data Explorer - Single SQL (**F1-DE-SINGLE**). The parameters follow the description pattern of **Column x**.
- Info Data Explorer - Multiple SQLs (**F1-DE**). For this zone type, all SQLs are executed and the zone displays a union of all the results. The parameters follow the description pattern of **Column x for SQL y**. There are some mnemonics that do not make sense to differ within rows of the same column. For example, the column label. For these mnemonics, only the value in SQL 1 is considered for that column. The table below indicates which mnemonics follow this rule.
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**). For this zone type, only one SQL is executed. The **SQL y Condition** parameter may be used to control this. The system will execute the first SQL whose condition is satisfied (or with no condition populated). The parameters follow the description pattern of **Column x for SQL y**.

The following sections describe the various types of mnemonics.

Contents

- [Source Mnemonics](#)
- [Formatting Mnemonics](#)
- [Click Mnemonics](#)
- [Common Use Cases](#)

Source Mnemonics

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

Mnemonic	Description	Valid Values	Comments
source=	Defines how the column's value is derived.	SQLCOL	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 sqlcol= mnemonic.
		BO	Indicates that the source of the column's value comes from a business object . This type of column must also reference the bo= , input= and output= mnemonics to define how to interact with the business object.
		BS	Indicates that the source of the column's value comes from a business service . This type of column must also reference the bs= , input= and output= mnemonics to define how to interact with the business service.
		SS	Indicates that the source of the column's value comes from a service script . This type of column must also reference the ss= , input= and output= mnemonics to define how to interact with the service script.
		FORMULA	Indicates that the source of this column's value is calculated using a formula. This type of column must also reference the formula= mnemonic.
		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 setfunc= mnemonic.
		ICON	Indicates that the source of this column's value is a display icon reference (meaning that an icon will be displayed in the column). This type of column must also reference the icon= mnemonic to define the icon reference. NOTE: When using this source mnemonic, the formatting mnemonic type= is not applicable.
		FKREF	Indicates that the source of this column's value is an FK reference (meaning that the FK reference's context menu and information string will be displayed in the column and

Mnemonic	Description	Valid Values	Comments
			<p>will be enabled for navigation). This type of column must also reference the fkref= and input= mnemonics to define how the FK reference is called.</p> <p>NOTE: When using this source mnemonic, the formatting mnemonictype= is not applicable.</p> <p>NOTE: To show a foreign key's information but not enable navigation, refer to Common Use Cases for the recommended technique.</p>
		SPECIFIED	<p>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 spec= mnemonic.</p>
		MSG	<p>Indicates that the source of this column is a message from the message table (along with any substitution variables). This type of column must also reference the msg= mnemonic.</p>
sqlcol=	Defines the column in the SQL statement when source=SQLCOL .	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, sqlcol=3 would display the 3rd column in the SELECT statement).
bo=	<p>Defines the business object to invoke when source=BO.</p> <p>This mnemonic must be used in conjunction with the input= and output= mnemonics to define how information is sent to / received from the business object.</p>	'Business Object Code'	
bs=	<p>Defines the business service to invoke when source=BS.</p> <p>This mnemonic must be used in conjunction with the input= and output= mnemonics to define how</p>	'Business Service Code'	

Mnemonic	Description	Valid Values	Comments
	information is sent to/received from the business service.		
ss=	<p>Defines the service script to invoke when source=SS.</p> <p>This mnemonic must be used in conjunction with the input= and output= mnemonics to define how information is sent to / received from the service script.</p>	'Service Script Code'	
fkref=	<p>Defines the FK reference used to retrieve the column's information when source=FKREF.</p> <p>This mnemonic must be used in conjunction with the input= mnemonic to define how information is sent to the FK reference to build the information.</p>	<p>Cx</p> <hr/> <p>COLUMN_NAME</p> <hr/> <p>'FK Reference Code'</p>	<p>This means FK reference code is defined in an earlier column. For example, define C1 if column 1 defines the FK reference value.</p> <hr/> <p>This means the FK reference was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.</p> <hr/> <p>This means the FK Reference is defined directly. For example 'F1-ROLE'.</p>
formula=	<p>Defines the formula to use when source=FORMULA.</p> <p>Examples:</p> <ul style="list-style-type: none"> formula=C1*.90/C2 formula=(C1/C2)*100 	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> <hr/> <p>Refer to Expression Parser for information about the functions supported.</p>
setfunc=	<p>Defines the function to apply to the rows of a given column when source=SETFUNC.</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"> MAX. This derives the maximum value of all rows in the column. MIN. This derives the minimum value of all rows in the column. TOT. This derives the sum (total value) of all rows in the column. ACC. This derives the cumulative total of all rows up to an including the current row.
input=	<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>Cx</p> <hr/> <p>COLUMN_NAME</p>	<p>Where Cx represents the value of a previous column. If the value to pass is in the first column, reference C1.</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>

Mnemonic	Description	Valid Values	Comments
	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.	'literal value' userTimeZone installationTimeZone	This means a literal value within the single quotes should be passed in. This means the current user's time zone should be passed in. This is typically used with the business service F1-ShiftDateTime to convert data in the storage time zone to the user's time zone for display. This means the installation time zone should be passed in. This is typically used with the business service F1-ShiftDateTime to convert data in the storage time zone to the installation time zone for display.
		Examples: <ul style="list-style-type: none"> • input=[USER_ID=C1] • input=[USER_ID=USER_ID] • input=[input/targetTimeZone=userTimeZone] 	
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: output=personInfo
pagingkey=	This mnemonic is only applicable when the Enable Paging parameter has been configured. It indicates that this column is one of the keys used by the SQL statement to orchestrate paging through results. This mnemonic can only be specified when the source=SQLCOL . FASTPATH: Refer to Pagination Configuration for more information.	Y N	This is the default, meaning that you don't need to indicate pagingkey=N at all to indicate that the column is not one of the paging keys.

NOTE: If multiple columns are configured with the same source BO, BS or SS and the same input data, the system caches the output from the first call and reuses the results for subsequent columns.

Formatting Mnemonics

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

NOTE: For the **F1-DE** zone type, zone displays the union of all the different SQLs.

Mnemonic	Description	Valid Values	Comments
type=	Defines how the column's value is formatted.	STRING	Columns of this type capture a string. This is the default value.

Mnemonic	Description	Valid Values	Comments
	<p>NOTE: Icon and Foreign Key columns. The source=source mnemonic 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 type= mnemonic is not relevant as either an icon or a context menu / info string will appear in the column.</p>	DATE	Columns of this type capture a date and will be displayed using the user's display profile .
		TIME	Columns of this type capture a time (in database format) and will be displayed using the user's display profile .
		DATE/TIME	Columns of this type capture a date and time (in database format) and will be displayed using the user's display profile .
		MONEY	Columns of this type capture a monetary field. This type of column may also reference the cur= mnemonic. If the cur mnemonic is not specified, the currency code on the installation record is used.
		NUMBER	Columns of this type capture a numeric field. This type of column may also reference the dec= mnemonic.
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 source. Note that some sources don't have a default value and omitting this mnemonic will result in a blank label.</p> <p>NOTE: In the case of the zone type Info Data Explorer - Multiple SQLs (F1-DE), the system will use the column definitions in SQL 1 to define the labels.</p>	FIELD_NAME	Enter a valid field 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 currency code applied when type=MONEY 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 C1 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'.

Mnemonic	Description	Valid Values	Comments
dec=	<p>Defines the number of decimal places when type=NUMBER.</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. Simply specifying n (without an R) means that decimal places should be truncated. For example, entering dec=4 will display 4 decimal places and truncate the remainder.</p> <p>NOTE: Formatting only. 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 dec=0, the column will be shown with no decimal places (truncated), however any references to the column in subsequent calculations 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>
char=	<p>This mnemonic applies special character(s) to the column's value.</p>	'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, char='%' will place a percent sign after the value. If you want to position the word 'minutes' before a value, enter char='minutes []'. If you want to output a value like BUDGET \$123.12 (YTD), enter char='BUDGET [] (YTD)'.</p>
suppress=	<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>NOTE: In the case of the zone type Info Data Explorer - Multiple SQLs</p>	<p>true</p> <p>false</p>	<p>This is the default, meaning that you don't need to indicate suppress=false at all to indicate that the field should be shown.</p>

Mnemonic	Description	Valid Values	Comments
	(F1–DE), the system will apply the settings for the column definitions in SQL 1 to all subsequent SQLs.		
suppressSearch=	<p>This is used to indicate a column should not be displayed when the zone is invoked in search mode only.</p> <p>NOTE: In the case of the zone type Info Data Explorer - Multiple SQLs (F1–DE), the system will apply the settings for the column definitions in SQL 1 to all subsequent SQLs.</p>	<p>true</p> <p>false</p>	<p>This is the default, meaning that you don't need to indicate suppressSearch=false at all to indicate that the field should be shown.</p>
suppressExport=	<p>This is used to indicate a column should not be downloaded to Excel.</p> <p>NOTE: In the case of the zone type Info Data Explorer - Multiple SQLs (F1–DE), the system will apply the settings for the column definitions in SQL 1 to all subsequent SQLs.</p>	<p>true</p> <p>false</p>	<p>This is the default, meaning that you don't need to indicate suppressExport=false at all to indicate that the field should be included in a download.</p>
width=	<p>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.</p>	n	<p>Where n is a number between 0 and 999.</p> <p>NOTE: If there is no available breaking point in the data, the column will be longer than the specified number of pixels. The length of the column's label (which appears in the column's heading) may also make the width wider than specified. In the case of the zone type Info Data Explorer - Multiple SQLs (F1–DE), the system will set the width based on the column definitions in SQL 1.</p>
color=	<p>This is used to override the column's text color.</p>	A valid HTML "named" color	<p>For example color=yellow.</p> <p>NOTE: Refer to Color Contrast for information about the use of the HTML color 'red' and its impact on accessibility.</p>

Mnemonic	Description	Valid Values	Comments
		A valid RGB color model combination	For example color=#E0292F or color=#CCCCCC . Note that the # is required.
bgcolor=	This is used to override the column's background color.	A valid HTML "named" color A valid RGB color model combination	Similar to the color= mnemonic. Similar to the color= mnemonic.
order=	Defines the column's default sort order.	ASC DESC	Indicates that the order is ascending. This is the default meaning that it is not necessary to indicate order=ASC . Indicates that the order is descending.
rowHeader=	Designates the column as a row header for accessibility purposes.	true	By default, the first data column of the data explorer results is identified as the row header, for accessibility tools. If the data in the first column does not uniquely identify the row, use this mnemonic to explicitly mark a different column or multiple columns as the row header. NOTE: In the case of the zone type Info Data Explorer - Multiple SQLs (F1-DE), the system will look only at the columns in SQL 1 that have this mnemonic defined and apply it to all results for those columns.

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
navopt=	Defines the navigation option that references the target transaction or script when the user clicks a column. Note, this mnemonic should be used in conjunction with the context= mnemonic to define what information is sent to the navigation option's target transaction. This mnemonic is ignored if source=FKREF because the FK reference code defines the hyperlink's destination.	Cx COLUMN_NAME 'Navigation Option Code'	This means navigation option code is defined in an earlier column. For example, define C1 if column 1 defines the navigation option. This means the navigation option was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause. Example: navopt=MAIN_PORTAL This means the navigation option code is defined directly. For example navopt='userMaint' .
context=	This is used to define one or more context fields and values passed to the target navigation option to go along with the navopt= mnemonic.	Cx	Where Cx represents the value of a previous column. For example, if the value to pass is in the first column, reference C1 .

Mnemonic	Description	Valid Values	Comments
	<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>	<p>COLUMN_NAME</p> <hr/> <p>'literal value'</p>	<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>
bpa=	<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 tempstorage= mnemonic to define the temporary storage values that will be initiated when the script is executed.</p> <p>This mnemonic is ignored if source=FKREF because the FK reference code defines the hyperlink's destination.</p>	<p>Cx</p> <hr/> <p>COLUMN_NAME</p> <hr/> <p>'BPA Script Code'</p>	<p>Indicates that the BPA script is defined in a previous column.</p> <hr/> <p>This means the BPA script to execute was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.</p> <hr/> <p>This means that the BPA script to execute is defined directly.</p>
tempstorage=	<p>This is used to define how temporary storage variables are initiated when the bpa= 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 C1.</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>
list=	<p>This is used to enable work list capability for this column.</p> <p>You may optionally populate the listdesc= mnemonic to override the text that will be placed in the worklist zone.</p>	true	<p>Setting list=true 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 work list zone.</p> <p>NOTE: In the case of the zone type Info Data Explorer - Multiple SQLs (F1-DE), the output may be showing a union of the results</p>

Mnemonic	Description	Valid Values	Comments
			of multiple SQL statements. In this case, if some of the SQL statements configure a given column with list=true , 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. Also note that when determining which columns should have the worklist icon when building the zone, the system only looks at the configuration for the columns in SQL 1.
listdesc=	This is an optional mnemonic when using the list= 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 C1 .
listbroadcast=	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.	true	Use this setting to turn on the feature.

Common Use Cases

This section provides some common configuration that can be copied and pasted for newly created zones.

Use Case	Code Snippet
Displaying the description of a lookup value that has been retrieved in the SQL statement. The business service checks for an override label.	<pre>source=BS bs='F1-GetLookupDescription' label=FLD_NAME input=[fieldName='FLD_NAME' fieldValue=FLD_NAME] output=description</pre>
Display a foreign key with its information but suppressing the navigation. (This is useful when you have a foreign key that is additional information in a zone but would cause confusion to enable navigation. To display foreign key reference with its information and enabled for navigation, use the source of FKREF as described above.)	<pre>source=BS bs='F1-GetForeignKeyReference' input=[input/fkReference=FK_REF_CD input/fkValue1=BUS_OBJ_CD] output=output/infoDescription label=DESCR</pre>
The first example assumes that you know the FK Reference. The second example uses a business service that receives the maintenance object as input.	<pre>source=BS bs='F1-GetFKReferenceDetails' input=[input/maintenanceObject='F1-EXT LKUP' input/pkValue1=BUS_OBJ_CD input/pkValue2=C1]</pre>

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
type=	This mnemonic defines the appearance of the action in the zone header.	LINK	Indicates that the action is shown as a textual hyperlink.
		ICON	Indicates that the action is shown as a graphical icon.
		BUTTON	Indicates that the action is shown as an HTML button.
		ASIS	Indicates that the parameter will provide the HTML to be used for the action.
action=	This mnemonic defines the action to take when the link/icon/button is clicked. This is ignored when the type=ASIS .	NAVIGATION	Indicates that the action is navigation to a page.
		SCRIPT	Indicates that the action is to run a BPA script.
navopt=	Defines the navigation option to use when the action=NAVIGATION .	'NAV_OPT_CD'	Enter a reference to a valid navigation option in single quotes.
bpa=	Defines the script to run when the action=SCRIPT .	'SCRIPT_CD'	Enter a reference to a valid BPA script in single quotes.
icon=	Indicates the icon to use when type=ICON .	DISP_ICON_CD	Enter a reference to a valid display icon.
		'path'	Enter an explicit path to the icon, for example 'images/gotoZone.gif'.
asis=	This is required when the type=ASIS . 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']	
label=	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.
context=[target1=source1 target2=source2]	This is used to pass context data when navigating to a	FIELD_NAME	Indicates that the value should be taken from the field with

Mnemonic	Description	Valid Values	Comments
	<p>page or executing a BPA script. The mnemonic supports passing multiple values.</p> <p>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.</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>xpath</p> <p>'constant'</p>	<p>this name from portal context, global context or the page data model. The mnemonic sourceLoc is used for defining the source.</p> <p>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.</p> <p>Indicates that the value defined in single quotes should be passed.</p>
sourceLoc=	<p>This mnemonic defines the source of the FIELD_NAME's value in the context mnemonic.</p> <p>If this mnemonic is left blank, the default behavior is as follows:</p> <ul style="list-style-type: none"> - The portal context is checked. - If no portal context value is found, the global context is checked. - If neither value is available, the field is ignored. 	<p>G</p> <p>P</p> <p>D</p>	<p>Indicates that the field's value is retrieved from the global context.</p> <p>Indicates that the field's value is retrieved from the portal context.</p> <p>Indicates that the field's value is retrieved from the page data model.</p>
class=	<p>Use this mnemonic to override the look and feel of the link / icon / button using a different CSS style.</p>	<p>'className1' 'className2'</p>	<p>Enter one or more classes in single quotes. Multiple class names may be provided.</p>
style=	<p>Use this mnemonic to override the look and feel of the action element using the indicated css style.</p>	<p>Standard style= format.</p>	<p>All allowed css style definitions may be used.</p>

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=['Search']**

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
name=	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.	MD Field Name	
datasource=	<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"> - If the field has been broadcast from another zone, the broadcast value is used. - If no value is broadcast, the portal context is checked to determine if this field exists (if so, its value is taken). - If still no value, the global context is checked. - If still no value, no default value is shown. 	G	Indicates that the zone should look for the filter value in global context.
		P	Indicates that the zone should look for the filter value in portal context.
		D	Indicates that the zone should look for the filter value in the page data model.
type=	Defines the visual metaphor used to capture the filter values.	DATE	Filters of this type capture a date.
		DATE/TIME	Filters of this type capture a date and time.
		STRING	Filters of this type capture a string
		MONEY	Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic.
		NUMBER	Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic.
		LOOKUP	Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic.
		TABLE	Filters of this type capture an administrative table's value (code and description). This type of filter must also reference the table mnemonic.
		CHARTYPE	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.
label=	Defines the filter's label that appears in the zone's description bar and in the input area.	MD Field Name	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.
		'text'	Defines the text directly.
cur=	Defines the currency code applied when type=MONEY .	Currency Code	Enter a reference to a valid currency code.
dec=	Defines the number of decimal places when type=NUMBER .	Valid number	It is optional. If provided it should be an integer. If not provided, the

Mnemonic	Description	Valid Values	Comments
			number of decimals will default to the number of decimal places defined on the currency code specified on the installation record.
lookup=	Defines the lookup flag whose values appear when type=LOOKUP .	Lookup Field Name	Enter a reference to a valid lookup field name.
table=	Defines the admin table whose values appear when type=TABLE .	Table Name	Enter a reference to a valid control table name.
chartype=	Defines the characteristic type code whose values appear when type=CHARTYPE .	Char Type code	Enter a reference to a valid characteristic type code.
xpath=	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.	XPath	The type= mnemonic must also be appropriate for the map's input field, otherwise the query's SQL could fail.
likeable=	This mnemonic defines if a likeable search is performed on the entered value when type=STRING .	S	The query will add % to the suffix of the filter value.
		P	The query will add % to the prefix of the filter value.
		PS	The query will add % to the prefix and suffix of the filter value.
divide=	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., divide=above divide=below .	above	This results in a divider line placed above the filter.
		below	This results in a divider line placed below the filter.
searchField=	This mnemonic controls the initial population of the filter when the zone is launched as a search from a UI map.	MD Field Name	Enter the field name that exactly matches the searchField name specified in the oraSearchField HTML element in the UI map.
encrypt=	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] NOTE: The field name referenced here should be the source value of the field. However, the SQL should use the hashed value in its filter.	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 if another field has a certain value. The following is an example. encrypt=[CI_PERSON,PER_ID_NBR,ID_TYPE_NBR,'SSN'] . The WHERE_VALUE may also reference another filter. The following is an example. encrypt=[CI_PERSON,PER_ID_NBR,ID_TYPE_NBR,F1] .

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
name=	This mnemonic defines the name of the field that needs to be broadcast from other zones or populated in the portal context	FIELD_NAME	
datasource=	This mnemonic defines the source of the hidden filter's value. If this mnemonic is left blank, the default behavior is as follows: - If the field has been broadcast from another zone, the broadcast value is used. - If no value is broadcast, the portal context is checked to determine if this field exists (if so, its value is taken). - If still no value, the global context is checked. - If still no value, the zone appears as per the poprule mnemonic.	G P D	Indicates that the zone should look for the filter value in global context. Indicates that the zone should look for the filter value in portal context. Indicates that the zone should look for the filter value in the page data model.
poprule=	This mnemonic controls what happens if the hidden filter is not present.	R O	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. Indicates that the value is optional. If no value is required, the zone is still built without that value.
type=	Defines the visual metaphor used to capture the filter values.	DATE DATE/TIME STRING MONEY NUMBER	Filters of this type capture a date. Filters of this type capture a date and time. Filters of this type capture a string Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic. Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic.

Mnemonic	Description	Valid Values	Comments
		LOOKUP	Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic.
		TABLE	Filters of this type capture an administrative table's value (code and description). This type of filter must also reference the table mnemonic.
		CHARTYPE	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.
		ASIS	Filters of this type capture a list of values to be referenced within an 'IN' clause within the SQL statement.
label=	Defines the filter's label that appears in the zone's description bar.	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'	Defines the text directly.
cur=	Defines the currency code applied when type=MONEY .	CURRENCY_CD	Enter a reference to a valid currency code.
dec=	Defines the number of decimal places when type=NUMBER .	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.
lookup=	Defines the lookup flag whose values appear when type=LOOKUP .	LOOKUP_FIELD_NAME	Enter a reference to a valid lookup field name.
table=	Defines the admin table whose values appear when type=TABLE .	TABLE_NAME	Enter a reference to a valid admin table name.
chartype=	Defines the characteristic type code whose values appear when type=CHARTYPE .	CHAR_TYPE_CD	Enter a reference to a valid characteristic type code.
searchField=	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
script=	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.
type=	This mnemonic defines how the action should be rendered.	BUTTON	The action is rendered as a button. This is the default.
		LINK	The action is rendered as hypertext.
		ICON	The action is rendered as a graphic icon. For this option, the icon mnemonic is required.
icon=	This mnemonic defines the icon to display when type=ICON .	DISPLAY_ICON_CD	Enter a reference to a valid display icon code.
refresh=	This mnemonic indicates how and if a refresh should occur after the script completes.	NO	Indicates that no refresh is performed. This is the default.
		ZONE	Indicates that a refresh of the zone is performed.
		PORTAL	Indicates that a refresh of the entire portal is performed.
label=	By default, the button label, link text or icon tooltip will come from the 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.
list=	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.
context=[elementName1=rowData1 elementName2=rowData2]	This mnemonic is used to populate the list with the appropriate information from each selected row. The mnemonic supports passing multiple values. 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.	Cx	Indicates that the element should be populated with a value in the referenced column parameter.
		Px	Indicates that the element should be populated with a value in the referenced post processing parameter.
		COLUMN_NAME	Indicates that the element should be populated with a value from a column in the SQL statement.
		'constant'	Indicates that the value defined in single quotes should be passed.

Mnemonic	Description	Valid Values	Comments
	<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="610 426 935 590"><schema> <accountInfo type="list"> <accountId/> <name/> <amount/> <process/> </accountInfo> </schema></pre> <p>Example of list and context mnemonics.</p> <p>list=accountInfo</p> <p>context=[accountId=ACCT_ ID name=C2 amount=P3 process='O']</p>		
class=	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.
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.

Data Explorer Graph Configuration

For some data explorer zones, it is more effective to display the results in a graphical form such as a pie or line chart. Visualization option parameters enable the graph configuration to be integrated with the data explorer zone that retrieves the data.

The visualization option parameters support three options:

- **Grid** - displays the data in a standard data explorer grid
- **Chart** - displays the data in the form of a chart
- **Map** - provides the ability to invoke a UI Map to display the data

Up to five visualization option parameters may be specified. One of the options must be the standard grid, allowing for a maximum of four graphical options. The zone will display icons corresponding to the visualization types to allow you to switch between the various views. These icons are presented in the same order as they are defined in the parameter list.

NOTE: If the only display option is a standard grid, it is not necessary to specify the grid as a visualization parameter.

Visualization option parameters are provided on the Info Data Explorer - Single SQL (**F1-DE-SINGLE**) and Info Data Explorer - Multiple SQLs (**F1-DE**) zone types.

Visualization Option Parameter Configuration

Mnemonics are used to define the attributes for an option type within the parameter value.

For a **grid**, the mnemonic is 'optiontype=GRID'. No other mnemonics are required.

For a **map**, the mnemonics are 'optiontype=MAP map=MAP_CD schemaPath=xpath', where 'MAP_CD' is the ID of the UI Map and 'xpath' is the schema path of the UI Map. In most use cases the schema elements will match the result set from the data explorer, in which case no schemaPath mnemonic is needed. If the UI Map schema has additional elements present, the schemaPath attribute will define the group node that the data explorer information will be copied into.

For a **chart**, the option type mnemonics is 'optiontype=CHART'. This is followed by the appropriate chart configuration attributes, which are described in the section below.

Chart Configuration

The following attributes are used to configure a graphical representation of a set of data.

The '**oraChart:**' attribute defines the type of graph to display and its general configuration. The set of configuration parameters available for this attribute are as follows:

Parameter	Values	Description
type:	pie	Defines the type of chart to display.
	stacked	Required
	cluster	
	line	
	area	
	combo	
showLegend:	true	Defines if the chart should have a legend displayed.
	false	Optional (default is true)
legendPosition:	left	Defines where the legend should appear.
	right	Optional (default is right)
	bottom	Setting position to left or right will automatically render it vertically.
	top	Setting position to top or bottom will automatically render it horizontally.
legendBorder:	true	Defines if the legend should display with a border around it.
	false	Optional (default is false)
depth:	true	A value of true indicates a 3 dimensional depth for the chart.
	false	Optional (default false , which is a 2 dimensional chart)
animate:	true	A value of true indicates that the graph should animate when displayed.
	false	Optional (default is true). When using large data sets, consider disabling animation.
dataCursor:	on	A value of on enables hovering anywhere in the graph.
	off	

Parameter	Values	Description
		Optional (default is off). It is not applicable to pie charts.
orientation:	horizontal	Defines the chart orientation. This only applies to bar, line, area, combo charts. E.g., oraChart="type:cluster; orientation:horizontal" , defines horizontal cluster chart. Optional (default is vertical).

The **oraChartSeries** attribute defines the source information for the graph. Most charts support 5 of these attributes, with the following mnemonics:

- **oraChartSeries1**
- **oraChartSeries2**
- **oraChartSeries3**
- **oraChartSeries4**
- **oraChartSeries5**

Stacked charts support an unlimited number of series by continuing to add attributes **oraChartSeries6** and above, but beware of performance implications and memory limits when using an excessively high number of series. All attributes are identical in format and accept the same parameters, as described below.

NOTE: All the charts require **oraChartSeries1**. Stacked, Cluster and Line charts may optionally include the additional series attributes (for multiple bars/lines).

If you define multiple series, then data must be provided for all series defined. The data amounts can be 0 (zero) but they have to be present in order for the chart to display correctly.

The set of configuration parameters available for the **oraChartSeriesN** attribute are:

Parameter	Values	Description
list:	XPath value	Defines the XPath to the list in the XML that contains the data to chart. Required only for HTML definitions in UI Maps. Data explorer values are derived from the columns for the zone.
amount:	Cn Pn XPath value	Defines the element containing the amount to chart. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified. Required
xaxis:	Cn Pn XPath value	Defines the element containing the x-axis data. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element is specified. Required for Stacked, Cluster, Line, Area and Combo charts.

Parameter	Values	Description
xaxisFormat:	date dateTime time localDate string	<p>Defines the x-axis data format.</p> <p>If it is date, dateTime or time then the value is presented in the format as defined on the user's display profile.</p> <p>In case of localDate or string the data is displayed as is with no special formatting.</p> <p>Optional (Default is date).</p>
label:	Text value	<p>Defines the label for the amount being charted.</p> <p>Either this setting or labelPath: must be defined.</p>
labelPath:	Cn Pn XPath value	<p>Defines the element that provides the label for the amount being charted. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified.</p> <p>Either this setting or label: must be defined.</p>
currency:	A valid Currency code	<p>Defines the currency code for the amount being charted.</p> <p>Optional.</p>
currencyPath:	Cn Pn XPath value	<p>Defines the element that provides the currency code for the amount being charted. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified.</p> <p>Optional.</p>
hoverText:	Text value	<p>Defines the hover text for the chart element.</p> <p>Optional (A default hover text is always available.) Ignored if hoverTextPath: is defined.</p> <p>The following substitution variables are available.</p> <ul style="list-style-type: none"> • \$label This will be replaced with the label text as determined by the label: or labelPath: setting. • \$amount This will be replaced with the amount text as specified by the amount: setting. • \$axis This will be replaced with the x-axis text. • \$% This will be replaced by the percentage "slice" of the pie or bar. • \$newline This will force a line break.

Parameter	Values	Description
		<p>If the hover text defined contains any of the above values, they will be replaced by the equivalent text prior to being displayed.</p> <p>Example:</p> <pre>"hoverText:\$label\$newline\$amount"</pre>
hoverTextPath:	Cn Pn XPath value	<p>Defines the element that provides the hover text for the chart element. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified.</p> <p>The hover text in the element can utilize all the substitution functionality described above in the hoverText: description.</p> <p>Optional.</p>
type:	bar line area	<p>This attribute is used for the combo chart type only. It defines how each series on the combo chart should be presented.</p> <p>The following example defines a combo chart where one series is rendered as bars and another one as area.</p> <pre>oraChart="type:combo;" oraChartSeries1="xaxis:C3; label:C2; amount:C1;type:bar" oraChartSeries2="xaxis:C3; label:C2; amount:C1;type:area"</pre>
navOpt:	A valid Navigation Option code.	<p>Defines a navigation option to be activated when the chart element is clicked.</p> <p>Optional.</p>
navOptPath:	Cn Pn XPath value	<p>Defines the element that provides the navigation option to be activated when the chart element is clicked. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified.</p> <p>Optional</p> <p>Note that both navOpt: and navOptPath: may be configured. The navOptPath: navigation option is processed first. If a value is found it is used, otherwise the value in the navOpt: setting is used. This means that the chart definition can include a default navigation option and a navigation option present in the data will override it.</p>
key:	Cn Pn XPath value	<p>Defines the element that contains the key field data to be used in a navigation option. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified.</p>

Parameter	Values	Description
		<p>This is required if either navOpt: or navOptPath: is defined.</p> <p>NOTE: Only one key field can be configured for a navigation option.</p>
script:	A BPA script name	<p>Defines a BPA script to be activated when the chart element is clicked.</p> <p>Optional</p> <p>When a script is executed, all the elements from the chart data will be made available to the script as temporary variables.</p>
scriptPath:	Cn Pn XPath value	<p>Defines the element that provides the BPA script to be activated when the chart element is clicked. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified.</p> <p>Optional</p> <p>Note that both script: and scriptPath: can be configured. The scriptPath: script option is processed first. If a value is found it is used, otherwise the value in the script: setting is used. This means the chart definition can include a default script and a script present in the data will override it.</p>
color:	HTML Color code / RGB value	<p>Optional (default colors applied)</p> <p>Defines the color for the series. The format is a valid HTML color code, e.g. green or blue. All valid color names are defined in this link: http://www.w3schools.com/html/html_colornames.asp.</p> <p>Alternatively an RGB format may be used. (00FF00 is green and 0000FF is blue)</p> <p>NOTE: Refer to Color Contrast for information about the use of the HTML color 'red' and its impact on accessibility.</p>
colorPath:	Cn Pn XPath value	<p>Defines the element containing a color for the series. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified.</p> <p>The valid formats as described in the color: setting apply.</p> <p>Optional (default colors applied)</p>
pieColors:	HTML color code / RGB values	<p>Defines the colors for the pie series. Any number of HTML color codes or RGB color</p>

Parameter	Values	Description
		values can be specified, separated by spaces. Examples: pieColors: green blue pieColors: 00FF00 0000FF Optional (default colors are applied if the series exceeds the values specified)

The **oraChartBroadcast** attribute provides the ability to broadcast a value from a chart to the portal context. An unlimited number of fields can be broadcast, configured as name/value pairs; for example: `oraChartBroadcast="FIELD_NAME1:C1;FIELD_NAME2:C2;"` where `FIELD_NAMEn` is the name of the portal context field to be broadcast and `Cn` (or `Pn`) is the element that contains the data value to be broadcast. Broadcast is triggered by clicking on a corresponding section of the chart.

For HTML definitions in UI Maps, the syntax is `oraChartBroadcast="FIELD_NAME1:xpath1;FIELD_NAME2:xpath2;"` where 'xpath_n' is the element in the list schema that contains the data value to be broadcast.

Zone Configuration Examples

Zone with a simple pie chart

In this example, the zone SQL summarizes To Do entries by Role. The column definitions are:

- C1 - role description
- C2 - count of To Do entries by role
- C3 - role ID

Visualization Option 1

This option shows To Do entries by Role as a pie chart. The role ID will be broadcast to the portal when the corresponding section of the pie chart is clicked.

```
optiontype=CHART
oraChart="type:pie;"
oraChartSeries1="labelPath:C1; amount:C2;hoverText:$label$newline$amount;"
oraChartBroadcast="ROLE_ID:C3"
```

Zone with two chart types

In this example, the zone SQL summarizes To Do entries by Priority and also calculates separate counts by To Do status within priority. The column definitions are:

- C1 - priority description
- C2 - count of To Do entries by priority
- C3 - priority flag value
- C4 - count of Open To Dos by priority
- C5 - count of Being Worked On To Dos by priority
- C6 - Label for the Open To Dos bar in the cluster chart
- C7 - Label for the Being Worked On To Dos bar in the cluster chart

Visualization Option 1

This option shows To Do entries by Priority as a pie chart. The priority value will be broadcast to the portal when the corresponding section of the pie chart is clicked.

```
optiontype=CHART
oraChart="type:pie; legendBorder:true;"
```

```
oraChartSeries1="labelPath:C1; amount:C2;hoverText:$label$newline$amount; "  
oraChartBroadcast="TD_PRIORITY_FLG:C3"
```

Visualization Option 2

This option shows To Do entries by Priority as a cluster chart with two data series, one for Open To Dos and another for Being Worked On To Dos . The corresponding priority value will be broadcast to the portal when clicking either the Open or Being Worked on bar of the cluster chart.

```
optiontype=CHART oraChart="type:cluster; showLegend:true;"  
oraChartSeries1="xaxis:C1; xaxisFormat:string; labelPath:C6; amount:C4;hoverText:$label$newline  
$amount; "  
oraChartSeries2="xaxis:C1; xaxisFormat:string; labelPath:C7; amount:C5;hoverText:$label$newline  
$amount; "  
oraChartBroadcast="TD_PRIORITY_FLG:C3"
```

Pagination Configuration

The various data explorer zones in the product support the ability to configure pagination so that a user can 'page through' a large set of results using "previous" and "next" buttons or links.

There are several zone parameters that are impacted when attempting to configure this functionality. The following steps highlight the configuration.

- The **Enable Pagination** parameter must be configured to define the basic setup for pagination for the zone. This parameter defines whether the "previous" and "next" actions are defined as buttons, links or icons and indicates the location of the actions. It also allows an indication as to whether the additional rows are simply appended, rather than shown in a new "page". Refer to the parameter's embedded help for information about the specific syntax.
- It is recommended that the zone is configured with record count and page information by properly configuring the **Record Count Display** parameter. Refer to the parameter's embedded help for information about the specific syntax.
- Configure the **Number of Rows to Retrieve for SQL** parameter to define the number of records displayed per page. If this parameter is not specified the value in the **Number of Rows to Display** parameter is used.
- Configure the key that will be used for paging so that the system can keep track of the 'page break'. The data must be sorted by the paging key; as a result, the decision for identifying the paging key must take into account the design for the zone and the data being displayed. In addition, the paging key must be unique to ensure that the page breaks occur correctly. See below for configuration examples.
 - The **SQL Statement** must include additional clauses **PAGENEXT** and **PAGEPREV** based on the paging key. In addition, as mentioned above, the paging key must be used in the ORDER BY clause.
 - The **SQL Column** parameters must define the paging key mnemonic to be used in conjunction with the SQL statement paging clauses.
 - It is recommended to configure the **SQL Display Column** parameter to show the data in the same order as the ORDER BY clause.

The following zone types support this capability:

- Info Data Explorer - Single SQL (**F1-DE-SINGLE**).
- Info Data Explorer - Multiple SQLs (**F1-DE**). Note that zones of this type support a union of the results of all the SQL statements. As a result, pagination may only be enabled for zones of this type if a single SQL is used. The system is not able to keep track of the pagination across disparate SQL statements.
- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**).
- Multi Query Data Explorer (**F1-DE-MULQRY**). Zones of this type do not include configuration for SQL statements or column display. However, they do include configuration for the **Enable Pagination**. This parameter must be configured in order for pagination on the individual zones to work.

NOTE:

Zones used for a Business Service. Note that pagination is ignored when invoking a data explorer zone via a business service. In this scenario, the zone will return the first "chunk" of rows as defined by the Number of Rows parameters.

Zones with data visualization options. Pagination is not supported for zones that define visualization options to show data in a graphical form.

Examples

Simple Paging Key

In this example, the Extendable Lookup Value is defined as Column 1 (C1) and is marked as the paging key. This field is unique for the table and works well as a simple paging key.

```
SELECT A.F1_EXT_LOOKUP_VALUE, A.BUS_OBJ_CD
FROM
  F1_EXT_LOOKUP_VAL A,
  F1_EXT_LOOKUP_VAL_L B
WHERE
A.BUS_OBJ_CD = :H1
AND A.BUS_OBJ_CD = B.BUS_OBJ_CD
AND A.F1_EXT_LOOKUP_VALUE = B.F1_EXT_LOOKUP_VALUE
AND B.LANGUAGE_CD = :LANGUAGE
[(F1) AND UPPER(A.F1_EXT_LOOKUP_VALUE) like UPPER(:F1)]
[(F2) AND ((UPPER(B.DESCR_OVRD) like UPPER(:F2))
OR (B.DESCR_OVRD = ' ' AND UPPER(B.DESCR) like UPPER(:F2)))]
[(PAGENEXT) AND A.F1_EXT_LOOKUP_VALUE > :C1]
[(PAGEPREV) AND A.F1_EXT_LOOKUP_VALUE < :C1]
ORDER BY A.F1_EXT_LOOKUP_VALUE
```

Complex Paging Key

Most queries however do not sort by a unique value. In this case, the paging key needs to be set based on the sorting of the query and should include a unique field, such as the primary key, as the last paging key. In this example, the query is showing results sorted by To Do Type, Role and User. All fields, including the To Do Entry ID (the primary key) are marked as paging keys.

```
SELECT TD_TYPE_CD, ROLE_ID, ASSIGNED_TO, ASSIGNED_DTTM, TD_PRIORITY_FLG, TD_ENTRY_ID
FROM CI_TD_ENTRY
WHERE
ENTRY_STATUS_FLG IN ('O', 'W')
[(F1) and TD_TYPE_CD = :F1]
[(F2) AND ASSIGNED_TO = :F2]
[(F3) AND ROLE_ID = :F3]
[(PAGENEXT) and ((TD_TYPE_CD>:C1) or (TD_TYPE_CD=:C1 and ROLE_ID>:C2) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2
and ASSIGNED_TO>:C3) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2 and ASSIGNED_TO=:C3 AND TD_ENTRY_ID>:C4))]
[(PAGEPREV) and ((TD_TYPE_CD<:C1) or (TD_TYPE_CD=:C1 and ROLE_ID<:C2) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2
and ASSIGNED_TO<:C3) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2 and ASSIGNED_TO=:C3 AND TD_ENTRY_ID<:C4))]
ORDER BY TD_TYPE_CD, ROLE_ID, ASSIGNED_TO, TD_ENTRY_ID
```

Use Data Explorer for Derived Data

There are times when a design warrants displaying data in a data explorer zone that is not accessible via SQL. For example, perhaps the data is from another system and it requires a web service call. The [JMS Message Browser](#) is another example.

The product provides functionality in the data explorer that allows you to call a script after the user filters are populated but before the SQL is executed. The script can retrieve the data as appropriate, store the data in table format so that the SQL can retrieve the data from the table.

The following points provide more detail:

- Create a service script that retrieves the data as needed. This script should store the retrieved data in a temporary table.
- The product provides a table that may be used. It is called F1_GENERIC_GTT (Generic Global Temporary Table). There is a business service — Create Global Temporary Table Records (**F1-InsertGTTRecords**) that the service script may call to insert the records.

- Note that if the data is accessed via a web service call, it may be appropriate to execute the web service in a separate session using the business service F1-ExecuteScriptInNewSession to trap errors that may be issued by the web service call and provide a better error.
- In the data explorer zone use this service script in the zone's pre-processing script parameter. If any user or hidden filters should be passed into the script, the parameter supports mnemonics for this purpose. Refer to the parameter's embedded help for the supported syntax.
- The SQL for the data explorer should access the temporary table that was populated by the service script.

Configuring Timeline Zones

This topic highlights information related to configuring a [timeline zone](#). The zone type is **F1-TIMELINE**. A timeline zone contains one or more "lines" where each line shows when significant events have occurred. The output of each line is driven by an algorithm configured on a timeline zone. Each algorithm is responsible for retrieving a single type of data. For example, one algorithm may retrieve bills for an account in a given time period whereas another algorithm may retrieve payments for that account for the same time period.

The algorithms to configure for a timeline zone use the **Zone - Timeline** plug-in spot. Please note the following details about the behavior for algorithms for this plug-in spot.

- The timeline algorithm receives all the global context values currently populated. In addition, it receives a start and end date from the zone, based on the time period chosen by the user, along with the maximum number of events that can be reasonably displayed for the chosen time period. The algorithm should use this information to retrieve data for a given type of transaction related to one or more of the input context values for the provided time period.
- For each event found, the algorithm returns information about the event along with many options that assist the user in getting more detail about each event or acting on an event.
 - Event date
 - Primary key of the record (key / value pairs)
 - FK Reference. With this information, the timeline zone will display the appropriate info string to display in the zone's info area when clicking on the event. In addition, the FK reference identifies the appropriate navigation option to use when a user clicks the info string hypertext to view the record on its maintenance page.
 - Background Color and Text Color to use for the event. (Obsolete). This information is no longer used by the timeline zone and will be ignored.
 - Icon use for the event. (Obsolete). This information is no longer used by the timeline zone and will be ignored.
 - [BPA script](#) to launch when a user clicks on an event. (Optional). The algorithm may return one or more BPA scripts that a user may launch to act on an event. For example, for an event that has a status of **Error**, perhaps a BPA is provided to walk a user through resolving the error.

When a script is initiated from a timeline, the system puts the prime key of the event into a field in the page data model. The name of the field is the column name(s) of the event's prime key. For example, when a script associated with a payment event is kicked off, the system populates a field called PAY_ID with the prime-key of the selected payment.

Note that your specific edge application may supply algorithm types for a timeline zone as part of the base product. Although algorithm types may be provided, typically the product does not deliver algorithms because the parameters for the algorithms are driven by a particular implementation's business rules and preferences. As a result, the product will also not deliver pre-configured timeline zones. Please refer to your edge application's documentation for more information about what timeline algorithm types are delivered, if any and recommendations for configuration.

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 portal is used to view and maintain the configuration of portals. An implementation may define their own portals or may override some of the settings for base product provided portals.

You can access the portal from the **Admin Menu > System > Portal**. You are brought to a query portal with options for searching for a specific record. Once a portal record has been selected you are brought to the maintenance portal to view and maintain the selected record.

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

Refer to [Understanding Portals and Zones](#) for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Portal.** This zone provides information about the selected portal definition.
- **Options.** This zone lists various configuration options for the selected portal record.
- **Tabs.** This zone lists other tab portals if any are configured for a standalone portal. This zone is only visible when such tabs exist for the current standalone portal.
- **Context Sensitive Zones.** This zone lists context sensitive zones that appear on the dashboard when this portal is displayed . This zone is only visible when such zones exist for the current standalone portal.

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.
Use the “Menu Links” dashboard zone to navigate to the “Menu Application Security” portal where you can review and maintain application security information for the current menu.

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**.

Use the **Menu Query** portal to search for an existing menu definition. Once a menu is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

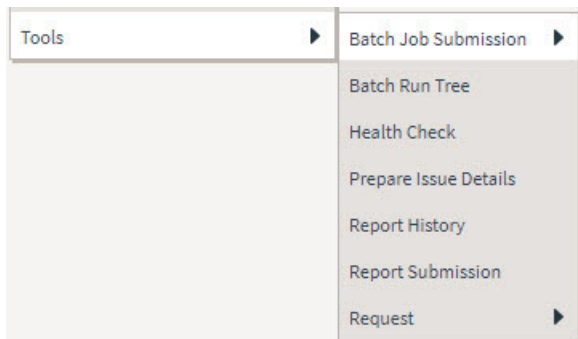
Menu - Menu Items

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

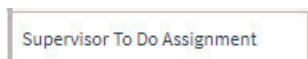
Each menu line can contain one or two menu items. The line's items control what happens when a user selects an option on the menu.

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.

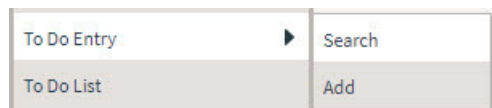
- The following is an example of a menu line with a single item that opens a submenu:



- 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 a carryover 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.

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 three types of navigation keys:

- **System navigation keys** define locations where the target for the navigation is a transaction or portal within the system. The navigation keys define the program component that identifies the page to navigate to.

- **External navigation keys** define a URL that identifies the target location. 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 application viewer links and URLs to external systems.
- **Help navigation keys** define a online help topic that identifies the specific page within the online help to launch. Help navigation keys may be related to a program component when the help is related to a specific page in system.

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 with no context included, while the second navigates to the page with context data provided to automatically display information related to that context.
- Please note that a wide variety of options can be defined on a navigation option. In addition to defining if data is passed to the page, it could also define search options. In addition, there are some navigation options that do not reference a navigation key but rather refer to a BPA script that should be launched.

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 Navigation Key Type of the navigation key being overridden is **External**, specify a Navigation Key Type of **Override (Other)** and define the appropriate URL Component.
- If the Navigation Key Type of navigation key being overridden is **System**, specify a Navigation Key Type 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.

Use the **Navigation Key Query** portal to search for an existing navigation key. Once a navigation key is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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

Navigation Key Type includes the following possible values:

- **External** indicates that the location is specified in the **URL Component** field.
- **Help** indicates that the navigation key is used to launch online help where the specific help topic is defined in the **URL Component** field.
- **Override (Other)** indicates that the navigation key overrides another navigation key of type **External** or **Help**. For this option, the name of the navigation key being overridden is populated in the **Overridden Navigation Key** field.
- **Override (System)** indicates that the navigation key overrides a system navigation key. For this option, the name of the navigation key being overridden is populated in the **Overridden Navigation Key** field.
- **System** indicates that the navigation key refers to a transaction in the system identified by its program component.

FASTPATH: Refer to [Navigation Key Types](#) for more information about navigation key types.

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 (Other)** or **Override (System)** is selected for the **Navigation Key Type**). Refer to [Overriding Navigation Keys](#) for more information.

URL Component is the specific URL or portion of the URL for the navigation key (external and help 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,resizable=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 navigation options that, in turn, reference the navigation key).

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.

Use the **Navigation Option Query** portal to search for an existing navigation option. Once a navigation option is selected, you are brought to the maintenance portal to view and maintain the selected record.

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 2nd section, **Alternate 2** is the 3rd 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.

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.

Understanding 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.

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, enter your desired verbiage in the corresponding override fields (and flush the [cache](#)).

Defining System Messages

This portal is used to search for message categories and 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.

You can access the portal from the **Admin Menu > System > Message**. You are brought to a query portal with options for searching for a specific message category or a message. Once a record has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to [Understanding System Messages](#) for more information.

Defining Message Categories

This portal is used to view and maintain a message category and its messages.

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 populate the Override Message Text. Note that even for message categories 80000 and higher, message numbers lower than 1000 are reserved for common base product messages.

You can access the portal from the **Admin Menu > System > Message**. You are brought to a query portal with options for searching for a specific message category or a specific message. Once a message category has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to [Understanding System Messages](#) for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Message Category.** This zone provides information about the selected message category.
- **Messages.** This zone lists and maintains messages for the selected category.

Defining Messages

This portal is used to view and maintain a specific message. An implementation may introduce messages used in custom processes or may choose to override the text for messages delivered by the product.

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 populate the Override Message Text. Note that even for message categories 80000 and higher, message numbers lower than 1000 are reserved for common base product messages.

You can access the portal from the **Admin Menu > System > Message**. You are brought to a query portal with options for searching for a specific message category or a specific message. Once a message has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to [Understanding System Messages](#) for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Message.** This zone provides information about the selected message.

Understanding Unified Search

Depending on your product, the toolbar search box may support a product specific search for business entities in addition to the standard search for menu items. For example, the search may support a query of business entities by their name, address, contact details etc. The unified search feature is a simplified version of your product's main search, that allows the user to lookup up records using free form search text or hint codes. For more complex queries, the user may navigate to the corresponding advanced search portal.

The configuration of a unified search option includes the following items:

- A **search service script** that implements the data query and defines the supported **filters**. The application service associated with this script is used to secure the unified search option.
- **Search filter rules** and **hints** for the supported filters. These rules are used to parse the text entered by the user to determine whether it qualifies as a valid filter value.
- The **advanced search** portal to use for more complex queries.

The unified search box interacts with a single unified search option implementation as follows:

- As the user types free form text in the search box, the application uses the **search filter rules** and **hints** to parse the entered text and translate it to corresponding query filters.
- The **search service script** is then called to query for entities that match these filters and present the results.
- Selecting a result record navigates to the corresponding portal.
- Selecting the **Advanced Search** link in the search box navigates to the general query portal associated with the unified search option.

If a unified search option is not enabled for your product or the user does not have security access to it then the search box may only be used for searching the menu.

The following sections describe various aspects related to the configuration of a unified search option.

The Search Script Controls Everything

A unified search is implemented as a single service script that controls the following:

- **Filters.** The script determines which filters are supported and how they are applied to return results efficiently. Each filter name must be a valid **Field** record.
- **Results.** The script returns a list of business entities that matches the query filters provided by the user, along with the navigation information needed to drill in to a selected record in the list.
- **Application security.** Only users that have access to the script's application service may use the unified search associated with the search script.

When the search box component initializes, the script is called once in **Prepare Search Details** mode to return the list of supported filter names. In this mode the script does not perform any query.

As the user types in text in the search box, the search box component analyzes the entered text to derive query filters and calls the search script in **Perform Search** mode to return matching results.

The search box component is designed to support any type of unified search implementation and as such, the interaction between the user interface component and the search script is generic. Filters are defined as name / value pairs and results for each matched entity includes details like primary, secondary and tertiary description lines, along with a line that describes the matching text for entry.

NOTE: If your organization needs to implement a custom unified search script, it is strongly recommended that you follow the base product's implementation of such scripts as a reference.

Filter Rules and Hints

A unified search is designed to support a predefined set of filters for querying data. For example, searching customers by their address, name, identifiers; or searching locations by address or name etc. Each filter is uniquely identified by a **Field** name as declared by the search service script.

As the user types in free text in the search box, the application applies parsing rules to determine whether the text may be used as the value of one or more filters supported by the unified search. For example, if the user types in the text "ave", parsing rules may determine that it can be used to filter results by a person's name or address but not their email address.

Each parsing rule, defined in the **Search Filter Rule**[extendable lookup](#), associates a filter's field name with a regular expression. Multiple filter rules defined for the same filter name are processed in execution order sequence and it is enough for one rule to return a match for the search text to be used as the value for the specific filter.

Alternatively, the user may use an explicit hint code to identify the filter they would like to use followed by the filter value. For example, an address filter may be associated with the hint "ad:" allowing the user to enter "ad: main" to look up records by address.

Each hint, defined in the **Search Filter Hint**[extendable lookup](#), associates a filter's field name with a hint code.

Note that filter rules and hints are defined for a specific filter name and not a specific unified search option. The same parsing rules may be shared across searches using the same filter names. For example the parsing rule to identify an "email" filter or the "ad:" hint for an address filter are the same regardless of how these are used by a search query. These rules are automatically associated with any unified search option that uses the same filter names.

NOTE: You may customize the parsing rules and hints associated with a filter by inactivating the base product's extendable lookup records and introducing custom rules that match your business requirements.

Which Unified Search Option is Used?

Each unified search option is defined in the **Unified Search**[extendable lookup](#), referencing the option's search service script and its corresponding advanced search portal. The unified search option is only enabled to users that have access to the search script's application service. When more than one unified search option is configured and enabled for a user, the application chooses the option with the highest sequence.

Customizing Unified Search

You may customize the parsing rules and hints associated with base product search filters by inactivating the base product's extendable lookup records and introducing custom rules that match your business requirements.

If you need to support a search for different business entities and filters, you need to implement a custom unified search. The configuration of a unified search option involves the following high level steps:

- Design and implement a search script that supports a predefined set of filters and returns the topmost set of matching results while taking into account performance considerations.
- Design hints and filter rules as needed for each of the supported filters. Review existing hints and filter rules associated with same filter names when applicable.
- Register each hint, if any in the **Search Filter Hint** extendable lookup.
- Register each filter rule, if any in the **Search Filter Rule** extendable lookup.
- Register the search script in the **Unified Search** extendable lookup.

Trees

Many edge products and implementations make extensive use of UI tree structures to display hierarchical relationships between records in the system. The framework provides an objects to define a tree structure and the rules on how to populate the nodes in the tree, and processes to interpret the configuration and dynamically render the tree display.

The topics in this section provide information about what is supplied in the framework. Your product may supply UI components with ready built trees for common use cases. Refer to your product documentation for more information.

Common Attributes of Trees

Your organization may have many portals and zones showing data in a hierarchical tree. Most trees in the system follow a similar pattern:

- Trees have a number of levels. There may be different types of tree nodes present in any given level; for instance a list of algorithm types for a script, followed by a list of script steps. Icons may be displayed to further identify the node's object type.
- The most common node types reference an entity within the system such as a schema, script, data area and so on. These nodes usually display the information string from the entity's FK reference and provide the ability to navigate to the entity. They may also allow the node context to be broadcast to other zones in the portal.
- In addition to the ability to navigate to the associated entity, many node types provide a context menu with further navigation paths.
- Some nodes are simply informational and are used to group a set of child nodes below a type or category. For example, the business object Options tree groups the option value nodes below summary nodes that describe the option type.

Each tree defines the nodes that make up the tree and the parent/child relationship between those nodes. A tree node defines which of the common attributes are applicable to the node and the algorithms responsible for retrieving the node data and, optionally, determining the node information to be displayed.

Populate Node Algorithm

A tree node displays a combination of the attributes listed above, typically for an object in context. For a root node, the object will normally be derived from a global or portal context value, such as the current script. For each lower level node, the context is derived from the context fields of its parent row; for instance, algorithms for a specific algorithm type.

The key responsibility of the populate node algorithm is to receive the tree code, node name and the parent context value (or values), determine if there is data that should be shown for this node, and return the a list of the context fields and values

applicable to each row in the node. The following highlights additional optional information that may be returned by the algorithm:

- Entity information. If the node is related to a specific entity, the maintenance object code and primary key values may be returned. These are used for determining default information for the node, including navigation details.
- A display icon value. This is used to override the icon defined on the tree node and/or the default icon for the entity as defined on the maintenance object or business object.
- An information string. This may be used to override the default information string for an entity or to display stand-alone text such as a summary or description of the child nodes to follow.

NOTE: The populate node algorithm is responsible for returning all the context values required to populate its child nodes. For an informational or summary node, this may require including the node's parent context fields in its own context. For example, in the script tree, the summary node describing the number of steps is followed by a list of steps for the script. The summary node populate node algorithm needs to include the script code passed in as parent context in its output context list, in order for the steps to be correctly retrieved.

Information Override Algorithm

The information override algorithm has two main purposes:

- Providing alternative information text for a tree node when the information provided isn't suitable for your implementation.
- Retrieving contextual insights related to the tree node entity, to be included in the node's information text. Only badge insight types are currently supported.

The override information algorithm receives the tree code, node name, maintenance object and prime key values for the node. The following highlights the information that may be returned by the algorithm:

- A display icon value.
- An information string. This will be used to override the default information string in the node's formatted string.
- Full node text. This will be used as the full text for the display, overriding the formatting defined on the node.
- A list of badge insight types and their context.

NOTE: If the information override algorithm returns insight types for the entity, it is also responsible for returning all the context values required for those insight types. Refer to [Contextual Insights](#) for more information.

Preparing a Node for Display

The following section highlights key steps in fetching a node's data and formatting it for display.

Determining Node Content

The following steps are performed to retrieve the components making up the tree node display:

- The Populate Node algorithm is called to determine all instance nodes for the tree node along with their related details.
- The Information Override algorithm is called for each node instance to allow for optional override of presentation related details.

The node display contents are determined as follows:

- The icon is determined from the following sources, in order: tree node definition, override information algorithm value, populate node algorithm value, maintenance object / business object default value.
- Broadcast usage is determined by the setting on the tree node definition.
- The information string is determined from the following sources, in order: override information algorithm value, populate node value, maintenance object / business object information algorithm. If the string is derived from the MO/BO algorithm, the short information text is used when available else the standard information is used.

- Context menu usage is determined by the settings on the tree node definition.
- The system supports a different style for header rows, including a larger font. Header style usage is determined by the setting on the tree definition and is only applicable if the node is a root node with only one instance.

Node Text Format

The text for nodes referencing a system entity commonly includes some combination of a label, the entity key and an information string. The node configuration provides a label field and a substitution format string field which may be used to define the node text format. The format string defines which of the three components should be included in the text, the order of those components and any surrounding text and HTML markups. For example, a format string of ‘%**label** - %**key** - %**info**’ will result in a node text like the example below:

▶  Algorithm Type - F1_PROSTINFO - Process Flow Information

If the format string is blank, the default text will include ‘%**info**’ only. If an override information algorithm is configured that returns the full node text, the text is presented without further formatting.

Auto Expansion

If the tree node definition indicates that the node should be expanded, the system will automatically fetch and display the data for its child node. This processing continues until there are no further child nodes marked for expansion.

Base Delivered Trees

Trees are system data and your product may already deliver several trees out of the box. You may choose to customize a base delivered tree in the following ways:

- You may add your own nodes to an existing tree definition.
- You may override the information string for a node by supplying your own **Information Override** algorithm. If a base override algorithm is configured on the node, your custom algorithm should be plugged in at a higher sequence number, to take precedence over the base algorithm.

Rendering a Tree

To render a tree on the UI, you would include the ou-tree web component in a UI map that references the tree definition.

NOTE: Refer to [Include a Web Component](#) for more information.

Defining Trees

This portal is used to view and maintain information about a tree definition. Trees are commonly used to display hierarchical relationships between objects in the system. Refer to [Trees](#) for background information about defining trees.

You can access the portal from **Admin > System > Tree**. You are brought to a query portal with options for searching for a specific tree definition. Once a tree has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones will appear as part of the portal's **Main** tab page

- **Tree.** This zone provides information about the selected tree.
- **Tree Nodes.** This zone lists the nodes for this tree in hierarchical order. Use the zone actions to add or maintain a node in the tree.
- **Tree Node.** This zone displays the details for a tree node. It is only visible if a tree node name has been broadcast from the Tree Nodes zone.

Contextual Insights

Contextual Insights provide a powerful way to surface focused intelligence and alerts to users in the context of the core screens for any applications built on Oracle Utilities Application Framework. There are multiple options for the visual patterns when surfacing Contextual Insights and the actions that can be performed by users are easily configurable as well. The subsequent topics provide detail about the Insight Type and Insight Group objects that are used to configure the information that may then be displayed in appropriate screens.

Insight Types

The insight type object contains all the detailed configuration for displaying one or more alerts / important information for a record in context. The following sections provide more information about all the configuration managed by the insight type.

Insight Algorithm

Each insight type is used to display some information, typically related to an object in context. It is the responsibility of the insight type algorithm to receive the context value (or values), determine if there is information that should be shown for this context, return the text to display.

The following highlights additional optional information that may be returned by the algorithm:

- A secondary message. This is only applicable to some visual structures where more than one line of information may be displayed.
- A severity value. This is used to override the visual display configuration if different emphasis should be used for different severity values. Refer to [Override Configuration](#) for more information.
- Entity information. If the insight is related to a specific entity, the maintenance object code and primary key values may be returned. This is useful if the insight type configures a BPA or user action algorithms that may perform logic for this entity.
- Navigation and User Action information. This is only applicable to visual structures that support user actions. The algorithm must return the list of all the actions that should be made visible on the insight. If the action includes navigation, the algorithm should return the navigation option and navigation context key and value pairs. Additionally, there is configuration for each action that must be defined. Refer to [User Actions](#) for more information.

Each algorithm can return multiple insights, if applicable. For example, if the insight is about To Do entries related to the object being displayed, you could design your insight algorithm to return a single insight summarizing all the To Do related information. Or you could choose to return separate insights based on To Do information so that you provide more granular information and customize actions or navigation based on the different conditions. For example, if there are To Do Entries that the current user may perform actions on and others that current user cannot update, the algorithm could return two separate insights: one with actions and one without.

NOTE: Preview mode. The insight type maintenance page includes an Insight Preview section that shows the output from the insight algorithm when invoked with an insight mode of "Preview". Every algorithm should include logic that returns appropriate text when called in this mode so that something is displayed in that section.

Visual Structures

Each insight type defines a visual structure which dictates what kind of information may be shown. The system has some fixed layout expectations for each visual structure type. The insight type configuration is used to provide the source for the possible layout.

NOTE: All the insights use color options as part of their configuration. Refer to [Color Option](#) for more information.

Badge Insight

The simplest visual structure is the badge. It is used to display a single line of quick text with a solid background (for visibility) that is meant to alert the user to important summary information or attributes.

3 Related To Dos

Inline Insight

The inline insight defines an icon and a single message. This is meant to be more descriptive than a badge. For this type of insight, a default light background is used so the recommendation is to select a color option for both the text and the icon that has only a foreground color defined or has a light background color.

 3 Related To Dos

List Insight

The list insight are intended to exist in a list of insights for the object. It defines an icon and can support two lines of text. In addition, this type of insight supports buttons or user actions. Refer to [User Actions](#) for more information. Note that for a list insight, if there is one valid action or 2 valid actions, the action buttons are shown. If there are three or more valid actions, one action is shown and then an 'Other Actions' button is visible. Clicking this shows the other available actions.

Note that the background color for the insight is used as the color for the full insight snippet. The text and the icon are shown on this background. The icon's background color is ignored.

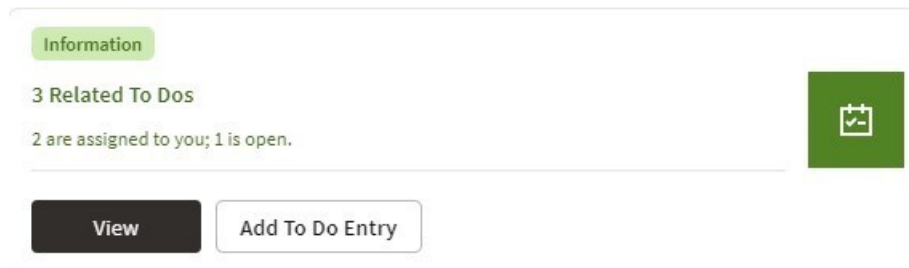


Card

The card insight is the most information rich, with up to two lines of text, [user actions](#), a large icon to call attention and header text. The header text is chosen from a dropdown list of valid options. The product provides out of the box header text of "Information", "Warning" and "Error". Implementations may add additional values to this using the lookup field **HEADER_CAT_FLG**.

Please note the following about the treatment of the various components of the card insight:

- The header text is rendered like a badge insight.
- For the message text displayed in the card, the background color is ignored so a color option with no background color or a light background color should be chosen.
- For the icon, the background color is used to create a square area for display and the icon is rendered in the foreground color.



Override Configuration

Each insight supports default visual setting for the various components of the insight. This includes the color option for the applicable components along with icons and header text, where applicable. In each case, override visual settings may be defined for a given severity number.

For example imagine the algorithm for a badge insight that returns a message with the count of related To Do entries uses a different color option if there are more than 10 related records. To do this, the algorithm can return a severity number and then the configuration user can define the override color option for this severity number.



20 Related To Dos

User Actions

The list and card insights support the definition of user actions that may be presented as buttons when displaying the insight. The actions may do one or more of the following:

- Perform algorithm logic. One or more user action algorithms may be associated with an action button. This should be used to perform server logic that does not require any user interaction. For example, you can provide an action to update the status of a record related to the insight. These algorithms are configured on the Insight Type.
- Navigate to another page. For this type of action, the insight algorithm must provide the navigation context and the navigation option to launch when clicking the button. It's possible to configure navigation and configure user action algorithms. In that case the algorithms are run first and then the navigation occurs.
- Launch a BPA. This is used if there is user interaction needed to perform additional logic and / or if there is conditional navigation such that the logic is needed before navigating. It's possible to configure a BPA and configure user action algorithms. In that case the algorithms are run first and then the BPA is launched. If the insight algorithm also provides navigation for this action, it is ignored in favor of the BPA. The BPA is configured on the Insight Type.

The list of valid actions to display at run time must be returned by the insight algorithm. This allows for the algorithm to have control over which actions to display, in case there are some actions that are conditional. In addition, each possible button requires configuration related to what text should display and the style of the button.

Button Text

The button text to display is chosen from a button text lookup. The lookup name is **INSIGHT_ACT_FLD_FLG**. Implementations may add additional values to this lookup to include new button text. In addition, if you would like to change button text on any product delivered button, you can override the lookup value description.

Action Class

Each action button defines an action class. Any action configured with the action class value of **Positive** value is shown first and is styled differently for emphasis. Other action class values are for information purposes.

Base Delivered Insight Types

Insight types are system data and your product may already deliver several insight types out of the box. You may choose to include one or more insight types in an [Insight Group](#).

Rendering an Insight

To render an insight for an insight type, you would include the ou-insights web component in a UI map that references this insight type. This is how the Insight Preview section of the insight type maintenance page renders the preview.

NOTE: Refer to [Include a Web Component](#) for more information.

Defining Insight Types

The insight type entity is used to define the configuration of a specific type of alert to display extra information related to one or more records in context. Refer to [Insight Types](#) for a detailed description of the supported functionality.

To view or maintain insight types, open **Admin > General > Insight Type**.

This is a standard [All-in-One portal](#).

The information captured on the insight type depends on the record's business object. Refer to the embedded help text for more information.

Related Insight Groups

If there are any insight groups that reference this insight type, the **Related Insight Groups** zone displays the linked insight groups.

Insight Groups

An insight group may be used to include several [insight types](#) together onto a single map. When defining an insight group, you configure the visual structures that the insight group should include. Only insight types for the configured visual structure are available.

Insight Class

Each insight group must reference a unique insight class. The insight class is defined using a lookup. The lookup field is **INSIGHT_CLASS_FLG**. Your product may already deliver base values for insight class. To display insights in a user interface your UI map would include the ou-insights web component. This component includes syntax to reference an insight class and the logic will find the insight group for that class and render all the insight types linked to that group.

Because the insight class is used as the reference in the web component rather than the insight group, it allows your product to supply base logic to show insight group information in a base delivered user interface. The base logic defines an insight class in base and references it on a base UI map. Your implementation can then define the insight group for this insight class and choose the insight types that should be rendered in that map. Refer to your specific product documentation to find out if there are any base delivered insight classes.

If your implementation identifies a new use case where one or more insight types should be displayed, once the insight types are configured, do the following:

- Define a new Insight Class lookup value.
- Define an Insight Group that references this insight class and configure the appropriate insight types.
- Include the appropriate [web component](#) in the map for your user interface where you wish to display this information.

Defining Insight Groups

The insight group entity allows implementations to define multiple related insight types that should be displayed together in a given user interface map or zone. Refer to [Insight Groups](#) for an overview of insight group configuration and functionality.

To view or maintain insight groups, open **Admin > General > Insight Group**.

This is a standard [All-in-One portal](#).

The information captured on the insight group depends on the record's business object. Refer to the embedded help text for more information.

Related Insight Types

The **Related Insight Types** zone displays all the insight types that are currently linked to the group. The Edit link in the zone header allows you to add or remove insight types. Only insight types that reference one of the **Valid Visual Structures** may be selected.

Color Configuration

The topics in this section describe some options for configuring colors to be used in displaying information in the user interface.

Defining Colors

The system provides the [extendable lookup](#) Color (**F1-Color**) to define valid colors to be used in the system. The base colors provided are ones related to the standard color palette used in Oracle products.

At this time, zones and UI maps that define specific colors do not support any syntax to reference a color extendable lookup. In the meantime, colors applied at the zone or map level can still reference HTML colors or reference hex codes directly.

Color Options

The color option entity is used to define a combination of colors to be used in user interface components. The system provides a standard set of color options used for contextual insights.

To view or maintain color options, open **Admin > System > Color Option**.

This is a standard [All-in-One portal](#).

The information captured on the color option depends on the record's business object. Refer to the embedded help text for more information.

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) or if it is a Scalable Vector Graphic (SVG) icon.
- **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.

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 custom style sheets into the system style set. Custom styles 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.

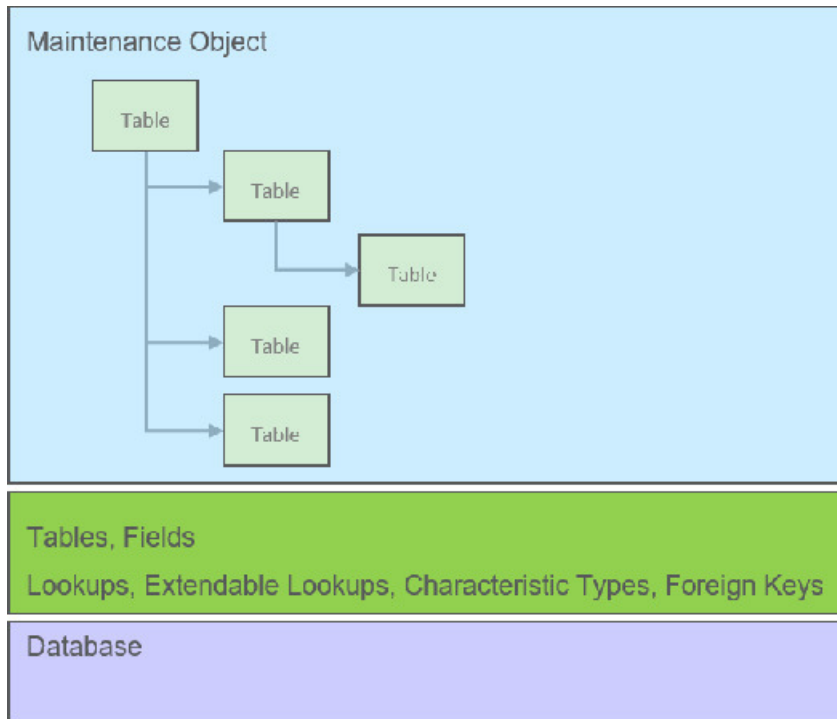
FASTPATH: Refer to the feature configuration for a detailed description of each option.

Database Options

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

Understanding The Data Model

The application uses meta-data to describe its data model, from plain description of its tables and fields to more complex hierarchies of maintenance objects.



The following points briefly outline key aspects of application's data model as illustrated in this diagram:

- Table and Field records describe all the tables maintained by the application.
- Valid values for certain fields are defined using corresponding Lookup and Extendable Lookup records.
- Characteristic types may be defined to support an extension and of a maintenance object via its characteristics table.
- Maintenance Objects define the physical structure of an entity. Business objects further refine maintenance objects to describe business entities. Refer to [The Big Picture of Business Objects](#) for more information.
- Foreign Key References define ways for searching and navigating to the maintenance portal of an entity.

The following sections further discusses concepts and guidelines related to the data model.

Fields

A Field definition describes the metadata needed by the application to support the capture and display of a piece of information. Fields may correspond to physical columns on database tables or represent any type of element used for calculation or presentation purposes.

The definition includes the following key details:

- Whether the field refers to a physical column on a database table or used as a work field.
- The type of data the field holds.
- The label to use when the field is presented.
- The help text to display when the user clicks on the help icon seeking additional information about this field.
- and more

The following sections further discusses concepts and guidelines related to defining fields.

Physical vs. Work Fields

A Field definition record exists for every physical column on a database table that is used by the application. Refer to [Understanding Tables](#) for more information.

A Field that does not represent a physical column on a database table is considered as work field and needs to be marked as such. Work fields are typically used as labels as well as describe internal elements used in business rules.

Data Type Information

The type of data captured by a field is defined using the following attributes:

- **Data Type** indicates the basic type of value the field holds. Valid values are **Character**, **Character Large Object**, **Date**, **DateTime**, **Number**, **Time**, **Varchar2** and **XML Type**.
- **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**, **Switch** and **URI**.
- **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.
- **Scale** is only applicable for number fields. It indicates the number of decimal places supported by the field.
- **Sign** is only applicable for numbers. It indicates if the data may contain positive or negative numbers.

Data Type Inheritance

A Field can be defined to inherit its data type attributes from the definition of another field. The field to inherit the attributes from is referenced as the **Base Field**. When using a base field reference, the data type attributes are defined once, on the base field, and all the fields inheriting their data type information from it are kept consistent with that definition as it may change over time. Any change of a field's data type attributes is automatically reflected on all the fields that reference it as their base field.

Data type information may either be specified directly on the field itself or by referencing a base field. The system does not support a way to partially override the base field's data type information.

It is recommended to leverage data type inheritance whenever possible, especially when the reason for creating a new field is to provide an alternate label for the original field in a certain context. In that situation the new field should reference the original field as its base field and provide the new labels and help text as needed.

Labels and Help Text

For fields that appear on a user interface, the description acts as a label to be used for presentation purposes. As per the product's multi-language support, a field is associated with a label for each of the languages enabled in the system.

A field's help text may be used to provide the user with additional information on how the field is used. When populated for a field that is visible on a user interface that supports display of embedded help, a help icon appears for the field, presenting the help text to the user when clicked.

Overriding Labels and Help Text

The label and help text provided for a product owned field may be overridden by your organization using the corresponding override fields.

The field's label and help text can be overridden for a specific table by populating the corresponding override information as part of the table / field definition. 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. For fixed pages, if the override label at the table / field level takes precedence over the override at the field level.

Alternate Labels

There are contexts where a separate label may be needed for the "date" and "time" parts of a date/time field. Such an example is when the field is used by the operational analytics product. For example, for a "Completion Date/Time" field you may provide the context label "Creation Date" for the **Date** context and "Creation Time" for the **Time** context. These context labels are optional.

In the same way, date fields, time fields and date/time fields may be associated with a **Date and Time Base** context label, which is a label without a reference to the word "Date", "Time" or "Date / Time". For example, for a "Completion Date/Time" field, the date and time base label may be "Completion". This label that can be used in combination with date or time attributes such as day, month, year, hours, minutes etc.

Overrides labels are available for these additional context labels.

Tables

A Table definition describes the metadata needed by the application to work with and access a database table or view.

The definition includes the following key details:

- The table's description in the application.
- The list of [fields](#) defined on the table and any foreign key constraints they establish with other tables.
- The table's prime key and how it is generated.
- Whether the table holds administrative, master or transactional data.
- Whether the table is released with product owned system data rows or not.
- The table's role in the conversion process.
- How should table content be upgraded.
- and more

The following sections further discusses concepts and guidelines related to defining tables.

System Tables

Tables released with rows that are owned by the base product are referred to as system tables. These tables include the **Owner** (OWNER_FLG) field to capture the owner of each row.

Refer to [System Data](#) for more information about naming conventions and standards around row record ownership.

The **Upgrade** setting on the table definition record controls what happens to the rows in a table when the system is upgraded to a new release. The most common setting allows customer owned rows to be merged with rows owned by the base product. Other settings may keep the content of the table as is, i.e. rows are not touched by the upgrade, or fully refresh the content of the table with base product rows.

Some fields on certain system owned tables are explicitly marked to allow customization. This means implementation are allowed to change these specific fields 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.

Designated Table Roles

Some tables are designed to support the following application roles:

- **Key Tables**
 - A key table holds a copy of a table's prime key only and is only applicable to tables with system generated keys.
 - Tables with a system generated keys reference their Key Table as part of their Table definition record.

- Records in the key table remain in the system even when the related entity is archived, preventing the same key from being allocated to a new record. Retaining the key allows the system to keep the integrity of any references that may still exist to the archived entity.
- **Language Tables**
 - To support multi-language implementations, all labels and descriptions that are presented to a user are kept in a designated language table where a row exists for each enabled language.
 - The product is released with the English content provided in the various language tables. An additional row is added to these tables for each enabled language along with the translated content for that language. Refer to [Support For Different Languages](#) for more information.
 - Tables associated with language sensitive fields reference their corresponding Language Table as part of their Table definition record.
- **Characteristics Tables**
 - Many entities in the system support one or more Characteristics tables, which are used to capture additional fields for the object that are not already supported by the entity's provided attributes.
 - A Characteristics table is designed to capture characteristics applicable to its specific entity only. The entity associated with a Characteristics table is referenced on the Table's definition record.
 - Refer to [Setting Up Characteristic Types & Values](#) for more information.
- **Entity Log Tables**
 - Many entities in the system support the ability to capture lifecycle and other important business events on a designated log table.
 - A log table captures the event details in the form of an application message and stores any parameters for the message in a child log parameters table. All log and log parameters tables follow a predefined structure allowing them to be consistently and efficiently implemented.
 - Typically log tables do not participate in the conversion process.

Table Constraints

Foreign key relationships between tables are not enforced at the database level but rather at the application level, providing the application more flexibility around data manipulation and referential integrity.

The following types of constraints may be defined on the Table's definition record:

- **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.

The system leverages a table's foreign key constraints information to performs referential integrity validation when rows in this table are deleted. These validations can be disabled by checking off the **Enable Referential Integrity** field on the Table definition record.

Primary Key Type

A table's primary key may be of the following types:

- **User-defined** means the user specifies the key when a record is added. This type of key is typically used for configuration and administrative tables.
- **System-generated** means a program generates a random numeric key for the record when it is added.

- This type of table is typically associated with a key table. Refer to the **Designated Table Roles** section for more information.
- System generated keys may be fully random or partially inherited from a parent table's key for performance reasons. Refer to the **System Generated Keys** for more information.
- **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.
- **Database Native Sequential Key** means a unique number is assigned using a database sequence object.
- **Other** means some other business rule allocates the table's prime-key or the prime key is made of multiple parts.

System Generated Keys

In general, randomly generated numeric system keys are used to attempt to evenly distribute records across a full range of possible IDs. Batch programs that use multiple threads will typically divide the threads using ID ranges and evenly distributed keys will help spread out the work across these threads.

If a table using a system generated key is a child of another table, the child table's key may **inherit** a portion of the random number from its parent table's key. Key inheritance for child records further extends the performance benefit. When considering partitioning, the recommendation for DBAs is to range partition data based on the primary key so that different batch threads operate on different partitions which reduces contention for hot blocks. Ideally the number of batch threads will be an exact multiple of the number of partitions. Batch programs that insert child data (for example batch Billing creation) also benefit from this design especially when the child tables are partitioned in the same way. The parent is often the driver of the batch process. If this is multi-threaded, then each thread is processing a set of parent records in a given ID range and all child records are being inserted into the same ID range.

While most entities with system generated keys either use fully random keys or partially inherited keys from a single parent, entities may be designed to implement more specific key generation rules.

When key inheritance is used, the table definition of the child table defines the following information:

- The number of most significant digits from the parent record's primary key to use as the **prefix** for the child record's key value.
- The **parent key field name** to inherit the key from. This information is provided for display and documentation purposes only and is only provided when the inheritance rule implemented by the entity may be documented this way. Typically the child key value is inherited from a single parent key but in some cases the inheritance rule may be more complex. For example, different types of records in the same child table may inherit their key from different parent key fields. In this case, multiple parent field names may be listed to describe the various inheritance options. When the rule is too complex, this information may not be provided on the table definition record.

Conversion Options

The conversion tool manages the process by which master and transaction legacy data, loaded into a staging schema, is validated and assigned new system generated keys before it is copied to the production schema. Refer to [Understanding The Conversion Process](#) for more information.

The tool classifies tables into the following roles:

- Administrative tables. These only exist in the production schema. Instead of a table, a view to the production table exists in the staging schema. These views are necessary to enforce foreign key references when data is validated in the staging schema
- Convertible tables that retain their prime keys. Data conversion is supported for these tables and their prime key values remain unchanged.
- Convertible tables that require a new system generated key to be assigned. Data conversion is supported for these tables and their prime key is reassigned by the system during conversion.

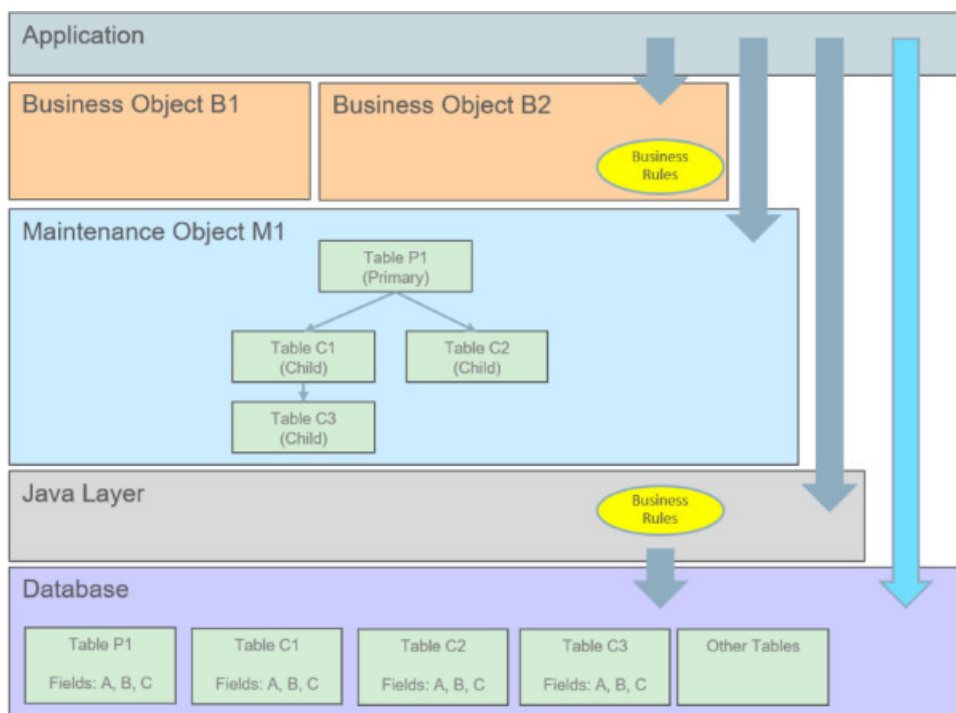
- Not convertible tables. Assigning new system generated keys for a record and replacing all foreign key references it might have with the new key is a complicated process that may not be supported for all tables.
- . The **Data Conversion Role** setting on the table definition record controls if / how the table is used by the conversion tool.

Date/Time Configuration

Fields capturing date/time information may be designed to store data in either local legal time or local standard time. Local legal time is captured as entered and displayed as captured in the database. It also assumes the date/time information refers to the time zone defined on the installation options. Refer to [Standard vs. Legal](#) for more information.

Maintenance Objects

A maintenance object defines the configuration of a given “entity” in the system. It includes a primary table and any additional child tables that together capture the physical data for the entity. The application reads, adds, updates and deletes an entity as a single object by invoking its maintenance object service.



The following points briefly outline key aspects of application’s interaction with maintenance objects as illustrated in this diagram:

- Database tables are maintained solely via the application’s Java layer. Direct updates to a table via its Java class is not common and typically reserved to special processes.
- A maintenance object orchestrates the update to an entity’s primary table and its child tables as a single object.
- A business object defines a logical representation of an entity, including all or a subset of the maintenance object’s tables and fields.
- The more common way to maintain an entity is via its maintenance object, either directly or via one of its business objects. Most entities support the use of business objects and as such their maintenance is typically made via their business objects.
- Direct access to database tables is supported for query purposes only.

The following sections further discusses concepts and guidelines related to defining tables.

Maintaining An Entire Object

A maintenance object includes a single primary table and one or more child tables where each child table can further have nested child tables. The table structure of a maintenance object represents the scope of data that should be maintained together as a single object:

- When an object is added, data is added to each corresponding table.
- When an object is updated, data is updated in each corresponding table.
- When an object is read, data is retrieved from all the tables.
- When an object is deleted, data is deleted from all the tables.

Algorithms

Several algorithm plug-in spots are also defined on the maintenance object, allowing for business rules that govern all records for this maintenance object.

Options

The maintenance object includes options that define important information related to the maintenance object that may be accessed for logic throughout the system.

Maintenance Objects vs. Business Objects

A maintenance object defines the physical structure of an entity as stored in the database. Different business objects may be defined on top of a maintenance object to represent different types of such entity. A business object includes only the relevant information applicable to the specific type of entity as well as enforces business rules applicable only to that type.

When a maintenance object supports business object configuration, the system uses the **Determine Business Object** maintenance object algorithm to determine the identifying business object that controls a given entity's data structure and rules.

Refer to [The Big Picture of Business Objects](#) for more information.

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.

Characteristic Types

Many maintenance objects in the system support characteristics tables, which are used to capture additional fields for the object that are not already supported by the object's provided attributes. Each characteristic is associated with a characteristic type, which defines attributes of the field you wish to capture.

While each characteristic is stored in a separate row in the characteristic table, the user interface metaphor used to capture and display them differ based on the type of page that is used to maintain the object.

- For portal based pages, the business object drives the display and maintenance of the object. The display and maintenance of each characteristic is determined by the business object's user interface design. For these types of pages, it is recommended that characteristics are defined as flattened elements allowing the user interface to display the characteristic as if it is another field. Refer to [The Flattening Nodes and Attributes](#) for more information about flattening characteristics on a business object schema.
- There are some fixed pages in the system that do not support customization of the user interface. For these objects, the characteristics are displayed / maintained as a generic list.

The topics in this section describe various aspects of a characteristic type definition and usage.

Characteristic Value Types

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 "skill", the information you setup on this characteristic type controls the valid values that may be specified by a user when defining another user's skill.

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. 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](#).
- **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.

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. 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.

Also note that the functionality described in the topics for [Referencing URIs](#) do not apply to this value given that the browser is responsible for connecting to the URI and does not go via server logic.

For references to a file, the recommendation is to use the Attachment functionality to link a file 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. To confirm if a given characteristic table supports searching, verify that it includes the field **SRCH_CHAR_VAL**.

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**.

Data Typing an Ad-hoc Characteristic Value

An **Ad hoc Value** characteristic type may further restrict the data type of the captured value by referencing a Field. When populated, the system validates that the captured characteristic value complies with the data type definition of the referenced field.

Valid Characteristic Entities

Characteristics are stored in designated tables, each associated with a unique characteristic entity value referenced on the table definition record. A characteristic type record lists all the valid characteristic entities it is allowed to be captured on.

Customizing Base Owned Characteristic Types

Implementations may customize the following aspects of a base owned characteristic type:

- Extend the list of valid values for a base owned **Predefined Value** characteristic type to include additional custom values. Such extension is only supported if the characteristic type is defined with the **Custom** option turned on.
- Provide a custom algorithm to validate an **Ad hoc Value** characteristic type.
- Indicate whether characteristics of this type may be used in searches.
- Extend the list of entities associated with the characteristic type.

Foreign Key References

A Foreign Key Reference defines the necessary information needed to reference an entity in a 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 [Characteristic Types](#) 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.
- 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.
- The name of the search page or zone used to look for a valid entity.

The topics in this section describe various aspects of a foreign key reference definition and usage.

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 information.

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 [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.

- 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.

NOTE: Not every FK reference provided with the product is configured with a search option. This may be the case if the base delivered pages use a dropdown for this foreign key rather than a search. In addition, base provided FK references that do provide a search may not be configured with both search options. It means that if linking this type of object as a characteristic, the search may not be available if the appropriate search is not configured.

Defining Tables

This portal is used to view and maintain information about an application table.

You can access the portal from the **Admin Menu > Database > Table**. You are brought to a query portal with options for searching for a specific table. Once a table has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to [Tables](#) for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Table.** This zone provides information about the selected table.
- **Fields.** This zone lists all fields of the selected table and provides various actions to maintain this information.

The following zones may appear as part of the portal's **Relationship** tab page

- **Entity Relationship Diagram.** This zone provides a visual representation of the table's relationship to other tables in the application's data model.

The following zones may appear as part of the portal's **Constraints** tab page

- **Table Constraints.** This zone lists the various key constraints defined for the selected table and provides various actions to maintain this information..
- **Referred By Constraints.** This zone lists all tables referring to the selected table via foreign key constraints.

Refer to [Tables](#) for more information.

Defining Fields

This portal is used to view and maintain information about a field. For product owned fields, you may use this portal to override the label of a field on the various pages in the system.

You can access the portal from the **Admin Menu > Database > Field**. You are brought to a query portal with options for searching for a specific field. Once a field has been selected you are brought to the maintenance portal to view and maintain the selected record.

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

Refer to [Fields](#) for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Field.** This zone provides information about the selected field.
- **Tables Using Field.** This zone lists tables referencing the field. It is not visible for work fields.

The following zone may appear as part of the portal's **References** tab page

- **Field References.** This zone list various configuration entities referencing the current field.

Defining Maintenance Objects

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 [Maintenance Objects](#) for more information.

Maintenance Object - Main

Navigate to **Admin > Database > Maintenance Object** to view information about a maintenance object.

Use the **Maintenance Object Query** portal to search for an existing maintenance object. Once a maintenance object is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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
Audit	Optional	Algorithms of this type are called to notify of any changes to the maintenance object's set of tables. These algorithms are invoked just before the commit at the end of a logical transaction. The system keeps track of what records are added or changed in the course of a transaction and all MO audit algorithms are

System Event	Optional / Required	Description
		executed in order of when each record was first added or updated.
Determine BO	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>
ILM Eligibility	Optional	<p>Algorithms of this type are used for maintenance objects that are enabled for object erasure for Information Lifecycle Management. 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>
Information	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>
Manage Erasure Schedule	Optional	<p>Algorithms of this type are used for maintenance objects that are enabled for object erasure, which is a measure to protect Data Privacy. They are triggered when certain system events occur for those objects. The algorithms are used to determine if the record needs to be scheduled for erasure and, if so, create or maintain an entry for the record in the Object Erasure Schedule.</p>
Revision Control	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 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>
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 State Transitions are Audited for more information.</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 state. The algorithm is responsible for logging the</p>

System Event	Optional / Required	Description
		<p>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>

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 Lookups

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.

Use the **Lookup Query** portal to search for an existing lookup. Once a lookup is selected, you are brought to the maintenance portal to view and maintain the selected record.

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 > General > 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.

Extendable Lookup Advanced Topics

This section provides some addition technical information about extendable lookup attributes

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

Option	Brief Description	Extendable Lookup Value Searchable by this Attribute?	Base Delivered Value Modifiable?
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 value is known when the attribute is needed (in which case a CLOB mapping is appropriate) or if the functionality expects to determine the lookup value 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.

Capturing a Password

If an extendable lookup includes configuration of a password for some functionality, the system supports automatic encryption of the password value if the schema maps the password to a characteristic using the characteristic type **F1-PWD**.

Defining Characteristic Types

This portal is used to view and maintain characteristic types. An implementation may define their own characteristic types or override some of the settings for base product provided characteristic types.

You can access the portal from the **Admin Menu > General > Characteristic Type**. You are brought to a query portal with options for searching for a specific record. Once a characteristic type record has been selected you are brought to the maintenance portal to view and maintain the selected record.

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

Refer to [Characteristic Types](#) for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Characteristic Type.** This zone provides information about the selected characteristic type.

Defining Foreign Key References

The topics in this section describe the transaction that allows you to define foreign key references.

Refer to [Foreign Key References](#) for more information.

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

Use the **Foreign Key Reference Query** portal to search for an existing foreign key reference. Once a foreign key reference is selected, you are brought to the maintenance portal to view and maintain the selected record.

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 record.

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 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 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 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.

Audit Trails

The topics in this section describe one way of auditing changes in the system. Note that this technique has limitations and may not be the best option for all situations.

- This functionality is configured at the table and field level. Many base maintenance objects use an XML or CLOB field to capture one or more elements using XML format, configured using a BO schema. This auditing technique is not able to capture changes to individual elements. It can only capture overall changes to the single field. The BO Audit plug-in is a better option for auditing changes to individual elements with the CLOB / XML field.
- The base table provided for supporting audits limits the field size of the before and after values to 254 bytes.
- Auditing is captured for each field. In some cases it is preferred to capture a before and after image for several fields at once. For example, if an address is changed, it's more user friendly to capture the before and after for the full address rather than the individual address components. The BO Audit plug-in allows for capturing a single audit record for multiple elements rather than granular changes for each element.

The subsequent topics highlight how to enable auditing for fields, and describe the 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. If any of the changed fields are marked for audit, the audit program inserts audit rows into the audit table.

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 the base audit program and base audit tables 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 base audit insert program 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 base delivered 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.). An appropriate java table name must be defined and an *_impl.java program for that java table name must be defined.
- The name of the new table must be referenced on the various tables whose changes should be logged in the new table.

NOTE: 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 the base delivered audit program 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 **CI_NAV_OPT**, etc. If you cannot find the table using the search facility on the [Table](#) portal, please contact customer support.

To enable auditing for a table:

- Navigate to the [Table](#) portal 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** . The product provides two programs.
com.splwg.base.domain.common.audit.DefaultTableAuditor is a standard table audit program. In addition **com.splwg.base.domain.common.audit.ModifiedTableAuditor** is supplied as an alternative. This program does not audit inserts or deletes of empty string data.
-

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](#) portal 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: You can also turn on the audit switches using the Field zone at the bottom of the [Table](#) portal.

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 > Database > 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.

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 > Database > 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.

Note that use of the bundling add BO also benefits records that have optional foreign keys or foreign keys that are part of a child table. That way the person creating the bundle does not have to worry about the sequence of the records. For example, an FK Reference may optionally reference a zone for searching. If a new FK Reference and its search zone are bundled together to copy to another region, the bundling add BO for FK Reference (which doesn't include the Zone) ensures that the FK reference could be added before the zone without getting any validation errors.

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:

Configuration Task	Scope of Task
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	Only needed if there are objects with recursive or optional 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. Go to the [Maintenance Object](#) page and search for the maintenance object. On the Options tab, add a new option with the type **Eligible For Bundling**. Set the value to "Y" and click **Save**.

Adding a Foreign Key Reference

Foreign key references are used for many purposes and most maintenance objects should have them anyway. Check the Maintenance Object options to see if there is an entry for the **Foreign Key Reference** option. If one is not configured, navigate to [FK Reference](#) and set up a foreign key reference for the primary table of the maintenance object. Configure that new FK reference on the maintenance object using an option.

Creating a Physical Business Object

Each maintenance object in a bundle must have a physical business object. Refer to [Creating a Physical Business Object](#) for more information.

Creating a Bundling Add Business Object

If the objects to be bundled have recursive foreign key references, optional foreign key references or child tables that include foreign key references, create a bundling add business object to avoid problems with referential integrity.

1. Navigate to [Business Object](#) 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 child tables and all elements that are not required.
4. Save the business object.
5. Navigate to [Maintenance Object](#) 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.

1. Navigate to [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.

Create an entity-specific query zone to search for the maintenance object:

1. Navigate to [Zone](#) and search for one of the base query zones, such as the Algorithm Search zone F1-BNALGS.
2. Click the **Duplicate** button in the record actions section.
3. Enter a name and description for the new zone.
4. Click **Save**.
5. Customize the details of the parameters for your maintenance object.

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, skip this step.

1. Navigate to [Zone](#) and search for the F1-BNDLENTQ Entity Search zone.

2. Duplicate this zone.
3. Remove any references to base query zones.

Add the new entity-specific query zone to the customized entity search zone:

1. Locate the customized entity search zone for your Bundle Export portal. This is the zone created in the previous step.
2. Add the name of the query zone you created in the first step to the list of Query Zones.

Add the customized entity search zone to the Bundle Export portal. This step needs to be done only once.

1. Navigate to [Portal](#) and search for the Bundle Export portal, F1BNDLEM.
2. In the zone list, add the entity search zone you created above. (Add the new zone after the base entity search zone).
3. 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 > Implementation Tools > 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 > Implementation Tools > 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 > Implementation Tools > 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) Navigate to [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) Go to the [Maintenance Object](#) page 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. Refer to [Creating a Physical Business Object](#) for more information.
2. Add a foreign key reference to the maintenance object so that the standard information string is displayed. Foreign key references are used for many purposes and most maintenance objects should have them anyway. Check the Maintenance Object options to see if there is an entry for the **Foreign Key Reference** option. If one is not configured, navigate to [FK Reference](#) and set up a foreign key reference for the primary table of the maintenance object. Configure that new FK reference on the maintenance object using an option.
3. Add the **Revision Control** zone to the maintenance object.
 - a) Navigate to [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) Go to the [Maintenance Object](#) page 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 > Implementation Tools > 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** 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.

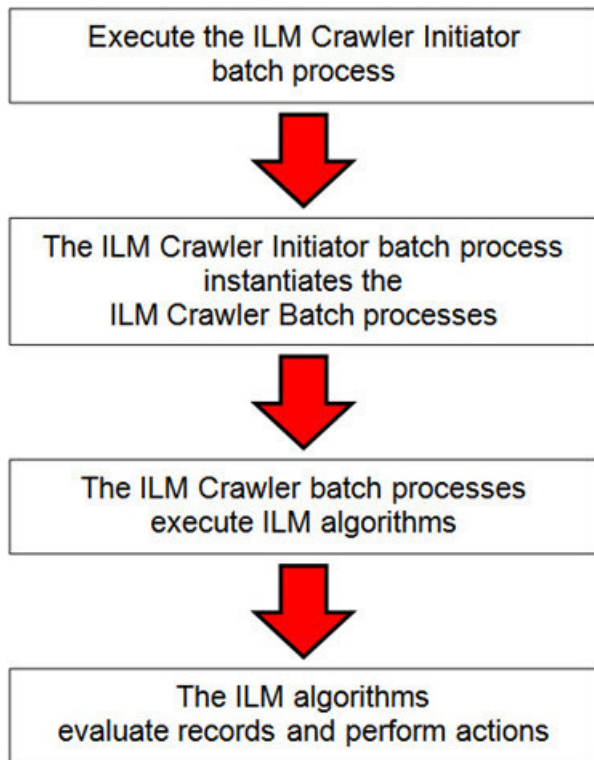
Batch Processes

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.

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 embedded 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 chooses 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"/>

```

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: Refer to [Schema Syntax](#) for the 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 [Schema Syntax](#) 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: Refer to [UI Hint Syntax](#) for detailed information about the supported 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 optionBO 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. 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.
- 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.

- 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](#). An example of this is the 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 Content 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.

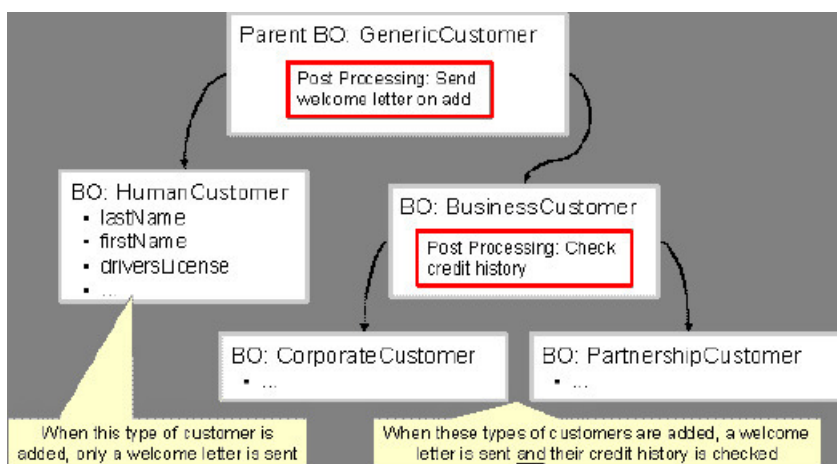
For all maintenance objects, the base product may provide additional BOs that are not meant to be "identifying" BOs, but instead are provided to support functionality to interact with the MO using the BO as a tool as described in [Invoking a BO](#).

- 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](#), [revision control](#) and the pre-compare algorithm for CMA to [adjust data prior to comparing](#). If there are any maintenance objects that do not have a base supplied physical BO, refer to [Creating a Physical Business Object](#) for steps on how to provide one.
- A "bundling add" BO may be supplied. Refer to [Recursive Key References](#) for more information as to why this type of BO may be supplied.

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.

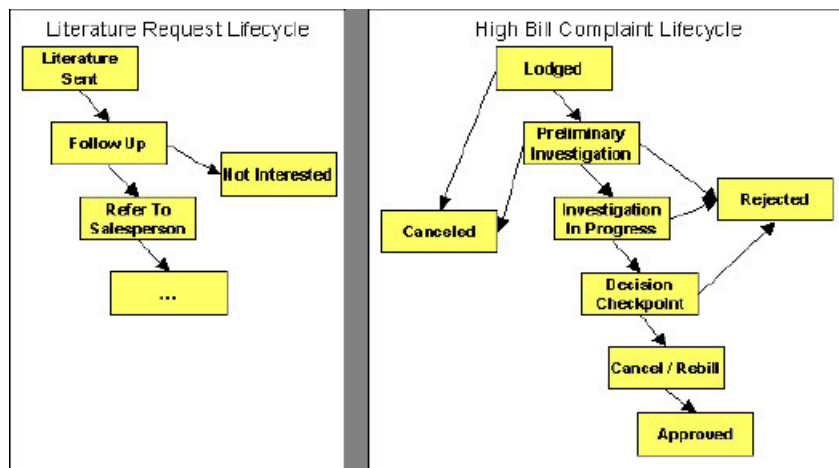
NOTE: User interface maps are inherited. When determining if the business object has a UI map to use for UI rendering, the system looks for display and maintenance maps linked to the BO as options. If the identifying BO does not have maps defined, the system follows "up the chain" of inheritance until 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.

Also note that when a record is processed by the monitor batch program, by default the BO Post Processing, BO Audit and MO Audit algorithms are not executed. However, it is possible for an enter algorithm to indicate that the other algorithms should be executed by the batch process by setting the "force post processing" indicator to true.

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.

Also note that when a record is processed by the monitor batch program, by default the BO Post Processing, BO Audit and MO Audit algorithms are not executed. However, it is possible for an exit algorithm to indicate that the other algorithms should be executed by the batch process by setting the "force post processing" indicator to true.

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.

Also note that when a record is processed by the monitor batch program, by default the BO Post Processing, BO Audit and MO Audit algorithms are not executed. However, it is possible for a monitor algorithm to indicate that the other algorithms should be executed by the batch process by setting the “force post processing” indicator to true.

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.

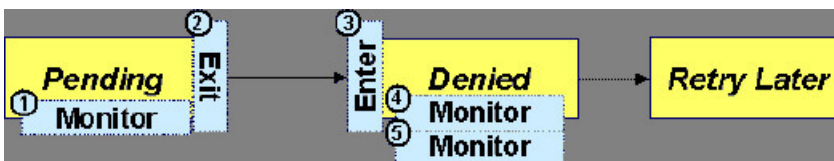
The system executes all state-specific algorithms at all levels in the inheritance chain starting from the highest-level parent business object moving on to lower levels.

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 Error Maintenance 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

A monitor batch process may be used to transition a business object into its next state by executing the monitor algorithms associated with the current state of the entity. The use cases for performing the monitor logic in batch are as follows:

- The record may be waiting for something else to occur before transitioning. The monitor algorithm may be coded to determine if the condition is satisfied and initiate the transition then. For example perhaps when entering a state, a field activity is generated and the record should exit the state when the field activity is complete. The monitor algorithm can check the status of the field activity.
- Perhaps a record is added or updated manually and the next step in the BO lifecycle includes a large amount of processing such that the logic should occur in batch. In this case the BO status is configured with an explicit reference to a batch control (referred to as "deferred"), which indicates to the system that the monitor algorithms should not be performed automatically (but should be deferred to batch). Later when the batch process runs, it selects all the records to process to progress the records.

NOTE: When a status includes a deferred batch control, it may also be configured to allow a user to manually transition the record to the next state rather than waiting for batch. When a user manually transitions a record that includes monitor algorithms, those algorithms are not executed.

- Perhaps a record is added or updated in batch, but a subsequent step in the overall lifecycle should be processed later. This may be accomplished by ensuring that the batch control linked to the state to process later does not match the batch control that added or updated the record.
- Monitor processes may also be used to periodically perform some logic related to the record without actually transitioning the record.

Note that only the parent business object may refer to a deferred monitor batch process. However, any business object in the “[inheritance](#)” chain may be configured with monitor algorithms, which will all be executed.

The base package provides a periodic monitoring batch process for each maintenance object that supports a configurable BO lifecycle. The process periodically executes the monitoring algorithms associated with the current state of an entity, excluding states explicitly referencing a deferred monitoring batch process that is for a different batch control.

A deferred monitoring process selects records whose current state references this particular batch control as their monitor process. Deferred monitor process is only needed when an object has different use cases for monitoring the same type of records with different schedules. In this case only one periodic monitor batch should be configured. Other monitors should be configured to restrict by batch control so that there is no overlap in processing.

NOTE: MO option configuration. The maintenance object includes options to indicate the batch controls delivered for periodic and deferred monitor batch controls.

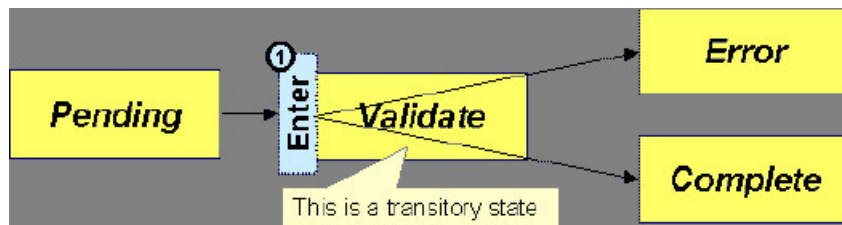
Your business rules will dictate the execution frequency of each monitoring process and the order in which they should be scheduled. Refer to [Monitor Background Processes](#) in the background process chapter for more information about the parameters supported for this type of batch process.

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 by default the [business rules](#) plugged in to the business object are not executed. However, it is possible for an Enter algorithm, Exit algorithm or Monitor algorithm to indicate that the other algorithms should be executed by the batch process. If the “force post processing” indicator is set to true, then the batch process invokes the BO Post Processing, BO Audit and MO Audit algorithms.

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.

Opening UI Maps Before Entering A State

You can define a UI Map to capture additional elements before a business entity can enter a given state. You do this by configuring the **State Transition UI Map** [state-specific option](#) on the appropriate state on the business object.

There may be circumstances in which the status pre-processing logic can determine the value of the elements, in which case it is not necessary to invoke the map. The system provides a data area (**F1-StateTransitionCommon**) which contains a boolean element to indicate if the state transition map can be skipped. This data area can be included in your pre-processing script to allow the element to be set according to your business logic. The base maintenance script (**F1-MainProc**) references this element to determine whether to open the state transition map.

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 determine the identifying BO .
BO Validation algorithms executed.	
State transition rules are performed if the status has changed.	<p>BO Status Exit algorithms for the "old" status executed.</p> <p>Status updated to the new value.</p> <p>BO Status Enter algorithms for the "new" status executed.</p> <p>If no error — MO Transition algorithms are executed.</p> <p>FASTPATH: Refer to State Transitions are Audited for more information.</p> <p>If error and there are "save points" the MO Transition Error algorithms are executed.</p> <p>FASTPATH: Refer to Keeping An Entity In Its Last Successful State for more information.</p> <p>Otherwise, the error is reported.</p>
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. The [Monitor Batch Process](#) does not invoke the maintenance service. By default 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). However, it is possible for an Enter algorithm, Exit algorithm or Monitor algorithm to indicate that the other algorithms should be executed by the batch process. If the "force post processing" indicator is set to true, then the batch process invokes the BO Post Processing, BO Audit and MO Audit algorithms.

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 launch help topics related to valid schema syntax and UI Hint syntax in one click.
- **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**

Use the **Business Object Query** portal to search for an existing business object. Once a business object is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

The options grid allows you to configure the business 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 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](#).

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.

FASTPATH: Refer to [Schema Syntax](#) and [UI Hint syntax](#) for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **Schema Tips** zone in the dashboard provides links to launch these help topics directly.

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 that each maintenance object has an underlying service and it's possible that some of the child tables in the maintenance object are not included in the service definition. If that is the case, a message is issued and the generated schema would need to be adjusted to remove that child table.

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.

System Event	Optional / Required	Description
Audit	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>

System Event	Optional / Required	Description
		The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.
Information	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 Business Object inheritance for more information.</p>
Post-Processing	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 Business Object inheritance for more information.</p>
Pre-Processing	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 Business Object inheritance for more information.</p>
Validation	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 Business Object inheritance for more information.</p>

FASTPATH: Refer to [BO Algorithm Execution Summary](#) for more information about how these algorithms fit within the business object processing.

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** 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 Business Object Inheritance for more information.</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 Business Object Inheritance for more information.</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 Business Object Inheritance for more information.</p>

FASTPATH: Refer to [BO Algorithm Execution Summary](#) for more information about how these algorithms fit within other business object algorithms.

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 embedded 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.

Advanced BO Tips and Techniques

The topics in this section describe some advanced tips and techniques for configuring business objects.

Creating 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. There are several processes in the system that may use a maintenance object's physical BO.

To create a physical business object for the maintenance object:

1. Navigate to [Business Object](#) 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. Navigate to [Maintenance Object](#) 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.

Managing To Do Entries

The product provides several base algorithm types that may be used to manage To Do entries through status changes for a given record via BO lifecycle plug-ins.

Create To Do Entry

The product supplies a BO status Enter algorithm type **Generic To Do Creation (F1-TDCREATE)** that creates a To Do entry based on parameter configuration. Refer to the algorithm type description for more information about how it determines the To Do type or To Do role and how to populate the appropriate message text onto the To Do. This algorithm may be used in conjunction with the Retry Logic (below).

If your implementation has a business rule that requires a To Do entry to be created when entering a given BO status and the logic provided by the algorithm type meets the needs of the business rule, this algorithm type may be used. Create an algorithm for the algorithm type, populate the algorithm parameters according to the business rules and plug the new algorithm into the appropriate business object status as an Enter algorithm.

Retry Logic

The algorithm type **Retry for To Dos (F1-TODORETRY)** is supplied for a special use case. It is a BO status monitor plug-in and may be used for a state that is a type of 'error' or 'waiting' state. It relies on the To Do entry creation logic to set a Retry Frequency. The algorithm transitions to the originating state to retry the logic. The idea is that the condition that caused the record to enter the 'error' or 'waiting' state may be resolved after some period of time has passed, allowing the record to progress in its lifecycle. Refer to the algorithm type description for more information about its logic.

To use this functionality, create an algorithm for this algorithm type, populate the algorithm parameters according to the business rules and plug the new algorithm into the appropriate business object status as a Monitor algorithm. The state should also have an algorithm for the **Generic To Do Creation** algorithm type plugged in as an Enter algorithm (or something equivalent) that sets the appropriate Retry Frequency.

To Do Completion

It is common that one or more To Do entries associate with a given record should be completed when exiting a state (if it is not already completed). The system supplies the algorithm type **Generic To Do Completion (F1-TODOCOMPL)** that may be used for this purpose. Note that the algorithm type functionality is not tied to any To Do creation logic. It may be used for any use case where To Do entries should be completed on exiting a state. Refer to the algorithm type description for more information about its functionality and how to prevent certain To Do entries from being automatically completed.

To use this functionality, create an algorithm for this algorithm type, populate the algorithm parameters according to the business rules and plug the new algorithm into the appropriate business object status as an Exit algorithm.

Submitting a Batch Job

The product provides a base algorithm type that submits a batch job when entering a BO state. This functionality allows for “event driven” batch submission where the event is the lifecycle transition for a certain record.

The algorithm type is **Create Batch Job Submission Entry for Batch Control (F1-SCHEDJOB)**. The batch control code is a parameter for the algorithm. Refer to the algorithm type description for more information about its logic.

To use this functionality, create an algorithm for this algorithm type, populate the algorithm parameter with the batch control that should be submitted and plug the new algorithm into the appropriate business object status as an Enter algorithm.

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.

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 Tips. The data area page includes a special Schema Tips zone that provides a link to launch help topics related to the [Advanced Schema Topics](#) help in one click.

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**.

Use the **Data Area Query** portal to search for an existing data area. Once a data area is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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.

FASTPATH: Refer to [Schema Syntax](#) and [UI Hint syntax](#) for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **Schema Tips** zone in the dashboard provides links to launch these help topics directly.

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.

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.

FASTPATH: Refer to [Schema Syntax](#) for 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.

Application Service is the application service that is used to provide [security](#) for the service. If the service is related to a maintenance object, the access modes for the application service should be the standard **Add, Change, Delete** and **Inquire**. For other services, the application service should have the **Execute** access mode.

Service Type indicates whether the service is a **Java Based Service** or a **Java (Converted) Service**. Note that services generated to support a portal in the system will not have a service type populated.

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.

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 launch help topics related to 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**

Use the **Business Service Query** portal to search for an existing business service. Once a business service is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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.

FASTPATH: Refer to [Schema Syntax](#) for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **Schema Tips** zone in the dashboard provides a link to launch this help topic directly.

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.

Useful Services and Business Services

The following section highlights some business services and services provided by the product that may be useful for implementations to use.

Data Explorer Service

The system provides a mechanism for performing an SQL select statement for use in scripting, Java plug-ins, or via a web service call. This is done by creating a zone using one of the data explorer zone types where the SQL is defined. Then, create a business service using the Data Explorer service (**FWLZDEXP**).

NOTE: There are numerous business services delivered with the base product that reference this service that may be used as a template.

The following points highlight how to create your own business service for this service. Note that typically a separate business service exists for each zone.

- Enter a **Business Service** code and a **Description**. It is recommended to define the business service code to match the zone code so that it's easier to manage which business service invokes which zone.
- Select the **Service NameFWLZDEXP**.
- On the **Schema** tab, under the `<schema>` node, enter mapping for the fields that are required for the Data Explorer service:
 - The **Zone** should be mapped into service field **ZONE_CD** . Define the zone code as the default value.
 - For every **user filter** defined on the zone, create a schema mapping into the service field **Fx_VALUE**, where "x" is the filter number (from the zone parameters).
 - For every **hidden filter** defined on the zone, create a mapping into the service field **Hx_VALUE**, where "x" is the filter number (from the zone parameters).
 - The search results are returned as a list by the data explorer service. Each column value is in the service field **COL_VALUE** with an appropriate sequence number (**SEQNO**). The results can be [flattened](#) based on sequence number allowing for a logical element name to be defined.
 - Another useful field is **ROW_CNT**, which provides the number of rows retrieved by your search.

The following is an example of the schema for a BS that receives a business object code and returns a list of status values and their descriptions that allow status reasons to be defined.

```

<schema>
  <zone mapField="ZONE_CD" default="F1-BOSTSLST"/>
  <bo mapField="H1_VALUE" />>
  <rowCount mapField="ROW_CNT"/>>
  <results type="list" mapList="DE">>
    <status dataType="string" mapField="COL_VALUE">
      <row mapList="DE_VAL">>
        <SEQNO is="1" />>
      </row>>
    </status>>
    <description dataType="string" mapField="COL_VALUE">
      <row mapList="DE_VAL">>
        <SEQNO is="2" />>
      </row>>
    </description>>
  </results>>
</schema>

```

Maintenance Object Log Service

Many maintenance objects support a log table that follows a pattern of column names and behavior. The system provides a service called Generic MO Log Service (**F1MOLOGP**) that may be used to perform common functions related to log entries:

- Read log entries. If you pass a certain MO, primary key and log sequence number, the service will return the details of that log entry. The product provides a generic business service that may be used for this purpose — Generic MO - Retrieve Log Details (**F1-ReadMOLog**). Alternatively, it is possible to create a business service for a given MO where the MO code is assigned to the MO element using the default syntax. This allows business functionality specific to that maintenance object to use the specific BS.
- Add log. The service may be used to add a log entry. If a user log is added, then the comments from the user are populated in the detailed description. System generated log entries typically supply a message category / message number along with other information such as the status, a specific log type and optionally a related object reference (via a characteristic). The product provides a generic business service that may be used for this purpose — Add Generic MO Log (**F1-AddMOLog**). Alternatively, it is possible to create a business service for a given MO where the MO code is assigned to the MO element using the default syntax. This allows business functionality specific to that maintenance object to use the specific BS.

Base Business Services

The following table highlights some business services provided by the product that may be useful for custom logic for an implementation.

CAUTION: This is not intended to be a complete reference of Business Services. Refer to the business service page to find all the supported business services.

Business Object Related Services

Business Service Name	Description
F1-AutoTransitionBO	Performs monitoring algorithms associated with the current state of a given business object instance (which may result in subsequent state transitioning).
F1-CompareBusinessObjectData	Compares two versions of a given business object instance.
F1-DetermineBo	Determines the business object of a given instance of a maintenance object by executing the MO's Determine BO logic.
F1-GetRequiredFieldsForBOState	Returns the required fields for a given business object status.
F1-RetrieveBOOption	Returns BO option values for a given BO and option type.
F1-RetrieveBOStatusOption	Returns BO option values for a given BO, status and option type.

Business Service Name	Description
F1-RetrieveBOStatusOption	Retrieves a list of BOs for a given MO that are accessible for the current user.
F1-RetrieveBoStatusDescription	Return the description of a given BO status.
F1-RetrieveBusinessObjectLabel	Return the label appropriate for a given path (e.g. element) within a BO schema.
F1-RetrieveNextStates	Return a list of next possible states based on the input of a MO and its prime key, or a BO and one of its statuses.

Email Related Services

Business Service Name	Description
F1-EmailService	Sends an email message in real time.
F1-RetrieveEmailAddress	Retrieves the email addresses of users belonging to a To Do Role.
F1-RetrieveEnvironmentURL	Retrieves the current environment URL information for the installation.

Tools for Maps and Scripting

Business Service Name	Description
F1-AddWarning	This business service may be used to log a warning by providing the message details. The system gathers all the warning and displays them to the user all at once.
F1-CheckOnlineConnection	Check if Connection is Online or Batch.
F1-CheckWarning	Used to check if any warnings have been accumulated. If warnings have been accumulated, any logic that cannot be rolled back (when the warnings are shown) should not be performed.
F1-DateMath	Performs various date and time math calculations. Refer to the BS description for more details.
F1-DateTimeFormattingService	Formats a given date / time based on the user's display profile settings.
F1-ExecuteScriptInNewSession	Executes a Service Script in a new processing session/transaction.
F1-GetFieldLabel	Retrieves the label for a given field.
F1-GetForeignKeyReference	Returns foreign key reference information for a given FK Reference and primary key, including info description, navigation option, and context menu.
F1-GetFKReferenceDetails	Returns foreign key reference information for a given MO and primary key, including FK reference code, info description, navigation option, search zone and context menu.
F1-GetLookupDescription	Returns lookup description for a lookup field value given the lookup field name.
F1-GetExtLookUpVal	Returns the list of values for a given extendable lookup BO.
F1-GetMonthInYearAbbreviation	Returns a 3-character month abbreviation for an input date in system format.

Business Service Name	Description
F1-NumberAmountFormatter	Formats a given amount or number based on the user's display profile settings. It also may receive input to adjust the scale and optionally apply currency settings.
F1-OutmsgDispatcher	Dispatches a real-time message giving the user the option of whether to persist the message on the database, and whether to trap errors that may take place during the call. Refer to Real Time Messages for more information.
F1-OutmsgMediator	Alternative to F1-OutmsgDispatcher and may be a better option if the sender does not require an outbound message record to be instantiated. Refer to Real Time Messages for more information.
F1-RethrowError	Issues an application error using the input message category / number / parameters.
F1-RetrieveMODescription	Retrieves the description for a maintenance object.
F1-ReturnMessage	Returns the expanded message given a message category, number, parameters, and parameter types.
F1-SavePointDispatcher	Allows for a service script to be executed where exceptions are trapped and the transaction is rolled back to a save point set before the service script execution.

User Related Services

Business Service Name	Comments
F1-CheckApplicationSecurity	Checks a user's security for a given application service / access mode
F1-CheckUserAuthorization	Determine whether a given user is authorized for access based on the input application service, security code, and authorization level.
F1-DeterminelfUserCanApproveTD	Determine if the current user can approve a given To Do.

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 refer to [UI Map Attributes and Functions](#).

```

<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:	Greer, Johan
Home Phone:	(838) 030-0303
Business Phone:	(737) 393-3838
Cell Phone:	(444) 444-4040
FAX:	(373) 939-3939
Social Security:	939-30-3939
Drivers License:	C8392020
Email:	jurgen.greer@media.com
	<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: Greer, Johan
Last-name suffix, prefix first-name middle-name/initial

Home Phone: 838 - 030 - 0303

Business Phone: 737 - 393 - 3838

Cell Phone: 444 - 444 - 4040

FAX: 373 - 939 - 3939

Social Security: 939 - 30 - 3939

Drivers License: C8392020

Email: jurgan.greer@media.com

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 is visible on the UI map maintenance page. This zone provides several links to launch help topics related to valid schema syntax and UI Hint syntax in one click.

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**

Use the **UI Map Query** portal to search for an existing UI Map. Once a map is selected, you are brought to the maintenance portal to view and maintain the selected record.

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 XHTML Document**, **Complete HTML Document**, **XHTML Fragment** or an **HTML Fragment**. The default value is **Complete XHTML Document**.

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.

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: Refer to [UI Map Attributes and Functions](#) and [UI Map Standards](#) for more information about HTML definition syntax. These topics 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 [UI Map Attributes and Functions](#) 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.

FASTPATH: Refer to [Schema Syntax](#) for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **UI Map Tips** zone in the dashboard provides a link to launch this help topic directly.

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.

UI Map Attributes and Functions

NOTE: This topic uses the term "field" to refer to both the generic concept of displaying and capturing data in a 'field' as well as referring to the meta-data object supplied in the product to define [Fields](#). When referring to the latter, the term "MD Field" (meta-data Field) is used and a hyperlink to the Field documentation is provided.

Contents

- [Bind XML to HTML](#)
- [Build a Dropdown List](#)
- [Format Input and Output Fields](#)
- [Display Labels](#)
- [Enable UI Map Help](#)
- [Search Using a Pop-Up Explorer Zone](#)
- [Display Errors](#)
- [Fire JavaScript for Browser Events](#)
- [Hide Elements](#)
- [Invoke Schema Based Services](#)
- [Refresh a Rendered Map or Portal Page](#)
- [Embed Framework Navigation](#)
- [Launch BPA Script](#)
- [Exit UI Map with Bound Values](#)
- [Include a Map Fragment](#)
- [Include a Web Component](#)
- [Show Schema Default on Add](#)
- [Configure a Chart](#)
- [Upload and Download a CSV File](#)
- [Construct Portal Zone Map Fragments](#)
- [Required JavaScript Libraries](#)
- [Detecting Unsaved Changes](#)
- [Hiding Portal Tabs](#)

Bind XML to HTML

Only two different attributes are required to bind a UI Map's XML to its HTML. Both of these attributes require an XML document embedded within the HTML, where the XML is bounded by <xml> nodes.

WARNING: You must embed a pair of <xml></xml> tags within your HTML document for binding to occur.

Linking a Field

Syntax	Values	Description
oraField=" "	Field element XPath	This attribute is used to link an HTML element directly with an XML element, where the XML element is defined within the UI Map's XML schema. The attribute can be used with any <i>rendering</i> HTML element, such as: , <div>, and <input>.

- HTML for input element:

```
<html>
<body>
<table>
  <tr>
    <td>Address:</td>
    <td><input type="text" oraField="address" /></td>
  </tr>
  <tr>
    <td>City:</td>
    <td><input type="text" oraField="city" /></td>
  </tr>
  <tr>
    <td>State:</td>
```



```

        <td><input type="text" oraField="state"/></td>
    </tr>
    <tr>
        <td>Zip:</td>
        <td><input type="text" oraField="zip"/></td>
    </tr>
</table>
</body>
<xml>
    <root>
        <address>123 Main St</address>
        <city>Alameda</city>
        <state>CA</state>
        <zip>94770</zip>
    </root>
</xml>
</html>

```

Rendered HTML

Address:

City:

State:

Zip:

- HTML for span and div elements:

```

<html>
<body>

<div oraField="address"></div>
<span oraField="city"></span>
<span>,</span>
<span oraField="state"></span>
<span oraField="zip"></span>
<span oraField="country"></span>

</body>
<xml>
    <root>
        <address>123 Main St</address>
        <city>Alameda</city>
        <state>CA</state>
        <zip>94770</zip>
    </root>
</xml>
</html>

```

HTML rendered:

123 Main St
Alameda, CA 94770

Linking a List

This attribute is used to link an HTML table with an XML list, where the XML list is defined within the UI Map's XML schema. The purpose of the element is to trigger the framework to replicate the table's HTML for each occurrence of the list.

Syntax	Values	Description
oraList=" "	List element XPath	This attribute is used to link an HTML table with an XML list, where the XML list is defined within the UI Map's XML schema. The purpose of the element is to trigger the

Syntax	Values	Description
		framework to replicate the table's HTML for each occurrence of the list.

NOTE: The oraField attributes embedded within the list must contain XPath navigation relative to the list. See below for an example.

```

<html>
<head><title>Bind xml list element</title></head>
<body>
<table oraList="payment">
  <thead>
    <tr>
      <th><span>Pay Date</span></th>
      <th><span>Amount</span></th>
    </tr>
  </thead>
  <tr>
    <td>
      <span oraField="date" oraType="date"></span>
    </td>
    <td align="right">
      <span oraField="amount" oraType="money"></span>
    </td>
  </tr>
</table>
</body>
<xml>
<root>
  <payment>
    <date>2008-01-01</date>
    <amount>44.28</amount>
  </payment>
  <payment>
    <date>2008-02-01</date>
    <amount>32.87</amount>
  </payment>
  <payment>
    <date>2008-03-01</date>
    <amount>21.76</amount>
  </payment>
</root>
</xml>
</html>

```

HTML rendered:

```

Pay Date Amount
01-01-2008 $44.28
02-01-2008 $32.87
03-01-2008 $21.76

```

Build a Dropdown List

The following attributes are provided to build an HTML 'select' element, also called a dropdown, based on various sources.

Source	Syntax	Values	Examples
Lookup	oraSelect="lookup: ;"	Lookup field	<pre> ... <td>House Type:</td> <td> <select oraField="houseType" oraSelect="lookup:HOUSE_TYPE;"> </select> </pre>

Source	Syntax	Values	Examples
			</td>
Extendable Lookup	oraSelect="lookupBO: ;"	BO code	<pre> ... <td>UI Device Display Type:</ td> <td> <select oraField="uiDeviceType" oraSelect="lookupBO: Fl- DeviceDisplayTypes;"></ select> </td> </pre>
Characteristic Type (pre-defined)	oraSelect="charType: ;"	Characteristic Type code	<pre> ... <td>Usage:</td> <td> <select oraField="statusReasonUsage" oraSelect="charType:Fl- SRUSG;"></select> </td> ... </pre>
Control Table	<p>oraSelect="table: ;"</p> <p>NOTE: This attribute only works with tables that follow the standard control table structure where there is a related language table that includes that includes the column DESCR as its description column. Use the Application Viewer data dictionary to identify tables that qualify for this functionality.</p> <p>WARNING: The oraSelect function will only work if less than 500 values are displayed.</p>	Table name	<pre> ... <td>Currency: </td> <td> <select oraField="currency" oraSelect="table:CI_CURRENCY_CD;" </select> </td> ... </pre>
Page Service	oraSelect="service: ;"	Page Service name	<pre> ... <td>Country:</td> <td> <select oraField="country" oraSelect="service:CIPTCNTW;" </select> </td> ... </pre>
Embedded List	<p>Used to build a select dropdown based on a list within the map's XML.</p> <p>oraSelect="valuePath: ;descPath: ;"</p>	<p>After the valuePath, indicate the XPath of the element that holds the values. After the descPath, indicate the XPath of the element that holds the descriptions.</p>	<pre> <html> <body> <table summary="" border="0" cellpadding="1" cellspacing="1"> <tr> <td>Select: </td> <td><select oraSelect= "valuePath:list/value; descPath:list/desc" oraField="target"></ select></td> </tr> </table> </body> <xml> <root> <target>10</target> <list> <value>10</value> <desc>Ten</desc> </list> </root> </xml> </pre>

Source	Syntax	Values	Examples
			<pre> </list> <list> <value>20</value> <desc>Twenty</ desc> </list> <list> <value>40</value> <desc>Forty</desc> </list> </root> </xml> </html> </pre>
Service Script	oraSelect="ss: ;"	Service Script code	See below for additional syntax needed when using this function.
Business Service	oraSelect="bs: ;"	Business Service code	See below for additional syntax needed when using this function.

When specifying a service script or a business service, extra mapping information is needed to pass data to and from the service.

Syntax	Values	Description
oraSelectIn=" ;"	serviceXPath:element;	Used to pass the value of another element into the service (mapping to the service's XPath).
	serviceXPath:'Literal';	Used to pass a constant or literal to the service (mapping to the service's XPath).
oraSelectOut="valuePath: ; descPath: "	See examples below	Used to indicate which element in the service's output holds the values and which one holds the descriptions.

Example using a business service:

```

...
<td>External System: </td>
<td>
  <select oraField="externalSystem" oraSelect="bs:F1-RetrieveExternalSystems"
oraSelectIn="outboundMsgType:boGroup/parameters/
outboundMsgType;" oraSelectOut="valuePath:results/externalSystem;
descPath:results/description"></select>
</td>
...

```

This method for building dropdowns is often used when there is a dependency between elements and the list of valid values in a dropdown (for the child element) is based on another element in the map (the parent element). When the parent element is changed, it may be required to refresh the child element. This behavior can be implemented using the function called within an **onChange** event in the map. The syntax is **oraHandleDependentElements('dependent element');** Multiple target elements (dependents) can be named.

The following example is related to the above business service example where the list of external systems is specific for a given outbound message type, which is passed in as input. The snippet below shows the configuration for the outbound message type element to trigger a refresh of the external system's dropdown list.

```

...
<div>
  <label oraLabel="boGroup/parameters/outboundMsgType"></label>
  <span>
    <select oraSelect="table:F1_OUTMSG_TYPE" oraField="boGroup/parameters/outboundMsgType"
onChange="oraHandleDependentElements('boGroup/parameters/externalSystem');"></select>
  </span>
</div>

```

...

Format Input and Output Fields

The following attributes are designed to apply data type formatting for input and output fields.

Automatic Formatting

Syntax

oraSchemaDataTypes="false"

This attribute is used to trigger automatic formatting in the rendered HTML document. The automated formatting will occur according to the data type attributes defined in the UI map's schema. For details on specific data type formatting, please refer to the oraType attribute descriptions below.

WARNING: The attribute **oraSchemaDataTypes="true"** will be automatically injected into the UI map's HTML! If you do not wish to apply the schema's data types to the rendered HTML then you must specify this attribute in the body node with a value of false. The attribute **<body oraSchemaDataTypes="false">** is required to avoid automatic formatting!

- UI Map schema:

```
<schema>
  <schemaDate dataType="date" />
  <schemaDateTime dataType="dateTime" />
  <schemaFKRef fkRef="CI_USER" />
  <schemaLookup dataType="lookup" lookup="ACCESS_MODE" />
  <schemaMoney dataType="money" />
  <schemaNumber dataType="number" />
  <schemaTime dataType="time" />
</schema>
```

- UI Map HTML:

```
<html>
<body oraSchemaDataTypes="true">
<table border="1" cellpadding="1" cellspacing="1">
<tr><th>dataType</th><th>result type</th><th>input result</th><th> display-only result</th></tr>
<tr>
  <td rowspan="2">date (from schema)</td>
<td>raw</td>
<td><input oraField="schemaDate" oraType="string" /></td>
<td><span oraField="schemaDate" oraType="string"></span></td>
</tr>
<tr>
<td>rendered</td>
  <td><input oraField="schemaDate"></td>
<td><span oraField="schemaDate"></span></td>
</tr>
<tr>
  <td rowspan="2">dateTime (from schema)</td>
<td>raw</td>
<td><input oraField="schemaDateTime" oraType="string"></td>
<td><span oraField="schemaDateTime" oraType="string"></span></td>
</tr>
<tr>
<td>rendered</td>
  <td><input oraField="schemaDateTime"></td>
<td><span oraField="schemaDateTime"></span></td>
</tr>
```

```

        <tr>
            <td rowspan="2">fkRef (from schema)**</td>
<td>raw</td>
<td><input oraField="schemaFkRef" oraType="string"></td>
<td><span oraField="schemaFkRef" oraType="string"></span></td>
        </tr>
        <tr>
<td>rendered</td>
<td><input oraField="schemaFkRef"></td>
<td><span oraField="schemaFkRef"></span></td>
        </tr>

        <tr>
            <td rowspan="2">lookup (from schema)</td>
<td>raw</td>
<td><input oraField="schemaLookup" oraType="string"></td>
<td><span oraField="schemaLookup" oraType="string"></span></td>
        </tr>
        <tr>
<td>rendered</td>
<td><input oraField="schemaLookup"></td>
<td><span oraField="schemaLookup"></span></td>
        </tr>

        <tr>
            <td rowspan="2">money (from schema)</td>
<td>raw</td>
<td><input oraField="schemaMoney" oraType="string"></td>
<td><span oraField="schemaMoney" oraType="string"></span></td>
        </tr>
        <tr>
<td>rendered</td>
<td><input oraField="schemaMoney"></td>
<td><span oraField="schemaMoney"></span></td>
        </tr>

        <tr>
            <td rowspan="2">number (from schema)</td>
<td>raw</td>
<td><input oraField="schemaNumber" oraType="string"/></td>
<td><span oraField="schemaNumber" oraType="string"></span></td>
        </tr>
        <tr>
<td>rendered</td>
<td><input oraField="schemaNumber"></td>
<td><span oraField="schemaNumber"></span></td>
        </tr>

        <tr>
            <td rowspan="2">time (from schema)</td>
<td>raw</td>
<td><input oraField="schemaTime" oraType="string"></span></td>
<td><span oraField="schemaTime" oraType="string"></span></td>
        </tr>
        <tr>
<td>rendered</td>
<td><input oraField="schemaTime"></td>
<td><span oraField="schemaTime"></span></td>
        </tr>
</table>

</body>
<xml>
<root>
<schemaDate>2007-11-02</schemaDate>
<schemaDateTime>2007-11-02-23.45.00</schemaDateTime>
<schemaFkRef>USD</schemaFkRef>
<schemaLookup>A</schemaLookup>
<schemaMoney>1000000</schemaMoney>
<schemaNumber>5661976.11548</schemaNumber>
<schemaTime>23.45.00</schemaTime>

```

```

</root>
</xml>
</html>

```

HTML rendered.

dataType	result type	input result		display-only result
date	raw	<input type="text" value="2007-11-02"/>		2007-11-02
	rendered	<input type="text" value="11-02-2007"/>		11-02-2007
dateTime	raw	<input type="text" value="2007-11-02-23.45.00"/>		2007-11-02-23.45.00
	rendered	<input type="text" value="11-02-2007"/>	<input type="text" value="11:45PM"/>	11-02-2007 11:45PM
fkRef	raw	<input type="text" value="SYSUSER"/>		SYSUSER
	rendered	<input type="text" value="SYSUSER"/>		SYSUSER
lookup	raw	<input type="text" value="A"/>		A
	rendered	<input type="text" value="Add"/>		Add
money	raw	<input type="text" value="1000000"/>		1000000
	rendered	<input type="text" value="\$1,000,000.00"/>		\$1,000,000.00
number	raw	<input type="text" value="5661976.11548"/>		5661976.11548
	rendered	<input type="text" value="5,661,976.11548"/>		5,661,976.11548
time	raw	<input type="text" value="23.45.00"/>		23.45.00
	rendered	<input type="text" value="11:45PM"/>		11:45PM

Date Formatting

This function is used to display a date according to the user's display profile. For input fields, the setting formats the data when the user tabs out of the field.

Syntax

oraType="date"

```

<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Date: </td>
    <td><span oraField="date" oraType="date"></span></td>
  </tr>
  <tr>
    <td>Date: </td>
    <td><input oraField="date" oraType="date" /></td>
  </tr>
</table>
</body>
</html>
<root>
<date>2008-12-28</date>
</root>
</xml>
</html>

```

HTML rendered.

Date: 12-28-2008

Date:

Time Formatting

This function is used to display a time according to the user's display profile. For input fields, the setting formats the data when the user tabs out of the field.

Syntax

oraType="time"

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Time: </td>
    <td><span oraField="time" oraType="time"></span></td>
  </tr>
  <tr>
    <td>Time: </td>
    <td><input oraField="time" oraType="time" /></td>
  </tr>
</table>
</body>
<xml>
<root>
<time>00.28.54.389</time>
</root>
</xml>
</html>
```

HTML rendered.

Time: 12:28AM

Time:

Date and Time Formatting

This function is used to display a timestamp according to the user's display profile. If this function is used for an input element, it is broken into two pieces, one for [date](#) and one for [time](#). Optionally, the time portion of the date time element can be suppressed using the attribute value 'time:suppress'.

Syntax

oraType="dateTime; time:suppress"

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Date time: </td>
    <td><span oraField="dateTime" oraType="dateTime"></span></td>
  </tr>
  <tr>
    <td>Date only: </td>
    <td><span oraField="dateTime" oraType="dateTime; time:suppress"></span></td>
  </tr>
  <tr>
    <td>Date time: </td>
    <td><input oraField="dateTime" oraType="dateTime" /></td>
  </tr>
</table>
```



```

    <tr>
      <td>Date only: </td>
      <td><input oraField="dateTime" oraType="dateTime; time:suppress" /></td>
    </tr>
  </table>
</body>
<xml>
<root>
<dateTime>2009-11-01-00.28.54</dateTime>
</root>
</xml>
</html>

```

HTML rendered.

Date time: 11-01-2009 12:28AM

Date only: 11-01-2009

Date time:

Date only:

Date / Time Formatting with Standard Time

This true function is used to render a date / time element according to the daylight savings time schedule of the 'base' time zone. The 'base' time zone is specified on the Installation table and represents the database time zone. For input elements with this setting, all time entered is assumed to correspond with the daylight savings time schedule of the base time zone. If a time is entered that cannot be unambiguously translated to standard time, then the user will be required to provide a time zone label to indicate whether daylight savings time, or standard time, has been entered.

Syntax

oraType="dateTime; stdTime:true;"

```

<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Date time: </td>
    <td><span oraField="dateTime" oraType="dateTime; stdTime:true;"></span></td>
  </tr>
  <tr>
    <td>Date time: </td>
    <td><input oraField="dateTime" oraType="dateTime; stdTime:true;" /></td>
  </tr>
</table>
</body>
<xml>
<root>
<dateTime>2009-11-01-00.28.54</dateTime>
</root>
</xml>
</html>

```

HTML rendered.

NOTE: The time zone label is displayed because 1:28 is ambiguous otherwise. Legally, November 1, 2009 1:28 AM occurs twice because daylight savings time (DST) is removed at 2:00 AM. With the stdTime function time zone labels are only displayed when required to clarify time overlaps.

Date time: 11-01-2009 01:28AM PDT

Date time:

HTML rendered for the following day.

Date time: 11-02-2009 12:28AM

Date time:

Date and Time Formatting with Time Zone Reference

Syntax	Valid Values	Description
<code>oraType="dateTime; stdTimeRef: ;"</code>	Reference an XPath after the colon.	This function is used to render a date / time element according to the daylight savings time schedule of a time zone whose XPath is referenced. Note that the time processed is assumed to have been stored in the standard time of the referenced time zone - so only daylight savings time shifting will execute - not time zone shifting.
<code>oraType="dateTime; displayRef: ;"</code>	Reference an XPath after the colon.	This function is similar to the <code>stdTimeRef</code> function, except that this function will execute time zone shifting in addition to daylight savings time shifting. To use <code>displayRef</code> correctly, only associate it with time zone elements that have been stored in the base time zone.

For input elements, all time entered is assumed to correspond with the daylight savings time schedule of the referenced time zone. If a time is entered that cannot be unambiguously translated to standard time, then the user will be required to provide a time zone label to indicate whether daylight savings time, or standard time, has been entered.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Date time: </td>
    <td><span oraField="dateTime" oraType="dateTime; stdTimeRef:timeZone;"></span></td>
  </tr>
  <tr>
    <td>Date time: </td>
    <td><input oraField="dateTime" oraType="dateTime; stdTimeRef:timeZone;" /></td>
  </tr>
</table>
</body>
<xml>
<root>
<timeZone>US-EAST</timeZone>
<dateTime>2009-11-01-00.28.54</dateTime>
</root>
</xml>
</html>
```

HTML rendered.

NOTE: The time zone label is always displayed for a referenced time zone.

Date time: 11-01-2009 01:28AM EDT

Date time:

HTML rendered for the following day.

Date time: 11-02-2009 12:28AM EST

Date time:

Duration Formatting

Syntax

oraType="duration"

This function is used to display time duration. For an input element, the value entered by the user is translated from minutes to hour and minutes as appropriate. For example, an entered value of '90', is converted to '00:01:30' when tabbing out of the input field.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Duration: </td>
    <td><span oraField="duration" oraType="duration"></span></td>
  </tr>
  <tr>
    <td>Duration: </td>
    <td><input oraField="duration" oraType="duration"/></td>
  </tr>
</table>
</body>
<xml>
<root>
<duration>90</duration>
</root>
</xml>
</html>
```

HTML rendered.

Duration: 00:01:30

Duration:

Day in Month Formatting

Syntax

oraType="dayInMonth"

This function is used to display a concatenated day and month.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Day In Month: </td>
    <td><span oraField="dayMonth" oraType="dayInMonth"></span></td>
  </tr>
  <tr>
    <td>Day In Month: </td>
    <td><input oraField="dayMonth" oraType="dayInMonth"/></td>
  </tr>
</table>
</body>
<xml>
<root>
<dayMonth>0228</dayMonth>
</root>
</xml>
</html>
```

HTML rendered.

Day In Month: 02-28

Day In Month:

Month In Year Formatting

Syntax

oraType="monthInYear"

This function is used to display a concatenated month and year.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Month In Year: </td>
    <td><span oraField="month" oraType="monthInYear"></span></td>
  </tr>
  <tr>
    <td>Month In Year: </td>
    <td><input oraField="month" oraType="monthInYear" /></td>
  </tr>
</table>
</body>
<xml>
<root>
<month>200811</month>
</root>
</xml>
</html>
```

HTML rendered.

Month In Year: 11-2008

Month In Year:

Monetary Formatting

This function is used to display a number as a monetary amount. See the table for configuration options with respect to the currency. For input elements, an error is issued if a non-numeric value is entered.

Syntax	Description
oraType="money: "	Directly specify a currency code after the colon.
oraType="money;currencyRef: "	Reference an XPath (after the colon) for an element that references a currency code.
oraType="money"	If no currency or currency reference is specified, the installation currency is used.

NOTE: You must specify a pair of stylesheet references, `cisEnabled` and `cisDisabled`, in the map's header for right alignment. The stylesheet controls how the field will be rendered. If you want to alter the rendering you must override the `oraMoney` style.

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css" />
  <link rel="stylesheet" type="text/css" href="cisEnabled.css" />
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
```

```

    <tr>
      <td>Amount, currency specified:</td>
    <td><span oraType="money:EUR" oraField="totalAmt "></span></td>
  </tr>
  <tr>
    <td>Amount, default currency:</td>
    <td><span oraType="money" oraField="totalAmt "></span></td>
  </tr>
  <tr>
    <td>Amount, default input:</td>
    <td><input oraType="money" oraField="totalAmt " /></td>
  </tr>
  <tr>
    <td>Amount, currency reference:</td>
    <td><input oraType="money;currencyRef:cur" oraField="totalAmt " /></td>
  </tr>
</table>
</body>
<xml>
<root>
<totalAmt>50500.09</totalAmt>
<cur>EUR</cur>
</root>
</xml>
</html>

```

HTML rendered

```

Amount, currency specified: €50500.09
Amount, default currency:  $50500.09
Amount, default input:    
Amount, currency reference: 

```

Number Formatting

This function is used to display a number or validate an input value. For input elements, the system will return an error if a non-numeric value is entered.

Syntax

oraType="number"

NOTE: You must specify a pair of stylesheet references, `cisEnabled` and `cisDisabled`, in the map's header for right alignment. The stylesheet controls how the field will be rendered. If you want to alter the rendering you must override the `oraNumber` style.

```

<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css" />
  <link rel="stylesheet" type="text/css" href="cisEnabled.css" />
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Count:</td>
    <td><span oraType="number" oraField="count "></span></td>
  </tr>
  <tr>
    <td>Count, input:</td>
    <td><input oraType="number" oraField="count " /></td>
  </tr>
</table>
</body>
<xml>
<root>
<count>989</count>

```

```
</root>
</xml>
</html>
```

HTML rendered.

Count: 989
Count, input:

FK Reference Formatting

By default, when an element with an **fkRef** oraType is displayed, an info string, context menu, navigation, and search are enabled (if the FK reference has been configured accordingly). Syntax is provided to allow you to selectively turn off any of these features.

Note that you can enable the foreign key hyperlink separately as well, refer to [Embed Framework Navigation](#) for more information. The various attributes used to control foreign key reference functionality are as follows. Note that in every case, the default value is **true**. A value of **false** should be used to disable the feature.

Syntax

oraType="fkRef:true|false; info:true|false; context:true|false; navigation:true|false; search:true|false"

- **fkRef.** A value of 'true' enables all of the following foreign key reference processing. Use a value of 'false' to disable automatic FK Reference processing.
- **info.** A value of 'true' renders an [information string](#) on the UI map, if applicable.
- **context.** A value of 'true' renders a context menu to appear before the foreign key reference element, if applicable.
- **navigation.** A value of 'true' causes the information string to be rendered as a hyperlink, if applicable. Clicking the hyperlink [navigates](#) to the appropriate page.
- **search.** A value of 'true' displays a search icon that launches the [search zone](#) if applicable.

NOTE: Foreign key navigation and context menu functionality is only available for UI maps presented in a portal zone. UI Maps presented during BPA script processing cannot support navigation options. Search functionality is only available for input HTML elements.

- UI Map schema:

```
<schema>
  <bo fkRef="F1-BOMO" />
</schema>
```

- UI Map HTML:

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
<td>Business Object</td>
<td><span oraField="bo" oraType="fkRef:true; info:true; context:true; navigation:true;"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
<bo>F1-COUNTRY</bo>
</root>
</xml>
```

```
</html>
```

- HTML rendered.

BUSINESS OBJECT

▼ Country

Lookup Formatting

This function is used to display the description of a lookup value.

Syntax	Valid Values
<code>oraType="lookup: "</code>	Lookup field name after the colon.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Status:</td>
    <td><span oraField="status" oraType="lookup: BATCH_JOB_STAT_FLG"></span></td>
  </tr>
</table>
</body>
</html>
<xml>
<root>
  <status>PD</status>
</root>
</xml>
</html>
```

HTML rendered.

Status: Pending

Extendable Lookup Formatting

This function is used to display the description of an extendable lookup value.

Syntax	Valid Values
<code>oraType="lookupBO: "</code>	Business Object code name after the colon.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Value:</td>
    <td><span oraField="status" oraType="lookupBO: F1-DeviceDisplayTypes"></span></td>
  </tr>
</table>
</body>
</html>
<xml>
<root>
  <status>oraTablet</status>
</root>
</xml>
</html>
```

HTML rendered.

Value: Tablet size

Characteristic Type Formatting

This function is used to display the description of a predefined characteristic value.

Syntax	Valid Values
<code>oraType="charType: "</code>	Characteristic Type code after the colon.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Skill:</td>
    <td><span oraType="charType:CM-SKILL" oraField="skill"></span></td>
  </tr>
</table>
</body>
</html>
<xml>
<root>
  <skill>10</skill>
</root>
</xml>
</html>
```

HTML rendered.

Skill: Quality assurance

Control Table Formatting

This function is used to display the description of a control table that has an associated language table.

Syntax	Valid Values
<code>oraType="table: "</code>	Table code after the colon.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Currency:</td>
    <td><span oraType="table:CI_CURRENCY_CD" oraField="curr"></span></td>
  </tr>
</table>
</body>
</html>
<xml>
<root>
  <curr>USD</curr>
</root>
</xml>
</html>
```

HTML rendered.

Currency: United States Dollars

Add / Remove Grid Formatting

Syntax	Description
<code>oraType="addGridRow"</code>	The addGridRow function is used to build "insert row" dialog into the UI map.

Syntax**Description****oraType="deleteGridRow"**

- An 'add' image is displayed.
- Clicking the image inserts a new row in the grid.
- If the list is empty, by default, an empty grid row is automatically be added. This means that the user will always see at least one grid row when this attribute is used.

The deleteGridRow function is used to build "delete row" dialog into the UI map.

- A 'delete' image is displayed.
- Clicking the image removes the adjacent row from the grid.

NOTE: Because add and delete dialogs are relevant only inside a table, these attributes must be specified within a <td> element.

WARNING: These attributes are designed to work with the business object action of 'replace' rather than 'update'. Therefore, if the map contains a grid, the business object action of 'replace' must be used to update the business object. Refer to [BO Replace Action](#) for more information.

Example:

```
<html>
<head>
<title>Demonstrate Grid Add and Grid Delete OraTypes</title>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table oraList="listEntry">
  <thead>
    <tr>
      <th/>
      <th/>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td oraType="addGridRow"></td>
      <td oraType="deleteGridRow"></td>
      <td>
        <input oraField="date" oraType="date"></input>
      </td>
      <td align="right">
        <input oraField="amount" oraType="money"></input>
      </td>
    </tr>
  </tbody>
</table>
</body>
<xml>
<root>
  <listEntry>
    <date>2008-01-01</date>
    <amount>44.28</amount>
  </listEntry>
  <listEntry>
    <date>2008-02-01</date>
    <amount>32.87</amount>
  </listEntry>
  <listEntry>
    <date>2008-03-01</date>
```

```

<amount>21.76</amount>
  </listEntry>
</root>
</xml>
</html>

```

HTML rendered.

	DATE		AMOUNT
+	01-01-2008		\$44.28
+	02-01-2008		\$32.87
+	03-01-2008		\$21.76

Unformatted Elements

This function is used to display the contents of an element that contains 'raw' data as defined for the schema element being rendered.

Syntax

oraType="raw"

- UI Map schema:

```

<schema>
  <rawStuff type="raw"/>
</schema>

```

- UI Map HTML:

```

<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Raw Stuff:</td>
    <td><span oraType="raw" oraField="rawStuff"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
  <rawStuff>
    <ele1>text in element 1</ele1>
    <group1>
      <ele2>text inside element 2, group 1</ele2>
      <ele3>text inside element 3, group 1</ele3>
    </group1>
  </rawStuff>
</root>
</xml>
</html>

```

HTML rendered.

Raw Stuff:	<pre><ele1>text in element 1</ele1> <group1> <ele2>text inside element 2, group 1</ele2> <ele3>text inside element 3, group 1</ele3> </group1></pre>
------------	--

String Formatting

This function is used to display the contents of an element, as XML pretty-print, when the element contains escaped XML.

Syntax

oraType="xmlString"

NOTE: It is not required, but the pretty print of the rendered XML is enhanced if you specify a pair of stylesheet references, cisEnabled and cisDisabled, in the map's header.

Example:

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>XML Stuff:</td>
    <td><span oraType="xmlString" oraField="xmlStuff"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
<xmlStuff>
  <ele1>text in element 1</ele1>
  <group1>
    <ele2>text inside element 2, group 1</ele2>
    <ele3>text inside element 3, group 1</ele3>
  </group1>
</xmlStuff>
</root>
</xml>
</html>
```

HTML rendered.

XML Stuff:	<pre><ele1>text in element 1</ele1> <group1> <ele2>text inside element 2, group 1</ele2> <ele3>text inside element 3, group 1</ele3> </group1></pre>
------------	--

HTML rendered without oraType="xmlString"

XML Stuff:	<pre><ele1>text in element 1</ele1><group1><ele2>text inside element 2, group 1</ele2><ele3>text inside element 3, group 1</ele3></group1></pre>
------------	--

HTML Formatting

This function is used to display the contents of an element as HTML as opposed to plain text. An element defined as oraType="fkref" is automatically rendered as HTML.

Syntax

`oraType="html"`

WARNING:

To avoid execution of malicious HTML not all HTML tags are supported. The list of supported tags is defined in the "F1-HTMLWhiteList" managed content definition.

If unsupported HTML is detected the entire element is escaped and rendered as plain text. It is therefore recommended to properly escape any source string that contributes to the final HTML element if it is not expected to contain valid HTML. This way only the offending string is escaped and not the entire element.

If the HTML element is composed in scripting refer to the 'escape' function described in the [Edit Data Syntax](#) for more information. Use the `WebStringUtilities.asHTML` java API for escaping text composed in Java.

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
  <tr>
    <td>Info :</td>
    <td><span oraType="html" oraField="info"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
  <info><b>text in bold</b></info>
</root>
</xml>
</html>
```

HTML rendered.

Info : **text in bold**

HTML rendered without `oraType="html"`

Info : text in bold

Display Labels

Derive Label from an Element

This attribute is used to obtain a language sensitive label for a ``, `<td>`, or `<input>` HTML element.

Syntax

`oraLabel=" "`

Valid Values

XPath of an element in the UI map schema. The element must reference either the `mapField=`, `mdField=`, or `label=` attribute.

NOTE: You can also define a field directly in your HTML for label definition, refer to [Deriving Label from a Field](#) for more information.

NOTE: If the schema contains multiple attributes, the oraLabel attribute will pick the label to render according to the following hierarchy: The label attribute overrides the mdField attribute, which in turn will override the mapField attribute.

- UI Map schema:

```
<schema>
  <user mapField="USER_ID"/>
  <info type="group" mapXML="CLOB">
    <city label="Metro Area"/>
    <age mdField="AGE_LBL"/>
  </info>
</schema>
```

- HTML:

```
<html>
<head><title oraMdLabel="BUS_LBL"></title></head>
<body>
<table>
  <tr>
    <td oraLabel="user"></td>
    <td><input oraField="user"/></td>
  </tr>
  <tr>
    <td oraLabel="info/city"></td>
    <td><input oraField="info/city"/></td>
  </tr>
  <tr>
    <td oraLabel="info/age"></td>
    <td><input oraField="info/age"/></td>
  </tr>
  <tr>
    <td/>
    <td><input type="button" oraMdLabel="ACCEPT_LBL"/></td>
  </tr>
</table>
</body>
<xml>
  <root>
    <user>RWINSTON</user>
    <info>
      <city>Alameda</city>
      <age>32</age>
    </info>
  </root>
</xml>
</html>
```

HTML rendered:

User	<input type="text" value="RWINSTON"/>
Metro Area	<input type="text" value="Alameda"/>
Age	<input type="text" value="32"/>
	<input type="button" value="Accept"/>

Deriving Label from a Field

This attribute is used to obtain a language sensitive label for a , <td>, <input>, or <title> HTML element. The label text is derived from the field referenced.

Syntax**Valid Values**

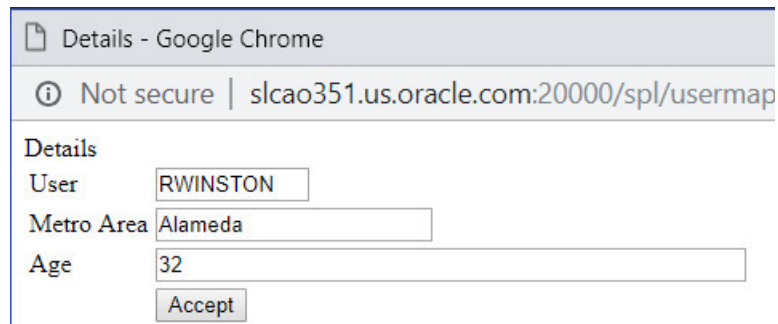
oraMdLabel=" "MD Field code.

NOTE: You can also define labels derived from the map's schema definition, refer to [Derive Label from an Element](#) for more information.

- HTML:

```
<html>
<head><title oraMdLabel="F1_DETAILS_LBL"></title></head>
<body>
<table>
  <tr>
    <td oraLabel="user"></td>
    <td><input oraField="user"/></td>
  </tr>
  <tr>
    <td oraLabel="info/city"></td>
    <td><input oraField="info/city"/></td>
  </tr>
  <tr>
    <td oraLabel="info/age"></td>
    <td><input oraField="info/age"/></td>
  </tr>
  <tr>
    <td/>
    <td><input type="button" oraMdLabel="ACCEPT_LBL"/></td>
  </tr>
</table>
</body>
<xml>
  <root>
    <user>RWINSTON</user>
    <info>
      <city>Alameda</city>
      <age>32</age>
    </info>
  </root>
</xml>
</html>
```

HTML rendered:



The screenshot shows a web browser window titled "Details - Google Chrome". The address bar displays "Not secure | slcao351.us.oracle.com:20000/spl/usermap". Below the address bar, there is a "Details" section with three input fields: "User" containing "RWINSTON", "Metro Area" containing "Alameda", and "Age" containing "32". At the bottom of this section is an "Accept" button.

Enable UI Map Help

The [Display Labels](#) section describes ways to derive the label for an element using an underlying [MD Field](#). In addition, if the same MD field contains help text, the system will automatically generate a tool tip adjacent to the element label. Clicking the tool tip allows the user to view the help text.

It is possible to change the rendering of the tool tip. Refer to [Custom Look And Feel Options](#) for more information

Search Using a Pop-Up Explorer Zone

Search Option

This attribute is used to enable search zone functionality for input HTML elements.

Syntax	Valid Values
<code>oraSearch=" "</code>	Zone code.

NOTE: The `oraSearch` attribute is similar to the `oraType` attribute, because it will be 'automatically' included into HTML via the `oraSchemaDataTypes` attribute. This means that coding the `oraSearch` attribute into UI Map HTML is only required if a search zone has not been specified in the schema, or in the schema element's FK reference.

- Example of defining the search in the HTML. UI Map's Schema:

```
<schema>
  <uiMap/>
</schema>
```

UI Map's HTML

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>UI Map with Search </td>
    <td><input oraField="uiMap" oraSearch="F1-UISRCH"></td>
  </tr>
</table>
</body>
<xml>
<root>
  <uiMap/>
</root>
</xml>
</html>
```

- Example of defining the search in the schema. UI Map's Schema:

```
<schema>
  <uiMap search="F1-UISRCH"/>
</schema>
```

UI Map's HTML

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>UI Map with Search </td>
    <td><input oraField="uiMap"></td>
  </tr>
</table>
</body>
<xml>
<root>
  <uiMap/>
</root>
</xml>
</html>
```

- Example where the FK reference defines the search zone. UI Map's Schema:

```
<schema>
  <uiMap fkRef="F1-UISRC"/>
```

```
</schema>
```

UI Map's HTML

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>UI Map with Search </td>
    <td><input oraField="uiMap"></td>
  </tr>
</table>
</body>
<xml>
<root>
  <uiMap/>
</root>
</xml>
</html>
```

In all cases, the HTML rendered is the same.

UI MAP WITH SEARCH

Initializing Search Fields

This optional attribute is used to initialize search zone filters. Multiple filters may be initialized. This attribute can only be used in conjunction with the oraSearch attribute.

Syntax	Valid Values	Field Value Options	Comments
<code>oraSearchField=""</code>	One or more pairs of field name and field value separated by colon. Each pair is separated by a semi-colon. <code>oraSearchField="fieldName:fieldValue: ..."</code>	No value	If you do not specify a field value, then the value of the input element containing the oraSearchField attribute will be used.
	The field name is used to identify the zone filter to initialize when the search is launched. The field name must match the value of the searchField mnemonic specified on a search zone user filter or hidden filter parameter.	XPath 'literal'	Indicate the XPath to the schema element that contains the value to use. Indicate a literal value to supply.

NOTE: If you do not specify an oraSearchField attribute and the schema element has a search enabled fkRef specified, the framework automatically builds an oraSearchField where the field name is equal to the FK reference's key (MD) [fields](#).

WARNING: The pop-up explorer zone can be invoked one of two ways: By clicking on the search button, or by hitting the enter key from the field to the left of the button. If you click on the button then no search field information will be passed to the zone. Search field information is only used to initialize zone filter values when enter is pressed.

Two filter values are initialized as shown in the following example:

```
<schema>
  <bo/>
  <uiMap/>
</schema>
```

```
<html>
<body>
```



```

<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>UI Map with Search </td>
    <td><input oraField="uiMap" oraSearch="F1-
UISRCH" oraSearchField="MAP_CD; BUS_OBJ_CD:bo;"></td>
  </tr>
</table>
</body>
<xml>
<root>
  <bo/>
  <uiMap/>
</root>
</xml>
</html>

```

Mapping Returned Search Fields

This optional attribute is used to direct values returned by the search zone. Multiple fields may be specified. This attribute can only be used in conjunction with the oraSearch attribute.

Syntax	Valid Values	Field Value Options	Comments
oraSearchOut=" "	One or more pairs of field name and field value separated by colon. Each pair is separated by a semi-colon.	No value	If you do not specify a field value, then the input element containing the oraSearchField attribute receives the returned value.
oraSearchOut="field name:xpath target; ..."	The field name is used to identify the search result returned from the query zone. The field name must match the ELEMENT_NAME mnemonic defined within the explorer zone's search results parameter.	XPath	Indicate the XPath to the schema element that should receive the returned value.

NOTE: If you do not specify an oraSearchOut attribute, the framework will build a default where the field name will be set equal to the oraSearchField's field name.

Two values are returned in the following example:

```

<schema>
  <bo/>
  <mo/>
</schema>

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>BO Search </td>
    <td><input oraField="bo" oraSearch="Z1-BOSRCH" oraSearchOut="BUS_OBJ_CD; MO_CD:mo;"></td>
  </tr>
</table>
</body>
<xml>
<root>
  <bo/>
  <mo/>
</root>
</xml>
</html>

```

Display Errors

Display Error Variables

One of the following error variables may be displayed.

Syntax

`oraErrorVar="ERRMSG-TEXT"`

`oraErrorVar="ERRMSG-LONG"`

`oraErrorVar="ERRMSG-CATEGORY"`

`oraErrorVar="ERRMSG-NUMBER"`

```
...  
<table width="100%" cellpadding="12">  
  <tr class="oraErrorText">  
    <td>  
      <a href="" onclick="oraShowErrorAlert(); return false;">  
<span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>  
      </a>  
    </td>  
  </tr>  
</table>  
...
```

HTML rendered



Last name required

User Id BOND007

First Name

Last Name

Highlight a Field in Error

NOTE: For more information on throwing an error, refer to the [Terminate Statement](#) in the Edit Data Syntax.

Syntax

`oraError="automate:true|false; prefix: "`

Comments

Specifying **automate:true** automatically enables highlighting of the element in error when issuing an error. Note that **true** is the default and doesn't need to be specified. Specify **automate:false** to turn off field highlighting.

The system uses a match on the element name referenced in the error to the element names in the UI map. If the elements in the schema are within an XPath that may not match what is referenced by the error, use **prefix:XPath** to specify that.

NOTE: A pair of stylesheet references, `cisEnabled` and `cisDisabled`, must be specified for reference of the `oraError` style. The stylesheet controls how the field in error will be rendered. If you want to alter the rendering you must override the `oraError` style.

The following HTML example shows that the elements in the map are defined within a group called **boGroup**. The element name returned by the error will not include this group so in order for the field highlighting to work properly the **prefix:** attribute must indicate the group name.

```
<html>
<head>
  <title>User Zone Input</title>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body oraError="automate:true; prefix:boGroup">
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td>
      <a href="" onclick="oraShowErrorAlert(); return false;">
        <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
      </a>
    </td>
  </tr>
</table>
<table width="100%" border="0" cellpadding="4">
  <tr style="padding-top:30px;">
    <td align="right" class="label">User Id</td>
    <td>
      <span oraField="boGroup/userId" class="normal"/>
    </td>
  </tr>

  <tr>
    <td align="right" class="label">First Name</td>
    <td>
      <input oraField="boGroup/firstName" class="normal"/>
    </td>
  </tr>

  <tr>
    <td align="right" class="label">Last Name</td>
    <td>
      <input oraField="boGroup/lastName" class="normal"/>
    </td>
  </tr>
</table>
</body>

<xml>
<root>
<boGroup>
  <userId>BOND007</userId>
  <firstName>James</firstName>
  <lastName></lastName>
</boGroup>
</root>
</xml>
</html>
```

HTML rendered, where the error element thrown is equal to 'lastName':



Overriding the Error Element Name

In the rare occasion where the element name returned by the error doesn't match the element name in the map, you can add an explicit attribute to indicate the error element name.

Syntax	Valid Values	Comments
<code>oraErrorElement=</code>	"element name"	The element name referenced here must exactly match the name of the error element assigned when the error was thrown. More than one HTML field can be referenced by the same error element name.

NOTE: A pair of stylesheet references, `cisEnabled` and `cisDisabled`, must be specified for reference of the `oraError` style. The stylesheet controls how the field in error will be rendered. If you want to alter the rendering you must override the `oraError` style.

This illustrates a scenario where the element name associated with the error differs from the element name on the map.

```
<html>
<head>
  <title>User Zone Input</title>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td>
      <a href="#" onclick="oraShowErrorAlert(); return false;">
        <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
      </a>
    </td>
  </tr>
</table>
<table width="100%" border="0" cellpadding="4">
  <tr style="padding-top:30px;">
    <td align="right" class="label">User Id</td>
    <td>
      <span oraField="userId" class="normal"/>
    </td>
  </tr>

  <tr>
    <td align="right" class="label">First Name</td>
    <td>
      <input oraField="firstName" class="normal" oraErrorElement="firstName"/>
    </td>
  </tr>

  <tr>
    <td align="right" class="label">Last Name</td>
    <td>
      <input oraField="familyName" class="normal" oraErrorElement="lastName"/>
    </td>
  </tr>
</table>
</body>

<xml>
<root>
  <userId>BOND007</userId>
  <firstName>James</firstName>
  <familyName></familyName>
</root>
</xml>
</html>
```

HTML rendered.

Last name required

User Id BOND007

First Name

Last Name

Display Error Pop-Up

When the error text is displayed, this function may be used to pop-up the standard error dialog (which displays more information) when a user clicks the error message.

Syntax

oraShowErrorAlert(); return false;

```

<html>
<head>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td>
      <a href="" onclick="oraShowErrorAlert(); return false;">
        <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
      </a>
    </td>
  </tr>
</table>
<table>
  <tr>
    <td>>Address:</td>
    <td><input type="text" oraField="address"/></td>
  </tr>
  <tr>
    <td>>City:</td>
    <td><input type="text" oraField="city"/></td>
  </tr>
  <tr>
    <td>>State:</td>
    <td><input type="text" oraField="state"/></td>
  </tr>
  <tr>
    <td>>Zip:</td>
    <td><input type="text" oraField="zip"/></td>
  </tr>
  <td/>
  <td style="padding-top:15px;">
<oraInclude map="F1-SaveCancelButtons"/>
  </td>
</tr>
</table>
</body>
<xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
  </root>
</xml>
</html>

```

HTML rendered.

Address field missing

Address:

City:

State:

Zip:

Standard error pop-up dialog launched via click on error message:

slcao351.us.oracle.com:20000 says
Description is missing.

If you need support please supply the following info to system support:

Message number: 11001, 1853 Element: description

Fire JavaScript for Browser Events

Working with the JavaScript Framework

There are many JavaScript events that can be used within the HTML/Javascript environment. These include events such as onLoad, onBlur, onChange, etc. The UI Map Framework also makes use of some of these events. It is important that any UI Map you develop works with the framework in order to obtain consistent results (events may not always be executed in the same order at all times!).

WARNING:

The following describes the recommended approach for safely handling loading and processing field updates in your UI Maps.

If JavaScript is required within an XHTML UI Map or fragment, it will be necessary to bound it within a `![CDATA[]]` tag to ensure a valid XML document. Note that the tags themselves may need to be commented out to promote compatibility with older browsers. For example:

```
<script type="text/javascript">
```

```
/* <![CDATA[ */
```

```
//
```

```
// javascript
```

```
//
```

```
/* ]]> */
```

```
</script>
```

Element Change Event

Syntax	Valid Values
<code>oraChange=" "</code>	A JavaScript function.

At the time of UI Map load, if there is an event handler already attached to an HTML element, the framework removes it and attaches a combined event handler. The combined handler calls any **framework handler first** and then calls the other (custom) handlers.

WARNING: Note that the function must not be used to execute logic that will modify the associated field data value again - or an endless loop will occur.

In the following example the **oraInvokeBS** function is executed when the button is clicked.

```
<html>
  <head>
    <title>oraInvokeBS test</title>
  </head>
  <body>
    <table>
      <tr>
        <td>User Id:</td>
        <td>
          <input oraField= "xmlGroup/userId" />
          <input type="button" value="Search" oraChange="oraInvokeBS('UserSearch', 'xmlGroup');" />
        </td>
      </tr>
      <tr>
        <td/>
        <td>Search Results</td>
      </tr>
      <tr>
        <td/>
        <td>
          <table oraList="xmlGroup/searchList">
            <tr>
              <td><span oraField="userId"></span>
            </td>
            </tr>
          </table>
        </td>
      </tr>
    </table>
  </body>
</html>
<xml>
  <root>
    <xmlGroup>
      <userId/>
      <searchList>
        <userId></userId>
      </searchList>
    </xmlGroup>
  </root>
</xml>
</html>
```

Page Load Event

Syntax	Valid Values
<code>oraLoad=" "</code>	A JavaScript function.

WARNING: When executing `oraLoad` within a fragment UI map, and you need to execute a JavaScript function during page load (where the function invokes a business object, business service, or service script) you can use the special

syntax `oraLoad[$SEQUENCEID]`. For other special syntax used for map fragments, refer to the [Construct a Portal Zone Header](#) section.

- In the following example, the `oraDisplayNone` function is executed during page load:

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td oraLoad="oraDisplayNone(item,'userId','')">User Id: </td>
    <td><span oraField="userId"></span></td>
  </tr>
</table>
</body>
</html>
<xml>
<root>
  <userId>SPLAXT</userId>
</root>
</xml>
</html>
```

- HTML rendered

User Id: SPLAXT

After Page Load Event

Syntax	Valid Values
<code>oraAfterLoad=" "</code>	A JavaScript function.

In the following example the `oraGetValueFromScript` function is executed after page load.

```
<div>
<label for="boGroup_parameters_externalSystem" oraLabel="boGroup/parameters/externalSystem">
</label>
<span>
  <select oraSelect="bs:F1-RetrieveExternalSystems" class="oraInput"
    id="boGroup_parameters_externalSystem" oraField="boGroup/parameters/externalSystem"
    oraSelectOut="valuePath:results/externalSystem; descPath:results/description"
    oraSelectIn="outboundMsgType:boGroup/parameters/outboundMsgType"
    oraAfterLoad
      ="oraGetValueFromScript(document.getElementById('boGroup_parameters_externalSystem'));">
  </select>
</span>
</div>
```

Hide Elements

Hide Using a Function

The system provides multiple techniques for hiding an HTML element.

Syntax	Valid Values	Comments
<code>oraDisplayNone(item);</code>	(item, 'XPath', 'value', 'operator')	Used to hide an element based on the value of another element (referenced using its XPath). Enter a value of '' to interrogate a blank value. By default the operator is '='. This may be set instead to another operator such as '!=', '>', or '<'.

Syntax	Valid Values	Comments
	(item, function name, true false)	Used to indicate a JavaScript function, which must return a Boolean.
oralsSuppress="true"		Use this option for fields that need to be hidden always and not dependent on something else. Set this attribute on the HTML element (div element).

- Example where the User Id label is hidden when no User Id value exists.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td oraLoad="oraDisplayNone(item, 'userId', '')">User Id: </td>
    <td><span oraField="userId"></span></td>
  </tr>
</table>
</body>
<xml>
<root>
  <userId></userId>
</root>
</xml>
</html>
```

- Example where the Save button is hidden when the user does not have security access to the action of change ('C') for the application service 'F1-DFLTS'.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td oraLoad="oraDisplayNone(item, oraHasSecurity('F1-DFLTS', 'C'), false );">
      <input name="Save" type="button" onclick="oraInvokeBO('CM-
IndividualTaxpayer', null, 'replace')"/>
    </td>
  </tr>
</table>
</body>
<xml>
<root>
  <userId></userId>
</root>
</xml>
</html>
```

Check User's Security Access

The system provides two functions to check a user's security access to a given application service and access mode. These are commonly used for hiding elements.

Syntax	Parameters
oraHasSecurity()	'Application Service code'
	'Access Mode'
oraHasSecurityPath('x','y')	'Application Service XPath'
	'Access Mode XPath'

See the previous section for an example of the **oraHasSecurity** function. The following shows an example where the status button is hidden when the user does not have security access to the access mode 'ACT' of the application service 'FORMTST'.

```
<html>
<body>
<table>
  <tr>
    <td oraLoad="oraDisplayNone(item, oraHasSecurityPath('appService', 'accessMode'), false );">
      <input oraField="statusLabel" type="button" onclick="oraRunScript('UpdateState','status');"/>
    </td>
  </tr>
</table>
</body>
<xml>
<root>
  <status>ACTIVATE</status>
<statusLabel>Activate</statusLabel>
<appService>FORMTST</appService>
  <accessMode>ACT</accessMode>
</root>
</xml>
</html>
```

Invoke Schema Based Services

The system provides functions for invoking a business object, business service or service script.

Invoke BO Function

This function is used to perform a BO interaction directly from the UI map's HTML. It returns a 'true' or a 'false' depending on whether the invocation encounters an error.

Syntax	Parameters	Comments
oralInvokeBO()	'BO Name'	
	'XPath' or null	Identifies a group element via XPath. If you specify the word null , then the entire embedded XML object will be passed.
	'action'	Indicate the action to use. Valid values are add, delete, read, update, replace, fastAdd and fastUpdate . FASTPATH: Refer to BO Actions for more information about the various BO actions.

Example with the statement invoked in a JavaScript function.

```
function invokeBO {
  if (!oraInvokeBO('F1-User', 'xmlGroup', 'read')) {
    oraShowErrorAlert();
    return;
  }
}
```

Example with the statement invoked within onClick.

```
<input type="button" class="oraButton" oraMdLabel="DISPLAY_LBL" onClick="oraInvokeBO('F1-User');"/>
```

Invoke BS Function

This function is used to perform a business service interaction directly from the UI map's HTML. It returns a 'true' or a 'false' depending on whether the invocation encounters an error.

Syntax	Parameters	Comments
<code>oraInvokeBS()</code>	'BS Name' 'XPath' or null	Identifies a group element via XPath. If you specify the word null , then the entire embedded XML object will be passed.

Example with the statement coded within a JavaScript function.

```
function invokeBS {
    if (!oraInvokeBS('F1-UserSearch', 'xmlGroup')) {
        oraShowErrorAlert();
        return;
    }
}
```

Example with the statement invoked via `onClick`.

```
<input type="button" class="oraButton" oraMdLabel="DISPLAY_LBL" onClick="oraInvokeBS('F1-RetrieveInfo');"/>
```

Invoke SS Function

This function is used to perform a service script interaction directly from the UI map's HTML. It returns a 'true' or a 'false' depending on whether the invocation encounters an error.

Syntax	Parameters	Comments
<code>oraInvokeSS()</code>	'Service Script Name' 'XPath' or null	Identifies a group element via XPath. If you specify the word null , then the document belonging to the parent node will be passed. If the parent node is not enough, then the entire document can always be passed using the following syntax: <code>oraInvokeSS('service script', null, null, {\$SEQUENCEID})</code>

Example with the statement invoked within a JavaScript function:

```
function invokeSS {
    if (!oraInvokeSS('F1-GetUsers', 'xmlGroup')) {
        oraShowErrorAlert();
        return;
    }
}
```

Example with the statement invoked within `onClick`.

```
<input type="button" class="oraButton" oraMdLabel="DISPLAY_LBL" onClick="oraInvokeSS('F1-GetUserInfo');"/>
```

Refresh a Rendered Map or Portal Page

Refresh Map

This function is used to 'Refresh' only the map zone issuing the command.

Syntax

oraRefreshMap;

```
...
  <tr>
    <td/>
    <td><input type="button" onclick="oraRefreshMap();" value="Refresh"/></td>
  </tr>
...
```

Refresh Page

This function is used to refresh all zones in the portal.

Syntax

oraRefreshPage;

```
...
  <tr>
    <td/>
    <td><input type="button" onclick="oraRefreshPage();" value="Refresh"/></td>
  </tr>
...
```

Embed Framework Navigation

Navigate using Navigation Option

This function is used to navigate to another page using the information defined on a navigation option.

Syntax

oraNavigate();

Parameters

'Navigation Option code'

'Key XPath'

WARNING: This function is only intended for a UI map defined within a portal zone. It should not be used within a UI map launched by a BPA script.

The following example exhibits two possible uses of this function: as a URL and as a button. Note that the UI Map schema must contain a fkRef attribute. Refer to [FK Reference Formatting](#) for more information.

```
<schema>
  <userId fkRef="CI_USER"/>
</schema>

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>User Link: </td>
    <td><a href="" onclick="oraNavigate('userMaint','userId'); return false;">
      <span oraField="userId" oraType="fkRef:CI_USER"></span></a>
    </td>
  </tr>
</table>
<tr>
  <td>User Button: </td>
  <td><input type="submit" onclick="oraNavigate('userMaint','userId')"
    value="Go to User"/></td>
</tr>
</table>
</body>
<xml>
```

```
<root>
  <userId>SPLAXT</userId>
</root>
</xml>
</html>
```

HTML rendered.



Launch BPA Script

Launch BPA Script

Syntax	Parameters	Comments
<code>oraRunScript();</code>	'BPA script code'. 'XPath Element'	One or more element values may be passed into the BPA where it may be referenced as temporary variables.

WARNING: This function is only applicable to UI maps displayed in portal zones. UI maps launched within a running BPA script cannot directly launch another BPA script from the UI map's HTML. Instead, return a value from the UI map and execute a Perform Script or Transfer Control step type.

NOTE: It is incumbent on the script author to pull information out of temporary storage in the initial steps of the script.

In the following example, a temporary variable named 'personId' is created with value '1234567890' and the BPA script named 'Edit Address' is launched.

```
<html>
<body>
<table>
  <tr>
    <td>
      <div oraField="address"></div>
      <span oraField="city"></span>
      <span>,</span>
      <span oraField="state"></span>
      <span oraField="zip"></span>
      <span oraField="country"></span>
      <a href="" onClick="oraRunScript('Edit Address','personId');">edit</a>
    </td>
  </tr>
</table>
</body>
<xml>
  <root>
    <personId>1234567890</personId>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
  </root>
</xml>
</html>
```

HTML rendered.

123 Main St
Alameda, CA 94770 [edit](#)

Launch BPA Script With Values

This function is used to launch a BPA, providing name/value pairs to push into temporary storage. Multiple values can be passed. The BPA script can then reference the temporary variables by name.

Syntax	Parameters	Comments
<code>oraRunScriptWithValues();</code>	'BPA script code'. 'XPath Element Name':value	One or more pairs of element names and values.

NOTE: You would use this JavaScript function, instead of `oraRunScript`, if you need to push values to the BPA script that are not located in the UI Map's XML structure.

In the example below, a JavaScript function named 'editUser()' is responsible for launching the BPA script named 'UserEdit'. The temporary variable named 'userId' will be created with value 'CMURRAY'.

```
<html>
<head>
<script type="text/javascript">

function editUser() {
    var values = {'userId': 'CMURRAY'};
    oraRunScriptWithValues('UserEdit', values);
    return;
}

</script>
</head>
<body>
...
</body>
</html>
```

Exit UI Map with Bound Values

This function is used to exit a UI Map. When you quit the map you can specify a value to return to the script and, in addition, whether to return updated XML.

Syntax	Parameters	Comments
<code>oraSubmitMap();</code>	'Return Value' Boolean value	Indicates if the updated XML should be returned. Default is true .

In the following example, the Save button will return updated information, the Cancel button will not.

```
<html>
<body>
<table>
  <tr>
    <td/>
    <td style="padding-bottom:15px;">
      <a href="" onclick="oraShowErrorAlert(); return false;">
        <span oraErrorVar="ERRMSG-TEXT"></span></a>
      </td>
    </tr>
    <tr>
      <td >Address:</td>
      <td><input type="text" oraField="address"/></td>
    </tr>
    <tr>
      <td>City:</td>
```

```

    <td><input type="text" oraField="city"/></td>
  </tr>
  <tr>
    <td>State:</td>
    <td><input type="text" oraField="state"/></td>
  </tr>
  <tr>
    <td>Zip:</td>
    <td><input type="text" oraField="zip"/></td>
  </tr>
  <tr>
    <td/>
    <td style="padding-top:15px;">
      <input type="button" value="Save" onClick="oraSubmitMap('SAVE');"/>
      <input type="button" value="Cancel" onClick="oraSubmitMap('CANCEL',false);"/>
    </td>
  </tr>
</table>
</body>
<xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
  </root>
</xml>
</html>

```

Save and Cancel buttons rendered:

Include a Map Fragment

This function is used to embed a map fragment within another UI map. Note that it is possible to use the include node within a map or a map fragment.

Syntax	Parameters	Comments
<code><oralInclude map=' ' prefixPath=' '/></code>	map='UI Map Code' prefixPath='Xpath'	Optionally specify an xpath prefix to be appended onto every included oraField, oraLabel, oraList, oraSelect valuePath and descPath, oraDownloadData, and oraUploadData attribute value defined within the included UI map fragment's HTML. NOTE: This functionality only applies to XPath attribute values when those values do not appear beneath an oraList attribute. Any XPath value within a table containing

Syntax	Parameters	Comments
		an oraList attribute will not be affected by a prefixPath.

- An example of a map fragment with two buttons, named 'F1-SaveCancelButtons'.

```
<input onClick = "oraSubmitMap('SAVE');" oraMdLabel="SAVE_BTN_LBL" class="oraButton" type="button"/>
<input onClick = "oraSubmitMap('CANCEL',false);" oraMdLabel="CANCEL_LBL" class="oraButton" type="button"/>
```

- An example of a map that includes the map fragment named 'F1-SaveCancelButtons'.

```
...
<tr>
  <td colspan="2" align="center">
<oraInclude map="F1-SaveCancelButtons"/>
  </td>
</tr>
...
```

Include a Web Component

A web component is a custom HTML element that produces a specific user interface feature. Each supported web component typically has one or more defined HTML attributes that affect how the web component functions and how it appears. The more complex web components may have a corresponding system configuration object that defines the behavior and information displayed by the resulting user interface element. For example the web component that produces a "tree" of information has a Tree configuration table, which defines the details about how to retrieve information for the tree and how to display each node.

Common Attributes

Some common attributes that may be used for each web component are:

- A reference to the specific instance of the configuration object that governs the details to produce.
- The ability to pass in context data as name and value pairs.
- A 'mode' that support showing a preview of the feature

Some attributes support a **single value** that is referenced. For these types of attributes, the value may be provided using one of the following references:

- Literal value. The value can be referenced directly. Example:

```
treename="F1-Tree"
```

- XPath Reference. XPath references must use the explicit syntax **x[value]**. Example:

```
treename="x[migrationTree]"
```

- Global or Portal context reference. These references must use the explicit syntax **c[value]**.

```
treename="c[TREE_NAME]"
```

- Field label reference. These references must use the explicit syntax **md[value]**.

```
text=md[F1_ADD_LBL]"
```

- Field help text reference. These references must use the explicit syntax **mdh[value]**.

```
text=mdh[F1_OPERATIONS_LBL]"
```


For **complex attributes** that support one or more name / value pairs, the syntax for defining the names and values follows the same rules as above for literal values, XPath references and context. Colons (":") should be used to separate the name from the value and semi-colons (";") separate each pair.

```
context="FIELD_NAME1:x[value1];FIELD_NAME2:F1AB;x[fieldName3]:c[FIELD_NAME3];"
```

A given web component may have additional HTML attributes specific to its functionality.

NOTE: Any UI map that defines a web component must include the F1-OJETLIBSR map fragment within the "body" tag.

```
<body>
<oraInclude map="F1-OJETLIBSR" />
. .
</body>
```

NOTE: Standard HTML attributes may also be provided (e.g ID=".." and style=".."). These are defined as normal. It should be noted that the web components themselves may control some styling aspects directly and thus override any style settings provided on the HTML web component. For example, the Contextual Insights configuration provides settings for text color that would override any color defined in the HTML.

The following sections describe each supported web component.

Trees

Refer to [Trees](#) for more information about tree configuration.

Syntax	Parameters	Comments
<code><ou-tree...></ou-tree></code>	treename="singleValue"	You must supply a reference to the name of the tree object that defines the configuration of the tree to display. This attribute expects a single value that follows the syntax described in the single value section above.
	context="..."	Optionally pass in one or more name / value pairs. The syntax follows the example shown in the complex attributes information above.

The following example uses an explicit reference to a tree and does not pass any context. The particular tree relies on global context, which the tree node algorithm has access to.

```
<ou-tree treename="C1-PremiseTree"></ou-tree>
```

Contextual Insight

Refer to [Contextual Insights](#) for information about the configuration related to defining contextual insights.

Syntax	Parameters	Comments
<code><ou-insights ...></ou-insights></code>	insightType="singleValue"	Use this attribute to specify an insight type to display. Either this attribute or the insight class attribute must be supplied. This attribute expects a single value that follows the syntax described in the single value section above.
	insightClass="singleValue"	Use an insight class to display all the insight types for the insight group related to this class. This attribute expects a single value that follows the syntax described in the single value section above.

Syntax	Parameters	Comments
	context="..."	Optionally pass in one or more name / value pairs. The syntax follows the example shown in the complex attributes information above.
	mode="preview"	Optionally pass in a mode of preview to indicate to the underlying insight algorithm that prepares the data that the insight is being called in preview mode, with no context available. Ideally the algorithm returns sample information.
	class=" "	Optionally provide a CSS helper class to control aspects of the rendered insight. Supported values: class="card-full-width" - Applicable to card insights. Use this option class to ensure that the width for a card insight adjusts to the width of the zone that contains it. Card insights that don't set this parameter use a fixed width.

The following example is from the insight preview map where the insight type is the one in context. No context data is needed and the mode is 'preview'.

```
<ou-insights insightType="x[insightType]" mode="preview"></ou-insights>
```

The following example uses an insight class and passes in context values from XPath elements with explicitly defined context names.

```
<ou-insights insightClass="ClCI" context="PER_ID:x[personId];ACCT_ID:x[accountId];PREM_ID:x[promiseId];"></ou-insights>
```

Button

The button web component provides support for the standard set of button formats used within the system.

Syntax	Parameters	Comments
<ou-button...></ou-button>	type="singleValue"	You must define the type of button to display. This attribute expects a single value that follows the syntax described in the single value section above. The valid type values are: <ul style="list-style-type: none"> image - image only text - text only imageText - image followed by text menu - text plus a dropdown caret to open the menu imageMenu - image plus a dropdown caret to open the menu If no type attribute is supplied, a 'broken button' image will be displayed.
	img="singleValue"	Define the path of the image to display. Either this attribute or the image reference attribute

Syntax**Parameters****Comments**

		<p>must be supplied for a button type that includes an image. This attribute expects a single value that follows the syntax described in the single value section above.</p> <p>Note that for SVG images, if the reference includes only the SVG file name with no path, the path is assumed to be the Framework SVG repository.</p>
imgRef="singleValue"		<p>Define the display icon reference of the image to be used. Either this attribute or the image attribute must be supplied for a button type that includes an image. This attribute expects a single value that follows the syntax described in the single value section above.</p>
text="..."		<p>Define the various text elements needed by the button. The syntax follows the example shown in the complex attributes information above. The valid text element name values are:</p> <ul style="list-style-type: none">• label - the label text to be used for the button• tooltip - the hover text to be used for the button• aria-label - the ARIA label for the button
menu="singleValue"		<p>Define the menu to be used with the button. This attribute must be supplied for a button type that includes a menu control. This attribute expects a single value that follows the syntax described in the single value section above.</p>
disabled="singleValue"		<p>This is a standard HTML attribute. A value of "true" or just the presence of the text "disabled" will result in an initially disabled button.</p>
security="..."		<p>Optionally define the security settings for the button. The button will be hidden if the user does not have the appropriate security access. The syntax follows the example shown in the complex attributes information above. The valid security element name values are:</p> <ul style="list-style-type: none">• service - the application service whose security access governs access to the button• accessMode - the specific access mode for the service that determines a more granular level of access to the button

Syntax

Parameters

Comments

`clickAction="..."`

Optionally define the action to be taken when the button is 'clicked'. The syntax follows the example shown in the complex attributes information above. There are specific 'name' values for the attribute as follows:

- **action** - the action to be taken. Valid values are 'BPA' and 'NAVIGATE'
- **script** - the script to be performed when the action is 'BPA'
- **navopt** - the navigation option to be used when the action is 'NAVIGATE'

The attribute may also include other name/value pairs that define information to be passed into the script or used as navigation context fields.

In addition to the security attribute settings, the system will check that the user has access to the script or navigation path before performing the action.

The standard HTML 'onClick' attribute may still be used. If the HTML includes an 'onClick' attribute as well as the 'clickAction', the 'onClick' attribute will take precedence

The following is an example of a simple button with text.

```
<ou-button type="text" text="label:md[F1_BOOKMARK_LBL];"></ou-button>
```

The following is an example of an icon button using a display icon reference.

```
<ou-button type="image" imgRef="F1CMNT"></ou-button>
```

The following is an example of an image menu button.

```
<ou-button type="imageMenu" imgRef="F1STTGS" menu="CI_CONTEXTSCRIPT" text="tooltip:md[F1INSTY_VIS_LBL];"></ou-button>
```

The following is an example of text button that navigates to another page.

```
<ou-button type="text" text="label:md[ALG_CD];tooltip:md[GOTO_ALG]" clickAction="action:NAVIGATE; navopt:flalgqTa"></ou-button>
```

Show Schema Default on Add

Default values within in the UI map's schema will be displayed on a UI map's input fields if an embedded <action> node has a value of 'ADD' or blank.

Syntax

```
<action>ADD</action>
```

```
<action> </action>
```

The schema default for the <description> element will be displayed:

```
<schema>
  <action/>
  <boGroup type="group">
    <key/>
    <description default="enter description here"/>
  </boGroup>
</schema>

<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
  <tr>
    <td>Description </td>
    <td><input oraField="boGroup/description"/></td>
  </tr>
</table>
</body>
<xml>
<root>
  <action>ADD</action>
  <boGroup>
    <key/>
    <description/>
  </boGroup>
</root>
</xml>
</html>
```

HTML rendered.

Description	<input type="text" value="enter description here"/>
-------------	---

Configure a Chart

In addition to integrating charts with zones, the system supports using the same attributes within the UI Map HTML to configure a graphical representation of an XML list. The designer can control the type, size, position, and contents of the chart using these attributes. For details of the attributes and syntax, refer to [Data Explorer Graph Configuration](#).

Graph Examples

- Sample of a pie chart configuration:

```
<html>
<head>
<title>Pie Chart</title>
</head>
<body>

<div style="width:100%; height:290px;"
  oraChart="type:pie;"
  oraChartSeries1="list:set; labelPath:date; amount:amount; "
  oraChartBroadcast="BILL_ID:billId;">
</div>

</body>

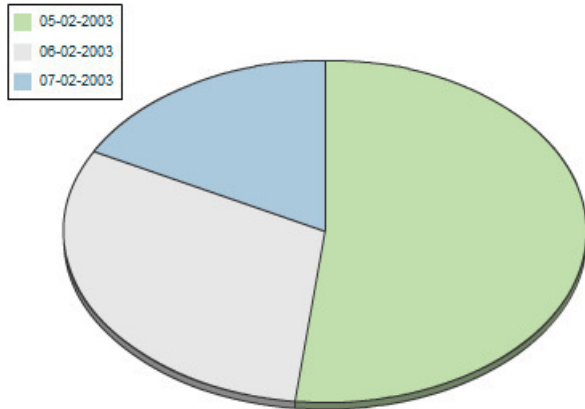
<xml>
<root>
  <set>
<date>05-02-2003</date>
<amount>163.24</amount>
<billId>592211649441</billId>
  </set>
  <set>
<date>06-02-2003</date>
<amount>97.29</amount>
<billId>592211649442</billId>
```

```

    </set>
    <set>
<date>07-02-2003</date>
<amount>54.38</amount>
<billId>592211649443</billId>
    </set>
</root>
</xml>
</html>

```

- A pie chart rendered for a single series:



- Sample of a line, cluster, or stacked graph configuration - each with two series:

```

<html>
<head>
<title>Stacked Chart</title>
</head>
<body>

<div style="width:100%; height:300px;"
  oraChart="type:line;"
  oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount; "
  oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance; "
  oraChartBroadcast="BILL_ID:billId;">
</div>

<div style="width:100%; height:300px;"
  oraChart="type:cluster;"
  oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount; "
  oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance; "
  oraChartBroadcast="BILL_ID:billId;">
</div>

<div style="width:100%; height:300px;"
  oraChart="type:stacked;"
  oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount; "
  oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance; "
  oraChartBroadcast="BILL_ID:billId;">
</div>

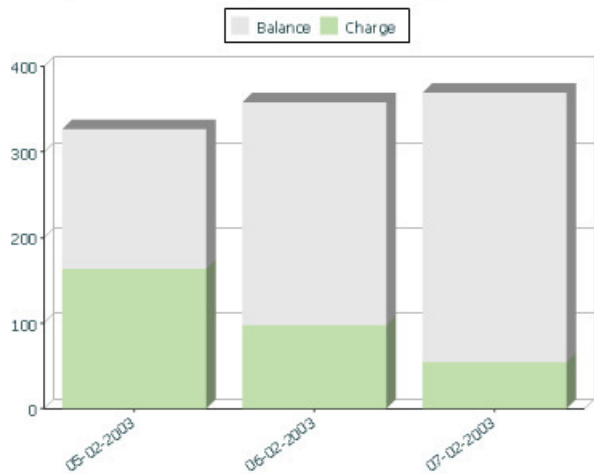
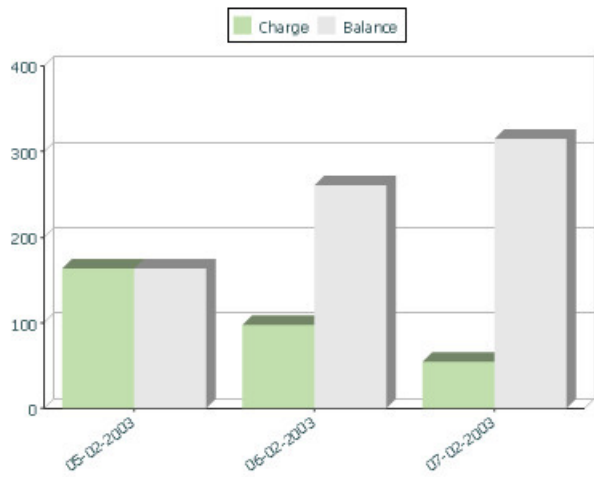
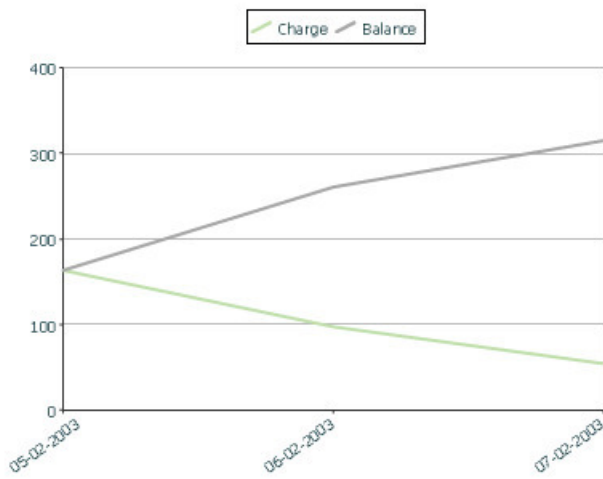
</body>

<xml>
<root>
  <set>
<date>05-02-2003</date>
<amount>163.24</amount>
<balance>163.24</balance>
<billId>592211649441</billId>
  </set>
  <set>
<date>06-02-2003</date>

```

```
<amount>97.29</amount>
<balance>260.53</balance>
<billId>592211649442</billId>
  </set>
  <set>
<date>07-02-2003</date>
<amount>54.38</amount>
<balance>314.91</balance>
<billId>592211649443</billId>
  </set>
</root>
</xml>
</html>
```

- Three types of chart rendered for two series each: line, cluster, and stacked.



Upload and Download a CSV File

The following HTML attributes can be used to manage both an upload and a download between a list defined within the map's schema and a CSV (comma separated value) file. Note that this technique is only recommended for a small to medium volume of data, for example no more than several hundred rows. For higher volumes, it is recommended to use batch upload / download functionality instead.

The syntax is `oraUploadData="type:embed;path:list xpath;useLabels:true;showCount:true"`

Upload configuration requires you to name a CSV file to be uploaded, and an XML list as target. By convention, each CSV row will create a separate list instance. Each comma-separated field in the file will be uploaded as a separate element in the list. To embed an upload dialog within a map, the **oraUploadData** attribute must be associated with a container element such as a div, td, or span.

The optional **useLabels:true** value indicates that while parsing the upload CSV file, the headers are expected to be labels

NOTE: If you do not specify the **useLabels:true** value and the XML target element name is "camelCase" then the corresponding spreadsheet header should be title case with a space between words, e.g.;"Camel Case". Letters and special characters are not considered a different word, for example Address1 must be uploaded into the target XML element address1.

Specifying the optional **showCount:true** value will display the number of records uploaded.

CAUTION: If you are using a grid in conjunction with the **oraUploadData** function, then you must maintain the grid's list with a 'replace' business object action. Refer to [BO Replace Action](#) for more information.

Sample of **oraUploadData="embed"** within a div element.

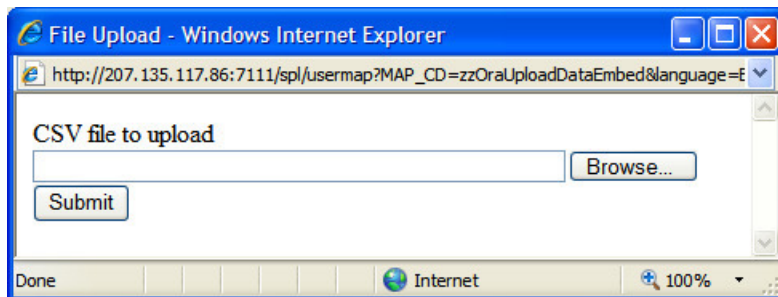
```
<html>
<head>
  <title>File Upload</title>
</head>
<body>

  <div oraUploadData="type:embed;path:myList" > </div>

</body>

<xml>
<root>
  <myList>
<id>838383930</id>
  <name>Janice Smith</name>
  </myList>
  <myList>
<id>737773730</id>
  <name>Bill McCollum</name>
  </myList>
</root>
</xml>
</html>
```

This file upload dialog will be embedded into the body of the page where the oraUploadData is defined.



oraUploadData="type:popup;path:list xpath;useLabels:true;showOk:true;showCount:true"

Upload configuration requires you to name a CSV file to be uploaded, and an XML list as target. By convention, each CSV row will create a separate list instance. Each comma-separated field in the file will be uploaded as a separate element in the list. To upload a CSV file using a pop-up dialog, the oraUploadData attribute must be associated with an input element such as a button, text link, or image.

The optional useLabels:true value is used to indicate that while parsing the upload CSV file, the headers are expected to be labels

NOTE: If you do not specify the useLabels:true value and the XML target element name is "camelCase" then the corresponding spreadsheet header should be title case with a space between words, e.g., "Camel Case". Letters and special characters are not considered a different word, for example Address1 must be uploaded into the target XML element address1.

Specifying the optional showOk:true value will display an "Ok" button once the upload finishes. The popup will stay open until the button is pressed. Additionally, specifying the showCount:true value will display number of records uploaded.

CAUTION: If you are using a grid in conjunction with the **oraUploadData** function, then you must maintain the grid's list with a 'replace' business object action. Refer to [BO Replace Action](#) for more information.

Sample of oraUploadData="popup" associated with a button:

```
<html>
<head>
  <title>File Upload</title>
</head>
<body>
  <input type="button" name="submitButton" oraUploadData="type:popup;path:myList;" value='Get Data'>
  <table oraList="myList">
    <tr/>
    <tr>
<td><span oraField="id"/></td>
<td><span oraField="name"/></td>
    </tr>
  </table>
</body>
<xml>
<root>
  <myList>
<id>838383930</id>
  <name>Janice Smith</name>
  </myList>
  <myList>
<id>737773730</id>
  <name>Bill McCollum</name>
  </myList>
</root>
</xml>
</html>
```

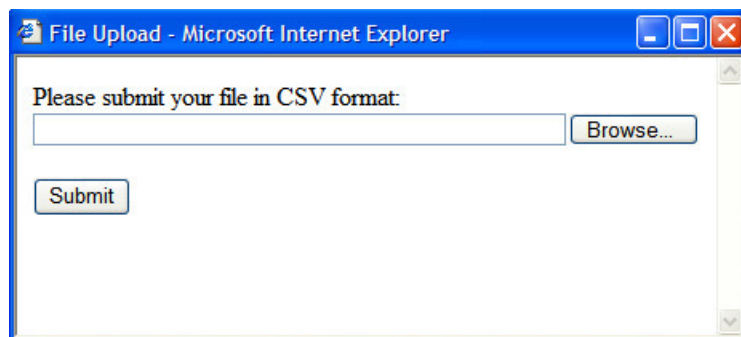
HTML Rendered:

Get Data

838383930 Janice Smith

737773730 Bill McCollum

Pressing the "Get Data" button will launch a standard file upload dialogue (provided by Framework) as shown below.



oraDownloadData="list xpath"

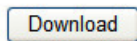
Download configuration requires you to name an XML list to be downloaded. By convention, each list instance will represent a separate row in the created file. By default every element of the list will be comma separated in the file.

NOTE: The number formatting is based on the user profile setting. For localities where the decimal symbol is a comma, an implementation may configure a property setting (`spl.csv.delimiter.useFromDisplayProfile=true`) to cause the system to use a semicolon as the delimiter that separates the elements rather than a comma.

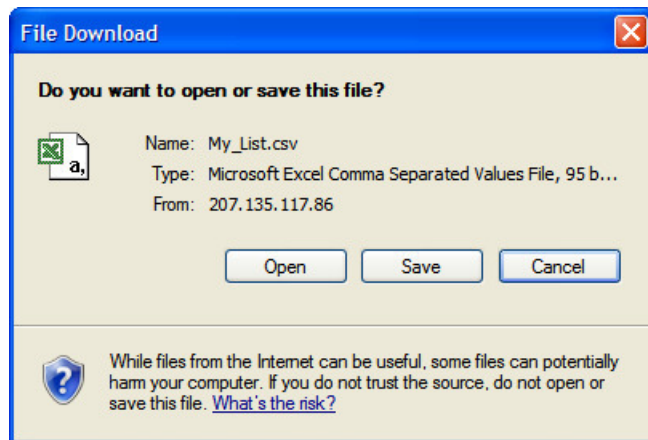
Sample of oraDownloadData.

```
<html>
<head>
<title>File Download</title></head>
<body>
<input type="button" name="downloadButton" oraDownloadData="myList" value="Download" />
</body>
<xml>
<root>
  <myList>
    <id>881-990987</id>
    <name>John Mayweather</name>
  </myList>
  <myList>
    <id>229-765467</id>
    <name>Anna Mayweather</name>
  </myList>
  <myList>
    <id>943-890432</id>
    <name>Andrew Brewster</name>
  </myList>
</root>
</xml>
</html>
```

HTML Rendered:



Pressing the "Download" button will launch a standard file download dialogue (provided by Framework) as shown below.



A successful download will result in a CSV file:

	A	B	C
1			
2	Id	Name	
3	881-99098	John Mayweather	
4	229-76546	Anna Mayweather	
5	943-89043	Andrew Brewster	
6			

To download data from a sub list use the attribute `oraDownloadDataInList` instead of `oraDownloadData`. The attribute `oraDownloadDataInList` will have the sub list name. The XPath of the sub list is used to pick data of the specific row from the parent list. Thus only the specific sub list is downloaded.

`oraDownloadDataUseLabels="true"`

The `oraDownloadDataUseLabels` attribute can be used in conjunction with the `oraDownloadData` attribute described above. Specify `oraDownloadDataUseLabels` if you want the generated CSV file to use the element labels for columns headers rather than element names.

Construct Portal Zone Map Fragments

Portal zones can reference a UI map for the zone header and filter area. This UI map is not a complete HTML document, but is instead configured as a UI Map fragment. When constructing a zone map fragment you can reference the following substitution variables. Note that these variables will be dynamically populated at run time with information particular to the map's zone within the portal:

Variable	Replacement Logic
<code>[\$ZONEDESCRIPTION]</code>	Zone's description text.
<code>[\$SEQUENCEID]</code>	Zone's sequence ID.
<code>[\$ZONENAME]</code>	Zone's name.
<code>[\$HELPTEXT]</code>	Zone's help text.
<code>[\$ZONEPARAMNAME]</code>	Zone parameter's value (or blank if it has not been specified).

WARNING:

- Refer to one of the following maps as examples: `F1-UIMapHeader` and `F1-ExplorerHeader`.
- These maps make use of the `oraInclude` tag to incorporate HTML fragments for the header menu and framework actions. Refer to the zone type parameters for the UI Map fragments you should include in your HTML.
- If you wish to have the "help text" icon appear next to your zone description, you should have `id="title_[$SEQUENCEID]"` on the `<td>` that contains your description.
- If it is necessary to encapsulate JavaScript within a UI Map fragment, it will be necessary to bound the JavaScript within a `![CDATA[]]` tag to ensure a valid XML document. Note that the tags themselves may need to be commented out to promote compatibility with older browsers. For example:

```
<script type="text/javascript">
/*![CDATA[ */
//
//javascript
//
/*]]> */
</script>
```

NOTE: If you wish to preserve the values of a filter input field, within a filter map fragment, for the framework 'Go Back' and 'Go Forward' functionality, you must associate the input field (text box, select, etc.) with a unique HTML id. Input field values associated with a unique id will be captured in the framework's 'memento'. The 'memento' is used to rebuild the input map when the portal zone is navigated to using the 'Go Back' or 'Go Forward' functionality.

NOTE: Many specialized functions exist to manipulate zone behavior, for example:

- **oraGetZoneSequence(zoneName).** Uses the zone's code to retrieve its sequence number.
- **oraIsZoneCollapsed(sequenceId).** Uses the zone's sequence to determine if collapsed.
- **oraHandleCollapse(seq).** Collapse a zone.
- **oraHandleExpand(seq,refresh).** Expand and/or refresh a zone.

All of these, and many more functions, are located within the JavaScript library [userMapSupport.js](#) described below.

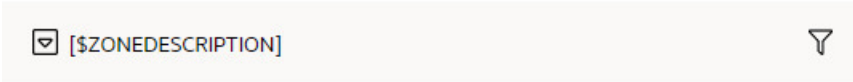
NOTE: When executing oraLoad within a fragment UI map, and you need to execute a JavaScript function during page load (where the function invokes a business object, business service, or service script) you can use the special syntax "oraLoad[\$SEQUENCEID]". Refer to the [Load Page Event](#) section for more information.

Example of oraLoad[\$SEQUENCEID] used within a function:

```
<script type="text/javascript">
function oraLoad[$SEQUENCEID]() {
checkRebateClaimStatus();
}

function checkRebateClaimStatus() {
    var work = id('analyticsFilterText[$SEQUENCEID]',
document).cells[0].innerText.split(' ');
    var rebateClaimId = work[work.length - 3];
    id('rebateClaimId', document).value = rebateClaimId;
oraInvokeSS('C1-CheckRCSt','checkRebateClaimStatus', false);
    var statusIndicator = id('statusInd', document).value;
    if (statusIndicator == 'C1PE' || statusIndicator == 'C1ID') {
        id('addRebateClaimLine', document).style.display = '';
    } else {
id('addRebateClaimLine', document).style.display = 'none';
    }
}
</script>
```

F1-ExplorerHeader rendered:



Invoking a Business Object

The oraInvokeBO function may be used within a portal zone header or zone filter map. It is similar to the command described in [Invoke BO Function](#) which allows for a business object to be invoked within the UI map's HTML. Refer to that section for a description of the first three parameters.

Syntax	Parameters	Comments
oraInvokeBO()	'BO Name'	
	'XPath' or null	
	'action'	
	null	This must be specified as the fourth argument.

Syntax	Parameters	Comments
	[\$SEQUENCEID]	This must be specified as the fifth argument.
	true false	Specify true if the fragment is used within a portal zone header. Specify false if the fragment is used with a zone filter map.

Example in a portal zone header:

```
oraInvokeBO('CM-User','xmlGroup','read', null, [$SEQUENCEID], true)
```

Invoking a Business Service

The oraInvokeBS function may be used within a portal zone header or zone filter map. It is similar to the command described in [Invoke BS Function](#) which allows for a business service to be invoked within the UI map's HTML. Refer to that section for a description of the first two parameters.

Syntax	Parameters	Comments
oraInvokeBS()	'BO Name'	
	'XPath' or null	
	null	This must be specified as the fourth argument.
	[\$SEQUENCEID]	This must be specified as the fifth argument.
	true false	Specify true if the fragment is used within a portal zone header. Specify false if the fragment is used with a zone filter map.

Example in a portal zone header:

```
oraInvokeBS('CM-UserSearch','xmlGroup', null, [$SEQUENCEID], true)
```

Invoking a Service Script

The oraInvokeSS function may be used within a portal zone header or zone filter map. It is similar to the command described in [Invoke SS Function](#) which allows for a service script to be invoked within the UI map's HTML. Refer to that section for a description of the first two parameters.

Syntax	Parameters	Comments
oraInvokeSS()	'Service Script Name'	
	'XPath' or null	
	null	This must be specified as the fourth argument.
	[\$SEQUENCEID]	This must be specified as the fifth argument.
	true false	Specify true if the fragment is used within a portal zone header. Specify false if the fragment is used with a zone filter map.

Example in a portal zone header:

```
oraInvokeSS('UserSearch','xmlGroup', null, [$SEQUENCEID], true)
```

Detecting Unsaved Changes

Use this function to return a Boolean set to true if there are unsaved changes. The system will interrogate the function when the user attempts to navigate and issue a warning accordingly. This function is only needed if a UI map is using custom javascript to manage elements such that the system is not able to detect whether changes have been made. Also note that it's the responsibility of the UI map javascript to manage the values in the Boolean used for this function.

```
function hasUnsavedChanges(){
    return isDirtyFlag;
}
```

Hiding Portal Tabs

The product provides the ability to use JavaScript to hide a tab on the current portal based on some condition using the oraAuthorizeTab JavaScript API. This API accepts a function as a parameter and turns off the tab index indicated.

For example, the UI Map may have a function to turn off one or more tab indexes.:

```
function overrideTabIndex(index){
    if (index == 2) return false;
    if (index == 3) return false;
}
```

The JavaScript is referenced "on load":

```
<body class="oraZoneMap"
onLoad="oraAuthorizeTabs(overrideTabIndex);">
```

Required JavaScript Libraries

All of the functionality described in this document depends on a pair of JavaScript libraries. If you are writing and executing your maps entirely within the UI map rendering framework - you do not need to manually insert the following libraries - the framework will insert them for you when the UI Map is rendered.

WARNING: When executing HTML outside of the framework you must include the following references explicitly within your HTML. In addition, the tool you use to render the HTML must have access to a physical copy of privateUserMapSupport.js for bind support.

```
src="privateUserMapSupport.js"
```

Your HTML document must reference this library to execute binding in a stand-alone environment.

WARNING: Referencing functions within this JavaScript library is dangerous - because these functions are owned by framework and they may be changed during version upgrade or via the normal patch process.

```
<script type="text/javascript" src="privateUserMapSupport.js"></script>
```

```
src="userMapSupport.js"
```

To take advantage of optional toolset features, you must reference this library.

NOTE: You can reference the functions within this JavaScript library to write custom functions within the UI map..

```
<script type="text/javascript" src="userMapSupport.js"></script>
```

```
onload="oraInitializeUserMap();"
```

To execute binding in a stand-alone environment, you must embed the following onload function into the <body> node.

```
<body onload="oraInitializeUserMap();">
```

UI Map Standards

Contents

- [Basic UI Map Templates](#)
- [Basic HTML and Styles](#)
- [Grids \(Tables of Data\)](#)
- [Action Buttons](#)
- [Available Styles](#)
- [Using OJET](#)

Basic UI Map Templates

All UI Maps share the same basic structure regardless of placement (page area, zone, pop-up) or usage (display only, input).

Sample XML

All information in this document is based upon the following XML structure.

```
<xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
    <contactInformation>
      <type>Home Phone</type>
      <number>510-555-2287</number>
    </contactInformation>
    <contactInformation>
      <type>Cell Phone</type>
      <number>510-555-4285</number>
    </contactInformation>
  </root>
</xml>
```

Display Only UI Map

```
<html>
<head>
  <title oraMdLabel="ADDRESS_LBL"></title>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body class="oraZoneMap">
<table cellpadding="4" cellspacing="4" width="100%">
  <colgroup>
    <col class="oraLabel oraTableLabel" />
    <col class="oraNormal oraTableData" />
  </colgroup>
  <tr>
    <td oraLabel="address"></td>
    <td oraField="address"></td>
  </tr>
  <tr>
    <td oraLabel="city"></td>
    <td oraField="city"></td>
  </tr>
  <tr>
    <td oraLabel="state"></td>
    <td oraField="state"></td>
  </tr>
  <tr>
    <td class="oraSectionEnd" oraLabel="zip"></td>
    <td class="oraSectionEnd" oraField="zip"></td>
  </tr>
```



```

<tr>
  <td colspan="2" class="oraSectionHeader" oraMdLabel="CONTACT_LBL"></td>
</tr>
<tr>
  <td colspan="2" class="oraSectionStart oraEmbeddedTable">
    <table oraList="contactInformation" cellspacing="2">
      <thead >
        <tr>
          <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/type"></span>
          </th>
          <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/number"></span>
          </th>
        </tr>
      </thead >
      <tbody>
        <tr>
          <td class="oraNormalAlt oraDisplayCell">
            <span oraField="type"></span>
          </td>
          <td class="oraNormal oraDisplayCell">
            <span oraField="number"></span>
          </td>
        </tr>
      </tbody>
    </table>
  </td>
</tr>
</table>
</body>
</xml>
  <root>
    <address>123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
    <contactInformation>
      <type>Home Phone</type>
      <number>510-555-2287</number>
    </contactInformation>
    <contactInformation>
      <type>Cell Phone</type>
      <number>510-555-4285</number>
    </contactInformation>
  </root>
</xml>
</html>

```

Input UI Map

```

<html>
<head>
  <title oraMdLabel="ADDRESS_LBL"></title>
  <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
  <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td><a href="" onclick="oraShowErrorAlert(); return false;">
      <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span></a>
    </td>
  </tr>
</table>
<table cellspacing="4" width="100%">
  <colgroup>
    <col class="oraLabel oraTableLabel" />
    <col class="oraNormal oraTableData" />
  </colgroup>
  <tr>
    <td oraLabel="address"></td>

```

```

        <td><input type="text" oraField="address"/></td>
</tr>
<tr>
    <td oraLabel="city"></td>
    <td><input type="text" oraField="city"/></td>
</tr>
<tr>
    <td oraLabel="state"></td>
    <td><input type="text" oraField="state"/></td>
</tr>
<tr>
    <td oraLabel="zip"></td>
    <td><input type="text" oraField="zip"/></td>
</tr>
<tr>
    <td colspan="2" class="oraSectionHeader" oraMdLabel="CONTACT_LBL"></td>
</tr>
<tr>
    <td colspan="2" class="oraSectionStart oraEmbeddedTable">
        <table oraList="contactInformation" cellspacing="2">
            <thead >
                <tr>
                    <th class="oraGridColumnHeaderButton"></th>
                    <th class="oraGridColumnHeaderButton"></th>
                    <th class="oraGridColumnHeader" nowrap="nowrap">
                        <span oraLabel="contactInformation/type"></span>
                    </th>
                    <th class="oraGridColumnHeader" nowrap="nowrap">
                        <span oraLabel="contactInformation/number"></span>
                    </th>
                </tr>
            </thead >
            <tbody>
                <tr>
                    <td oraType="addGridRow"></td>
                    <td oraType="deleteGridRow"></td>
                    <td>
                        <input type="text" oraField="type"/>
                    </td>
                    <td>
                        <input type="text" oraField="number"/>
                    </td>
                </tr>
            </tbody>
        </table>
    </td>
</tr>
<tr>
    <td colspan="2" class="oraSectionStart oraEmbeddedTable">
        <table cellspacing="2">
            <tr>
                <td>
                    <input class="oraButton" oraMdLabel="Cl_SAVE_LBL" type="button"
                        onClick="oraSubmitMap('OK');"/>
                </td>
                <td>
                    <input class="oraButton" oraMdLabel="CANCEL_LBL" type="button"
                        onClick="oraSubmitMap('CANCEL',false);"/>
                </td>
            </tr>
        </table>
    </td>
</tr>
</table>
</body>
<xml>
    <root>
        <address>123 Main St</address>
        <city>Alameda</city>
        <state>CA</state>
        <zip>94770</zip>
        <contactInformation>

```

```
        <type>Home Phone</type>
        <number>510-555-2287</number>
    </contactInformation>
    <contactInformation>
        <type>Cell Phone</type>
        <number>510-555-4285</number>
    </contactInformation>
</root>
</xml>
</html>
```

Basic HTML and Styles

The basic templates introduced the standard HTML and styles used for UI Maps. These standards are described individually in the following sections.

Stylesheets

The styles to apply the standard look to the maps are all contained in stylesheets. These stylesheets should be included in all UI Maps.

```
...
<link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
<link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
...
```

Title

Each UI Map should have a <title> tag.

```
...
<title oraMdLabel="ADDRESS_LBL"></title>
...
```

This will give the UI Map a descriptive title.

- If the UI Map is presented in a "pop-up", the title will be in the window title bar.
- If the UI Map is presented in the page area, the title will be added as a tag to the UI Map and will appear at the top of the UI Map.
- If the UI Map is presented as a zone map, it will be ignored. The <title> tag should still be included in the HTML as standard.

Zone Maps

When the map is presented in a zone as part of a portal, the UI Map should have a border so that the information is "contained" within the zone.

```
...
<body class="oraZoneMap">
...
```

Page Area Maps vs Pop-Up Maps

The presentation of the UI Maps can vary from design to design. The following standards have been applied to decide when to use a Page Area UI Map and when to use a Pop-Up Map:

- If there are multiple UI Maps in the sequence, always use the Page Area.
- If the UI Map has many input fields, always use a Page Area.
- If the UI Map is a "confirmation" type dialog or only has one or two input fields, use a Pop-Up.

NOTE: The difference between "just a few input fields" and "many input fields" can be discretionary. The final decision should rest with the dialog designer.

Error Messages

Input maps have a ability to present error messages to the User.

```
...
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td><a href="" onclick="oraShowErrorAlert(); return false;">
      <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span></a></td>
    </tr>
  </table>
...
```

This HTML structure provides the necessary elements and functions to display errors to the User. It should be directly after the <body> tag. When there is no error, nothing will be visible on the UI Map. It will be made visible if an error occurs and the UI Map is re-presented to the User. Clicking on the link (when visible) will result in a pop-up alert appearing with the long error message text.

Standard Layout and Styles

The information is presented on the UI Map by using a <table> to organize the information in rows and columns.

```
...
<table cellspacing="4" width="100%">
<colgroup>
<col class="oraLabel oraTableLabel" />
  <col class="oraNormal oraTableData" />
</colgroup>
...
```

The <colgroup> and <col> tags allow for the application of classes to the columns (the label is in the first column and the data is in the second column.). Using these tags mean that the class attribute (to apply styles) does not need to be defined on every <td>.

Grids (Tables of Data)

A UI Map could contain information that is best presented as a grid. These are referred to as "Embedded Tables". The embedded table can be used to display information or input information.

Example Embedded Table HTML

The embedded table will be included within a row (<tr>) of the basic layout:

```
...
<tr>
  <td colspan="2" class="oraEmbeddedTable">
    <table oraList="contactInformation" cellspacing="2">
      <thead >
        <tr>
          <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/type"></span>
          </th>
          <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/number"></span>
          </th>
        </tr>
      </thead >
      <tbody>
        <tr>
          <td class="oraNormalAlt oraDisplayCell">
            <span oraField="type"></span>
          </td>
          <td class="oraNormal oraDisplayCell">
            <span oraField="number"></span>
          </td>
        </tr>
      </tbody>
    </table>
  </td>
</tr>
```

```

    </td>
</tr>
...
<xml>
  <root>
    <address> 123 Main St</address>
    <city>Alameda</city>
    <state>CA</state>
    <zip>94770</zip>
    <contactInformation>
      <type>Home Phone</type>
      <number>510-555-2287</number>
    </contactInformation>
    <contactInformation>
      <type>Cell Phone</type>
      <number>510-555-4285</number>
    </contactInformation>
  </root>
</xml>

```

Embedding the Table

The embedded table is included within the overall table structure. The `colspan` attribute ensures that the embedded table can span the standard two columns of the overall layout table.

```

...
<tr>
  <td colspan="2" class="oraEmbeddedTable">
    ...
    ...
    ...
  </td>
</tr>
...

```

Embedded Table Structure

The embedded table is very similar to the basic layout table.

```

...
<table oraList="contactInformation" cellspacing="2">
<thead>
  ...
  ...
</thead>
<tbody>
  ...
  ...
</tbody>
</table>
...

```

- The `<table>` tag has a slightly smaller cellspacing and it defines the "list" element contained in the XML that will be used to provide the data.
- The `<thead>` element is used to give the embedded table headings for the columns.
- The `<tbody>` element is the element that will be repeated for each referenced "list" element in the XML. In the previous example, there are two "contactInformation" list elements, so the displayed embedded table will have two rows.

Column Headings

Embedded tables should have headings for the displayed columns. The `<thead>` tag defines these.

```

...
<thead>
  <tr>
    <th class="oraGridColumnHeader" nowrap="nowrap">
      <span oraLabel="contactInformation/type"></span>
    </th>

```

```

        <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/number"></span>
        </th>
    </tr>
</thead>
...

```

The "nowrap" attribute prevent the column heading from taking multiples lines. If multiples lines are required, the "nowrap" may be removed.

Input Fields

Embedded tables may be used for input as well as display only. The framework provides a convenient control to assist in the creation of editable embedded tables.

```

...
<tr>
    <td colspan="2" class="oraEmbeddedTable">
        <table oraList="contactInformation" cellspacing="2">
            <thead >
                <tr>
                    <th class="oraGridColumnHeaderButton"></th>
                    <th class="oraGridColumnHeaderButton"></th>
                    <th class="oraGridColumnHeader" nowrap="nowrap">
                        <span oraLabel="contactInformation/type"></span>
                    </th>
                    <th class="oraGridColumnHeader" nowrap="nowrap">
                        <span oraLabel="contactInformation/number"></span>
                    </th>
                </tr>
            </thead >
            <tbody>
                <tr>
                    <td oraType="addGridRow"></td>
                    <td oraType="deleteGridRow"></td>
                    <td>
                        <input type="text" oraField="type" />
                    </td>
                    <td>
                        <input type="text" oraField="number" />
                    </td>
                </tr>
            </tbody>
        </table>
    </td>
</tr>
...

```

There are two new columns added to the input embedded table.

- oraType="addGridRow" will add a "+" button to the row. This will allow the User to add an additional row to the existing grid.
- oraType="deleteGridRow" will add a "-" button to the row. This will allow the User to delete an existing row from the grid.

NOTE: The <thead> tag also requires these two new columns to be added.

These controls are, as standard, placed at the beginning of the row in the order shown. Either of the controls may be omitted if required (if, for example, Users are not permitted to delete information).

The presence of either of these controls will activate the "empty list" process. This means that if the XML has no data for the "list" specified, the input grid will display with an empty row ready for the input of new information.

Action Buttons

Example Action Button HTML

Action buttons are used to perform some specified function from the UI Map. The actions are as varied as the information being displayed/updated. Below are two common examples:

- Save. Normally used on an Input UI Map to allow a User to save any changes they have made.
- Cancel. Normally used on an Input UI Map to allow a User to cancel changes in progress.

```
...
<tr>
  <td colspan="2" class=" oraEmbeddedTable">
    <table cellspacing="2">
      <tr>
        <td>
          <input class="oraButton" oraMdLabel="C1_SAVE_LBL" type="button"
            onClick="oraSubmitMap( 'OK' );"/>
        </td>
        <td>
          <input class="oraButton" oraMdLabel="CANCEL_LBL" type="button"
            onClick="oraSubmitMap( 'CANCEL' ,false);"/>
        </td>
      </tr>
    </table>
  </td>
</tr>
...
```

Button Standards

The following points highlight some standards related to buttons.

- Buttons are included as an embedded table.
- Buttons should be grouped together. They should not be placed in different areas of the UI Map.
- The location of the buttons depends mainly on the type of UI Map.
 - Display Only UI Maps should have a Record Actions section in the upper right section of the UI map.
 - Input UI Maps should have the buttons at the foot of the UI Map (after all input fields).

Available Styles

Styles are all contained in the referenced CSS stylesheets. They are applied by the HTML "class" attribute. The actual style settings used are not documented here as they may be adjusted. This section only specifies when a particular style should be used.

NOTE: The "class" attribute may reference more than one style (class="oraLabel oraSectionEnd")

Style	Comments	Example
oraButton	Applied to <input> elements where the type is button.	... <input class="oraButton" oraMdLabel="CANCEL_LBL" type="button" onClick="oraSubmitMap('CANCEL' ,false);"/> ...
oraDisplayCell	Applied to the <td> tag of an embedded table. It defines how the table cell looks (not the data contained inside the cell).	... <td class="oraDisplayCell"> </td>

Style	Comments	Example
oraEmbeddedTable	Applied to the <td> tag that will contain the embedded table.	<pre> ... <tr> <td colspan="2" class=" oraEmbeddedTable"> <table cellpadding="2"> </table> </td> </tr> ... </pre>
oraError	<p>This style is applied to elements that are identified as "error elements". Refer to Display Errors for more information.</p> <p>NOTE: This style is not normally applied directly in the UI Map HTML</p>	
oraErrorText	This style is applied to the elements concerned with error messages.	<pre> ... <table width="100%" cellpadding="12"> <tr class="oraErrorText"> <td> </td> </tr> </table> ... </pre>
oraGridColumnHeader	This style is applied to the <td> tags that define column headers within embedded table.	<pre> ... <thead > <tr> <th class="oraGridColumnHeaderButton"></th> <th class="oraGridColumnHeaderButton"></th> <th class="oraGridColumnHeader" nowrap="nowrap"> </th> <th class="oraGridColumnHeader" nowrap="nowrap"> </th> </tr> </thead > ... </pre>
oraGridColumnHeaderButton	This style is applied to the <td> tags that define the column headers for the "+" and "-" buttons used on editable embedded tables.	<pre> ... <thead > <tr> <th class="oraGridColumnHeaderButton"> </th> <th class="oraGridColumnHeaderButton"> </th> <th class="oraGridColumnHeader" nowrap="nowrap"> </th> <th class="oraGridColumnHeader" nowrap="nowrap"> </th> </tr> </thead > ... </pre>
oraInput	<p>This style is applied to input fields:</p> <ul style="list-style-type: none"> <input type="text"> 	<pre> ... <input type="text" class="oraInput" oraField="address" /> ... </pre>

Style	Comments	Example
	<ul style="list-style-type: none"> • <input type="checkbox"> • <select> • <textarea> <p>NOTE: This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".</p>	
oraInputMoney	<p>This style is applied to input fields:</p> <ul style="list-style-type: none"> • <input type="text"> • <select> (rare) • <textarea> (not recommended) <p>NOTE: This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".</p>	<pre>... <input type="text" class="oraInputMoney" oraField="amount" /> ...</pre>
oraInputNumber	<p>This style is applied to input fields:</p> <ul style="list-style-type: none"> • <input type="text"> • <select> (rare) • <textarea> (not recommended) <p>NOTE: This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".</p>	<pre>... <input type="text" class="oraInputNumber" oraField="count" /> ...</pre>
oraLabel	<p>This style is applied to standard label fields that are right aligned.</p> <p>NOTE: This can normally be omitted as it is applied by the <col> tag.</p>	<pre>... <td class="oraLabel" oraLabel="address"></td> ...</pre>
oraLabelAlt	<p>This style is applied to standard label fields only if it is desired to have the label aligned to the left.</p>	<pre>... <td class="oraLabelAlt" oraLabel="address"></td> ...</pre>
oraLabelCenter	<p>This style is applied to standard label fields only if it is desired to have the label aligned in the center of the cell.</p>	<pre>... <td class="oraLabelCenter" oraLabel="address"></td> ...</pre>
oraLink	<p>This style is applied to foreign key references (links). This is automatically added by the UI Map framework but can also be used manually if desired.</p>	<pre>... <td class="oraLabel"> Google</td> ...</pre>
oraNormal	<p>This style is applied to standard data fields (display only) that are left aligned.</p>	<pre>... <td class="oraNormal" oraField="address"></td> ...</pre>

Style	Comments	Example
	<p>NOTE: This can normally be omitted as it is applied by the <col> tag.</p>	
oraNormalAlt	This style is applied to standard data fields (display only) only if it is desired to have the data aligned to the right.	<pre>... <td class=" oraNormalAlt" oraField="address"></td> ...</pre>
oraNormalCenter	This style is applied to standard data fields (display only) only if it is desired to have the data aligned in the center of the cell.	<pre>... <td class=" oraNormalCenter" oraField="address"></td> ...</pre>
oraPageTitle	<p>This style is applied to the element that contains the page title.</p> <p>NOTE: This style is not normally applied directly in the UI Map HTML. The is created by the UI Map framework when the UI Map is displayed in the page area.</p>	<pre>... ...</pre>
oraSectionEnd	<p>This style is applied to the <td> tags at the end of a "section" (group of elements). It provides some space to separate the section from the following information.</p> <p>NOTE: The style must be applied to both <td> tags or the label may be misaligned with the data/input.</p>	<pre>... <tr> <td class="oraSectionEnd" oraLabel="zip"> </td> <td class="oraSectionEnd" oraField="zip"> </td> </tr> ...</pre>
oraSectionHeader	<p>This style is applied to the <td> tag used to give a heading for a section within the information being displayed. It does not provide spacing before or after itself. The oraSectionStart and oraSectionEnd classes are used for this.</p> <p>NOTE: The section header should span both the label column and the data column.</p>	<pre>... <tr> <td class="oraSectionHeader" colspan="2" oraMdField="DATA_SECTION_LBL"></td> </tr> ...</pre>
oraSectionStart	<p>This style is applied to the <td> tags at the start of a "section" (group of elements). It provides some space to separate the section from the previous information (often a section header).</p> <p>NOTE: The style must be applied to both <td> tags or the label may be misaligned with the data/input.</p>	<pre>... <tr> <td class="oraSectionStart" oraLabel="zip"></td> <td class="oraSectionStart" oraField="zip"></td> </tr> ...</pre>
oraTableData	This style is applied to the <col> tag for the data column of the main table (second column). It is used to provide a percentage width for the horizontal space to be used for the information.	<pre>... <colgroup> <col class="oraLabel" oraTableLabel"/> <col class="oraNormal" oraTableData"/> </colgroup> ...</pre>
oraTableLabel	This style is applied to the <col> tag for the label column of the main table (first column). It is used	<pre>... <colgroup> <col class="oraLabel" oraTableLabel" /> ...</pre>

Style	Comments	Example
	to provide a percentage width for the horizontal space to be used for the labels.	<pre><col class="oraNormal oraTableData" /> </colgroup> ...</pre>
oraTinyText	This style is typically applied directly beneath an <input> tag to provide information or a hint to the user concerning information relevant to the input. For example, name or address format.	<pre>... <tr> <td oraLabel="address"></td> <td> <input type="text" oraField="address" /> <div class="oraTinyText" oraField="addressFormatHint"></div> </td> </tr> ...</pre>
oraZoneMap	This style is used applied when the UI Map is to be displayed as a zone on a portal.	<pre>... <body class="oraZoneMap"> ...</pre>

Using OJET

There are some UI maps delivered by the product that use UI widgets provided by Oracle JavaScript Extension Toolkit (OJET). Releases for OJET do not always align with releases of the framework. In addition, there are times when OJET adjusts APIs that the product uses. The framework will attempt to ensure that each release of the product has the latest and greatest version of the OJET libraries. Implementations are discouraged from attempting to use features in OJET that are not used by the product because the product is not necessarily testing those features and is not ensuring that upgrades to the APIs for those features are backward compatible.

Note that the product isolates the references to OJET into a UI map fragment that is included in the maps that use OJET widgets. This is so that changes to future versions of OJET are minimized to a single place. The map is called F1-OJETLIBS. If your implementation wants to use OJET, the recommendation is to use this UI map fragment.

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 this value is set to true it disables the generation of unique IDs, thus
replicating the old behavior. When this property is set to false or this property is missing
it enables the generation of unique IDs, thus enabling the list to be standards-compliant.
```

Process Flows

This section describes concepts and provides guidelines related to the configuration of various type of process flows.

Understanding Process Flows

A process flow is a user interface guiding a user through a series of actions in order to accomplish a specific task. The task can be as simple as the collection of information in order to update business data or involve more complex logic such as submitting and tracking batch processes, exchanging messages with an external system, etc.

This section describes topics related to designing and working with process flows.

A Process Flow Is Made Of Panels

Each process flow consist of a number of sequential "steps" needed to accomplish a certain task. Each step is represented by a stop on a progress bar and an associated panel.

Progress is linear in that each step may only lead to a single next step. A process flow always starts at a single initial step but allows for one or more final steps at which the user may choose to complete the process.

At any step the user may take the following actions:

- Enter data or take action as prompted by the step's panel.
- Continue to a next step, if any.
- Navigate back to any step they have previously visited.
- Finish the process flow if the current step is a final step.
- Save off their work on the process flow and either continue their work or navigate away. At a later time the user may resume their work on the process flow from where they have left off.
- Cancel the process flow.

A process flow type defines the entire metadata needed to control the behavior of process flows of a given type. This includes the sequence of steps, the panel and rules associated with each step and more.

Panel Presentation

The panel presented on each step of the process flow is rendered based on the panel type selected on the step configuration.

For a **Data Area** type of panel, the panel is rendered using UI Hints built into the schema of a specified data area.

For a **UI Map** type of panel, the panel is rendered using a specified UI map fragment.

For a **Panel Set** type of panel, the panel is rendered as a set of tabs, one for each panel that references the current panel as its parent panel. Clicking on each tab renders the UI based on the selected panel's type. The behavior of a panel set is

analogous to or can be thought of as a nested process flow within a single panel. A panel set may not include another panel set.

A Single UI Map

While the user enters data one panel at a time, data is collected and captured at the entire process flow level. All the elements edited and displayed on the various panels of the process flow must be included in the overall process schema and referenced by their appropriate XPath location in that schema. Each type of process flow defines its unique schema on a designated UI map.

The UI map's HTML should simply include the **Process Flow Controller (F1-ProcessController)** UI map fragment and nothing else. The latter is designed to render the user interface for a process flow based on the metadata defined on its process flow type.

Panel Scripts

A single **Pre-Processing** service script may be associated with a panel for the purpose of preparing the data before the panel is presented to the user. The system calls the script each time the user navigates to this panel on a non-finalized process flow.

A single **Post-Processing** service script may be associated with a panel for the purpose of validating the data entered by the user on that panel. The system calls the script each time the user exits the panel on a non-finalized process flow.

Process Flow Scripts

A single **Process Start** service script may be associated with a process flow for the purpose of preparing the data before a newly initiated process flow is presented to the user.

A single **Process End** service script is executed when the user clicks on the **Finish** or **Cancel Process** buttons to complete the process flow. The entire set of data collected by the process flow UI map schema is provided to this service script for final processing.

NOTE: The schema of these scripts should be the same as the **Process Flow UI map** schema. This would ensure that business rules have access to the entire data captured by the user.

Summary Panel

By default, when a user finishes a process flow, as well as when they view an already closed process, the process flow portal displays the last panel the user was working on. For more complex process flows, you may provide a panel that summarizes information from the entire process. When specified on the process flow type, the system displays that summary panel instead of the default view of a closed process. The user may toggle between the summary view and the detailed view at any time.

The summary panel may be implemented as a UI map fragment or a data area like other panels but may not be associated with any panel script.

Launching A New Process Flow

Each process flow must be initiated and executed by a designated BPA script that has the following simple steps:

- Set BPA area height to zero.
- Populate the temporary variable **\$processCode** with the code of the specific process flow type.
- Transfer control to the **Process Flow Navigation (F1-PROCEXEC)** common BPA script.

Launching a process flow of a specific type is assumed to be made from a designated menu option or within a specific context applicable to that type of work. As such, this item needs to be configured specifically for each type of process flow. The navigation option associated with that launching option should reference the BPA script associated with the corresponding type of process flow.

Saving Off Work

The user may save off their work on a process flow at any time. If enabled on the process flow type, the system would also automatically save the data when the user navigates to another panel or away from the process flow portal.

On either manual or automatic save operations, the **Process Manager** script defined on the process flow type is called to save off the entire process flow data on a designated record in the database.

The base product provides a generic **Process Flow** maintenance object that is designed to support the storage of any type of process flow as well a corresponding process manager script. Should process flows of a specific type be stored elsewhere, then a dedicated process manager script should be implemented and used instead.

Review Process Flow Records

A process flow record is created when a user saved off their work or the process flow type enables automatic saving. When the user finishes a process flow, depending on the process flow type configuration the record may be deleted or retained for audit purposes.

Reviewing process flow records of a specific type is assumed to be made from a designated menu option or within a specific context applicable to that type of work. As such this item needs to be configured specifically for each type of process flow. The navigation option associated with that item should reference a query portal designed to query and manage the specific type of process flow records.

In addition to standard query features, the query portal should allow the user to resume their work on a non-finalized process flow as well as review the data captured on a completed record. When implementing such a query portal, refer to the sample **Process Flow Query (F1PRSTRQ)** portal for key features. Your specific product may already include such query portals for process flow types it supports. Refer to your specific product documentation for additional information.

Designing Process Flows

A wide range of process flows may be designed to implement online tasks business users may need to perform. Refer to [Understanding Process Flows](#) for more information.

Below is a high-level summary of the steps required to design and configure various types of process flows.

Analyze The Task

The following provides a the high-level guidance in analyzing the necessary data and rules needed to support an implementation of a new type of process flow:

- Identify the data and business rules that are needed to complete the entire task.
 - When possible, organize data into logical groups of details defined as data areas.
 - Design logic for the final processing script that accepts the entire data entered along with the user's action to either complete or cancel the task.
 - If data needs to be initially prepared for a newly initiated process flow, design logic for a corresponding initial processing script.
 - Determine whether a summary panel may be useful for the user to review the outcome of a completed process.
- Design the sequence of panels the user should follow to complete the task. Identify the initial and final panels.
- Review each panel and determine the following:
 - Is it a single panel or a nested process flow (i.e. a panel set)?
 - What is the title of the panel and its label on the progress bar?
 - How should the layout be rendered? Use a data area that leverages UI Hints to describe the panel layout when possible. For more complex HTML requirements you may use a UI map fragment.

- Should details on this panel be defaulted with some initial values when the user enters this panel? If so, design logic for a corresponding pre-processing script.
- If data entered on this panel should be validated before the user exits the panel, design logic for a corresponding post-processing script.
- Design the summary panel if needed.
- Determine whether this type of process flow should be automatically saved by the system or solely rely on the user to control when data is saved.
- Determine whether a record of a completed process flow should be retained for audit purposes or deleted upon completion.
- Determine whether a user may delete in progress records of such process flow.
- Determine whether process flows of this type are saved and stored in a dedicated table in the database or are saved in the generic table provided by the base product. The assumption is that introducing new designated storage maintenance objects is not common. Refer to your edge product documentation for additional maintenance objects that may have been provided to support specific types of process flows shipped with the product.
- Determine the context in which a user can initiate, resume work on, and review process flows of that type.
 - The process flow may be launched from a menu or a favorite script etc.
 - Design the query portal a user would use to find and resume a saved off process flow of this type as well as review completed process flow records. Depending on the context of this task, the query may be associated with the same menu line that initiates the process flow.

Using UI Map Panels

Typically a map fragment is rendered as part of a complete UI map but when used as a process flow panel the system renders the panel assuming the fragment contains a complete map.

Therefore, such fragment should includes the following items:

- The standard error message section used for reporting errors.
- This line `<xml style="display:none"></xml>` at the end of the HTML content for proper binding of data to their HTML elements.

A UI map based panel allows for more control over some of the buttons available to the user. For example, the panel may prevent the user from proceeding to the next panel unless required information is entered. Refer to the explicit APIs listed in the process controller UI Map (F1-ProcessController) for more information.

Configure The Process Flow Type

The following describes the high-level steps needed to configure a new type of process flow:

- Create the UI map for the process flow. Its schema should contain all the details needed to complete the task. Its HTML should only include the process controller UI map fragment.
- Create a start processing service script if needed.
- Create a final processing service script such that it applies all the rules needed to complete or cancel the task.
- For each panel perform the following:
 - Create a UI map fragment for it unless it corresponds to a data area already included in the process flow schema. Make sure the XPath of each element references in the data area or UI map is the same XPath of that element in the process flow's UI map schema.
 - Create a pre-processing service script if needed. Not applicable to the summary panel.
 - Create a post-processing service script if needed. Not applicable to the summary panel.

- Set up a label and title MD fields if existing ones cannot be used.
- Create a BPA script for the process flow.
- Configure the [process flow type](#) record.
- Set up a navigation option and reference the BPA script on it. Associate the navigation option with a menu or provide other means for the user to initiate the process flow.
- Create a query portal for the specific type of process flow records. Add the new query portal to a menu or provide other means for the user to resume work on a process flow they have saved off or review completed records.

Control Visibility of Standard Buttons

You may leverage the following JavaScript functions to control the visibility of standard process flow buttons:

- Next: **enableNext()**, **disableNext()**, **hideNext()** , **showNext()**
- Finish: **enableFinish()**, **disableFinish()**, **hideFinish()** , **showFinish()**
- Save: **enableSave()**, **disableSave()**, **hideSave()** , **showSave()**.

Defining Process Flow Types

This portal is used to maintain process flow types.

Refer to [Understanding Process Flows](#) for more information.

You can access the portal from the **Admin Menu > System > Process Flow Type**.

The following zones may appear as part of the portal's **Main** tab page

- **Process Flow Type List.** This zone lists all process flow types. Broadcast a record to display the details of the selected process flow type.
- **Process Flow Type.** This zone provides information about the selected process flow type.

Action Method

Action method is an object that may be used to define configuration based on some criteria. For example, when initiating a Start Service scenario, the service request type may differ based on the CIS division and the customer class. When a user initiates a Start Service process, the system can prompt the user for information about the customer that can determine the CIS division and the customer class. The system can then call an algorithm passing this information to retrieve the appropriate service request type to use as defined in the action method. Throughout this section, the term 'criteria' is used to identify the information used to determine the desired configuration and the term 'details' or 'results' is used to identify the configuration information being determined.

If you were to design a use case to use this functionality consider the following:

- What information needs to be identified (the 'details'/'results')? The example above uses a configuration object as its example (service request type). However, the information you are determining could be anything that can differ by some combination of attributes. You could use this to determine which collection algorithm to execute based on criteria. Or the number of days to wait to continue with some process for some use case.
- When and where do I need to determine this information and what do I do with it once I receive it?
 - Is this needed as part of a user interaction?
 - Or is this needed for internal processing?

- What is the criteria needed to determine the desired information? Do I have this information at the time that I need it? Or do I need to prompt a user for the information?

The framework product supplies the generic objects needed to support this type of functionality. The individual edge products or implementations can build their use cases with the tools provided. The topics in this section further describes the provided features.

Action Method Role

The action method functionality is driven by the Action Method Role, which represents a given use case. Some examples of action method roles that may be relevant in an enterprise product are 'Start Service', 'Stop Service' and 'Transfer Service'. The valid action method roles are defined using an extendable lookup and referenced on the Action Method. One and only one action method may be defined for one action method role.

The action method role record may reference an action method role BPA script, if the use case for this Action Method Role involves user interaction. The responsibility of this script, if needed is to take the action method role value, determine the action method that references the role and identify the more specific Action Processing BPA script (defined on the business object) and transferring control to that script. This reference may be used for audit or information purposes, depending on how you have implemented the user experience. The action method role BPA is ideally configured on an appropriate menu entry or an action button on a specific user interface related to this use case.

If your business use case does not require user interaction, then an action method role BPA script is not necessary.

Action Method Business Object

The action method's business object is used to define the information that needs to be configured for both the criteria and the results. Continuing to use the above example, the business object for a Start Service use case defines a list that captures the CIS division and customer class along with the service request type to use. The information is configured in the XML data area for the action method and the schema design is dictated by the business requirements. For example, it may be that your use case dictates a default value for the results (in our case, the service request type) plus override values based on a division and customer class combination.

Get Action Method Details Plug-in

Besides defining the schema information for capturing the criteria and results, the business object defines the algorithm that should be invoked to retrieve the results based on the criteria.

The plug-in spot is **Get Action Method Details**. Its API is flexible in the information it receives and returns so that each action method use case can design the algorithm based on its business requirements.

The algorithm receives a list of 'Criteria Data' using a 'name' and up to 5 values (to handle multi-part primary keys). Typically only 'value 1' is needed for each entry. The algorithm type designed for a given action method business object can determine what information is receives. It could be that it expects the criteria elements defined on the action method. In our example, the algorithm type could expect to receive the CIS division and customer class. Alternatively, the algorithm type could be designed to receive a value that can be used to determine the criteria needed. For example, it could receive the account ID and use that information to determine the CIS division and customer class.

The algorithm returns one or more results as 'Details'. The information returned is based on the use case. In our example, the service request type is returned. Your business requirements may use the action method role to return multiple results, for example a service request type and a process flow type to invoke to capture all the information needed to start service.

The product provides a business service **F1-RetrieveActionMethodDetails** to call the algorithms for a given action method value.

Action Processing BPA

If your business requirement requires user interaction either to determine the criteria to pass to the Get Action Method Details algorithm or to use the resulting information to continue with an online action, an Action Processing BPA is needed. This BPA should be designed and linked to your action method business object as an option using the **Action Processing BPA Script** option type.

The design of this BPA depends on your business requirements. But at a high level, it may need to prompt the user for information needed to determine the criteria associated with the action method details, then call the **F1-RetrieveActionMethodDetails** business service to execute the algorithms for your action method. Once the results are returned, the script would proceed with the next step in the business use case.

NOTE: If any additional information needs to be gathered by the server before or after the call to the business service that executes the algorithms, then the recommendation is that the BPA should call a service script that does all the server level logic at one time.

Action Method Role vs. Action Method Business Object

The action method role defines the use case. The business object is used to design the schema for the criteria and the results and identifies the action processing BPA and the Get Action Method Details plug-in. It may warrant a unique business object for each action method role. However, there may be cases where more than one action method role and its action method can reuse the same business object. For example, imagine there are different service request types for Start Service, Stop Service and Transfer Service, however in all cases the valid value is driven by the CIS division and customer class. These three separate action method roles and action methods can reuse a common business object defining the criteria of CIS division and customer class determining the appropriate service request type.

Defining Action Methods

Refer to [Action Method](#) for an overview of action method functionality.

To maintain the action methods applicable to your implementation, open **Admin > System > Action Method**.

This is a standard [All-in-One portal](#) and includes the standard List and display zones for an action method.

The information captured on the action method depends on the business objects supported by your product or implementation. Refer to the embedded help text for more information.

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 an algorithm. Algorithms are called at strategic points (often referred to as a "plug-in spot") within system processing to perform specific code. The plug-in spots are preconfigured with the product and vary from object to object. Algorithms are used by the base product to provide out of the box functionality. The algorithm framework provides a means for implementations to insert their own custom code, which can be instead of the base delivered code or in addition to it.

NOTE: Algorithm = Plug-in. We use the terms plug-in and algorithm interchangeably throughout this documentation.

Algorithm Type Versus Algorithm

Often the system will just talk about algorithms but in actuality there are two important objects that provide this plug-in functionality.

- An **Algorithm Type** defines the plug-in spot. In addition, it defines program to execute. The program may be defined using Java or using a [plug-in script](#). The algorithm type also defines any parameters that must be supplied to algorithms of this type.
- An **Algorithm** is an instance of an **Algorithm Type**. If the algorithm type does not define any parameters, then there will be only one algorithm for the algorithm type. If the algorithm defines parameters, then there could be several algorithms for a given algorithm type, one for each unique set of parameter values for a given use case. The algorithm is the object that gets referenced on a given control table when configuring your system.

Algorithm Plug-in Spot

The plug-in spot, also called Algorithm Entity and System Event, defines several aspects of the functionality for algorithms that are executed.

- When in system processing are algorithms of this plug-in spot executed? For example, algorithms for the Installation Option - To Do Pre-creation are called when a To Do is getting added, before any insert to the database is performed.
- The API for the plug-in spot. This is often referred to as the "hard parameters" for the plug-in spot. What information is every algorithm provided when called? Additionally, if the algorithm is expected to return information to the caller, this is also part of the hard parameters. For example, the Installation Option - To Do Pre-creation plug-in spot receives all the information about the To Do that is about to be added. It may add or update values. It may also return an indication to not create the To Do, if desired.

NOTE: The hard parameters are different from the parameters defined on the algorithm type. Those are referred to as the "soft parameters". Those parameters are used to configure settings that the algorithm type code uses per its specific functionality.

- What is the responsibility of algorithms for the plug-in spot? When defining a plug-in spot, the product defines the responsibility of the code for the algorithm types created for that spot. For example, is the algorithm determining information and returning it? Or is it supposed to perform an update? Algorithms for the Installation Option - To Do Pre-creation plug-in spot should not do any updates. Rather they should adjust the To Do information (if desired) in the hard parameters and return them. The calling logic will use that adjusted information when inserting the To Do record.
- Where do you configure the algorithm? We have been using the Installation Option - To Do Pre-creation plug-in spot as an example. As the description suggests, these algorithms are configured on the Installation Option - Algorithm collection. Each plug-in spot defines where (on which control table) the algorithms are plugged in. The system standard for the description of the Algorithm Entity flag value that defines the available plug-in spots is "Control Table Name - Event". For example the To Do Type - Calculate Priority algorithm plug-in spot means that algorithms are configured on the To Do Type. This information also indicates the granularity of the algorithm. Installation Option algorithms are system wide. But an algorithm plugged in on To Do Type, for example, indicates that only To Do Entries for that To Do Type will have that algorithm executed.
- Are multiple algorithms allowed? There are some plug-in spots where it doesn't make sense to support multiple algorithms. An example of this is the Batch Control - Select Records plug-in. This plug-in is used by the plug-in driven batch process and is responsible for selecting the records to process. Only one algorithm of this type is allowed on a given Batch Control. An example of a plug-in spot that would support multiple algorithms is a validation plug-in spot. Note that when a plug-in spot supports multiple algorithms, there can be yet additional aspects of the plug-in spot design.
- In some cases, this type of plug-in spot simply executes each algorithm. A validation plug-in spot is an example of this.

- In some cases, the algorithm could return an indication to the plug-in spot 'driver' to not execute any subsequent algorithms. Business Object Status - Enter is an example of a plug-in spot that supports this capability.
- In some cases, the plug-in spot may support sharing information from one algorithm to the next. Migration Plan - Pre-Compare is an example of a plug-in spot that supports this capability.

Each plug-in spot is defined in the Algorithm Entity lookup (**ALG_ENTITY_FLG**). When defining an algorithm type or defining a plug-in script, once you select the algorithm entity value, a help icon should appear. Clicking this icon will show you the description of the plug-in spot. It may be that online help provides more information about a given plug-in spot as well. Note that to see the hard parameters defined for a given plug-in spot, go to the [Script](#) page in add mode and select the script type of plug-in and the desired algorithm entity lookup value. Then navigate to the Data Area tab. The **View Plug-In Script Data Area** link will display the hard parameters.

Base Product Algorithms

The base product provides many, many algorithm types and algorithms out of the box. Sometimes algorithms are delivered to provide support for common use cases that we know our clients will need. Many algorithms, however, are provided because we are implementing system functionality using the same infrastructure and capabilities that our implementation use for their own custom code.

When an algorithm type is delivered, the decision as to whether an algorithm is also provided depends on some factors related to whether soft parameters are defined for the algorithm type.

- If no parameters are defined for the algorithm type, a corresponding algorithm is typically also provided.
- If parameters are defined and the values are on configuration data that is normally owned by the implementation, a base algorithm will probably not be defined.
- If parameters are defined and the values are not dependent on configuration data, it's possible that one or more algorithms are provided.

To review the algorithm types that are delivered with the product, go to the Algorithm Type query. The query provides several search options, including searching by the algorithm entity lookup value.

How To Add A New Algorithm

Before you can add a new algorithm, you must determine if you can use one of the [algorithm types](#) supplied with the system by using the algorithm type query.

If you can use one of the algorithm types, simply add the algorithm and then reference it on the respective control table. Refer to [Defining 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 embedded 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.

Defining 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**.

Use the **Algorithm Type Query** portal to search for an existing algorithm type. Once an algorithm type is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

Defining Algorithms

If you need to introduce a new algorithm, open **Admin > System > Algorithm**.

Use the **Algorithm Query** portal to search for an existing algorithm. Once an algorithm is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

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.

Advanced Algorithm Tips

Initiating Actions that Cannot be Rolled Back

If you have an algorithm that creates a sync request or performs some other logic that could cause a message to be sent and cannot be rolled back, you should be sure to cater for warnings in the design.

- The algorithm should be performed after all validations that could accumulate warnings.
- The algorithm should include a step to call the Business Service **F1-CheckWarning** and see if any warnings have been accumulated. If so, don't perform the action. It means that the user will get a warning and if they click "OK", all the processing will happen again, but this time without accumulating any warnings. Refer to the plug-in script **F1-GnChgDCap** for an example.

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.

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 [Edit Data Syntax](#) for more information on edit data commands and examples of the use of dynamic data areas.

NOTE: Some server based scripts may not use data areas as means of defining or exchanging data, depending on script type and the chosen scripting technique. Refer to [The Big Picture Of Server Based Scripts](#) for an overview of server scripts and their applicable scripting options.

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 for user-invocable BPA scripts. 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. For service scripts, the application service is required.

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

We recommend considering the approaches outlined below when you construct scripts.

While designing your scripts, determine the most maintainable way to set them up. Rather than creating complex, monolithic scripts, we recommend dividing scripts into smaller sections. For example

- 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 using 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 sub script via a **Transfer control** step.

For debugging purposes, 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. There are two techniques you can use to assist in debugging these step types.

- You can instruct the system to display text in the script area.
- 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. This may be used for integrations from an external system or for use cases within the application where a link has been configured to launch the application in a new browser tab, allowing the original page to remain open. Refer to [Creating Application Links in External Applications](#) for details of what is supported.

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.

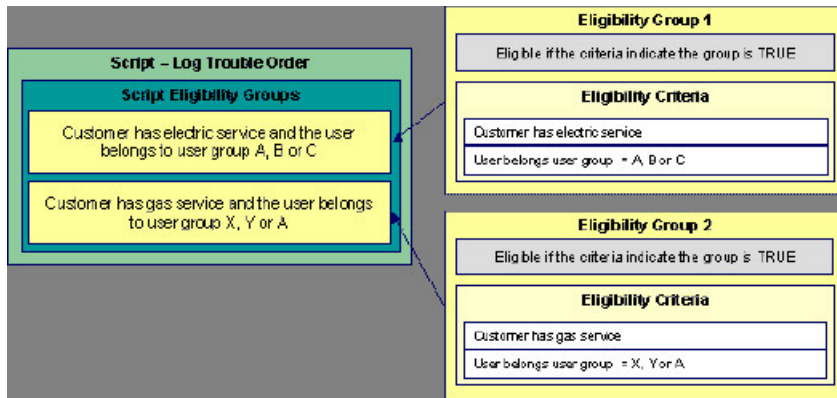
Criteria Groups versus Eligibility Criteria

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	Eligible	Check next group
2	Customer has gas service and the user belongs to user group X, Y or A	Eligible	Ineligible

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	Check next condition	Group is false	Group is false
20	User belongs to user group A, B or C	Group is true	Group is false	Group is false
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	Check next condition	Group is false	Group is false
20	User belongs to user group X, Y or A	Group is true	Group is false	Group is false

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	Eligible	Ineligible

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	Check next condition	Group is false	Group is false
30	Person characteristic: Date of Birth	Algorithm: True if senior	Group is true	Group is false	Group is false

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).

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). Note that this time span algorithm can be used to compare days, weeks and months as well.

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).

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	Eligible	Ineligible

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	Check next condition	Group is false	Group is false
20	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 algorithm type **SECF-USRNGRP** for an example of this type of logic.
- **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 three types of server-based scripts, **Plug-In** scripts, **Service** scripts and **Groovy Library** scripts.

- Plug-in scripts allow an implementation to develop routines that are executed from the system's various plug-in spots. 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. Service scripts are typically written using xpath scripting.
- Groovy Library scripts allow an implementation to develop groups of common routines written in the Groovy language which may be invoked from **Groovy Member** step types.

The topics in this section describe background topics relevant to server-based scripts.

Additional Coding Options For Server Scripts

Server based scripts often perform complex functions best supported by coding in languages with more comprehensive command sets than the base script steps. The system supports two common third-party languages for this purpose.

XML Path Language (XPath) is a language for querying and evaluating elements or nodes in an XML document. XPath commands and expressions can be used directly within [edit data](#) step types. The script engine version is used to define the applicable XPath version.

Groovy is an object-oriented, dynamic language for the Java platform. The framework supports the use of Groovy within server-based scripts to provide restricted and controlled access to Java-like code, particularly for cloud based implementations. The following topic provides more information on how to incorporate Groovy code into scripts.

Using Groovy Within Scripts

Groovy code can be incorporated in scripts using the step type **Groovy Members**. For each script with Groovy code, there will be a single Groovy class created by concatenating all **Groovy Members** steps.

For security, the product and third party Java classes available for scripting in Groovy are restricted. The allowable base classes may be viewed via the Groovy Javadocs feature in the Application Viewer and also via the 'View Groovy Javadocs' link in the context sensitive Script Tips zone. The list of allowable third party classes can be viewed via the 'View third party Groovy allowlist' link in the Script Tips zone.

NOTE: This system supports the use of Groovy for back end processing purposes. It is not intended for user interfaces. Groovy is also not applicable to BPA scripts.

Groovy Script Engine

For scripts invoked from within Java and entirely written in Groovy should use a script engine version of **Groovy**. The system processes this type of script in a similar way to code written in Java. This avoids the need to convert the script to and from XML format and allows the use of code that acts directly on the system objects with consequent performance benefits.

The following script types support the Groovy engine version:

Plug In Scripts

Plug-in Scripts can be configured to use the Groovy engine if they contain only Groovy oriented step types. The system provides an automatically generated superclass that defines the plug-in spot API. Internally, the Groovy code must conform to the system conventions for Java based algorithm types, including the inclusion of an 'invoke' method that is the plug-in entry point, and the definition of 'soft' parameters using annotations.

Groovy Library Scripts

Groovy Library Scripts provide the ability to create groups of common routines written in Groovy that can be called from within other scripts. Scripts of this type must include a single step type of **Groovy Library Interface** in which the publicly available methods in the library are listed. The supporting code for those methods is defined in one or more **Groovy Members** step types within the library script. The methods defined in the library can accept arguments and return values of any type. Scripts of this type use the Groovy engine by default and cannot include scripting step types.

Scripts that need to invoke methods from a Groovy library can use the `createLibraryScript` method provided by the system to instantiate the library interface. Ensure that the class being instantiated includes the `.class` filename extension.

Invoking Groovy within the Scripting Engine

If a plug-in script or a service script is configured to use a scripting engine version, it can include a mixture of regular and **Groovy Members** step types. The script step types will define the process to be executed. The **Groovy Members** steps contain code that can be called from **Edit Data** step types within the script using the `invokeGroovy` command. Only Groovy methods that receive no arguments and return void are supported using this command. Refer to the section on [edit data](#) steps for more details.

For scripts using this option, the framework provides a superclass containing methods that support common scripting actions such as move commands, string evaluation, and methods to invoke business objects, business services and service scripts. Refer to the Groovy specific Javadocs for details of the supported methods.

Importing Groovy Classes

All the Groovy classes used by your script should be explicitly declared in a **Groovy Imports** step type. When saving a script, if the system finds classes referenced in the script that are not explicitly declared, the system issues a warning and automatically adds import statements for the classes. Script writers should review the imports to confirm that they are the correct classes (in case there is a class with the same name in two different packages).

Backward Compatibility

In previous releases, the system did not require explicit imports and had logic to automatically import all callable classes available to Groovy at run-time. For any existing script that does not explicitly declare its imports, the system continues to fall back on importing all callable classes at runtime. However, the recommendation is to identify and update any such script as this is not the recommended practice.

The system provides a batch program **F1-CAGVY** that reports all scripts that required the automatic import step to successfully compile. In addition, this batch program is useful when the version of Groovy is upgraded to proactively detect possible issues in compilation of existing scripts that may occur due to changes in Groovy. Implementations are encouraged to run this batch program in the short term to identify and fix any scripts missing explicit imports and also for any Groovy version upgrade to review any implementation owned scripts that may be reported.

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.

As an alternative to writing a java program for a plug-in spot, the framework supports creating plug-ins using the defined script steps, Xpath commands, Groovy code or a combination of these three options.

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"

For plug-ins configured to use a script engine version, the best way to understand the 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.
- 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.

Plug-ins configured to use the **Groovy** engine version do not use an XML interface for the API and instead are processed in the same way as Java algorithms. The framework supplies a dynamically generated superclass that implements the plug-in spot for Groovy objects. Use the **View Script Superclass** hyperlink to view this superclass and the methods to set and get the hard parameters.

NOTE: For plug-in scripts using the **Groovy** engine version, soft parameters do not appear in the plug-in spot API as defined by the superclass. Plug-ins using only Groovy code define their soft parameters using annotations, in a similar way to Java algorithms, and fetch those values using methods defined in the algorithm code.

Setting Up Plug-In Scripts

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.

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 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: Refer to [Schema Nodes and Attributes](#) for a complete list of the XML nodes and attributes available to you when you construct a schema.

Invoking Service Scripts

Any type of script configured to use a scripting engine version 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.

Groovy Library Scripts

Just as service scripts can define common routines written in scripting language, Groovy library scripts are used to define groups of common components or methods written in Groovy. Groovy library code runs on the application server and performs server-based processing for scripts that utilize Groovy code.

The following topics describe basic concepts related to Groovy library scripts.

A Groovy Library Script's API

A Groovy library script's API is composed of one or more public methods whose code is defined in the script's steps. Those methods are defined in a step type of **Groovy Library Interface**. A Groovy library script must have one (and only one) step of this type.

Invoking Groovy Library Methods

Any type of script that supports the **Groovy Members** step type may invoke common methods defined in Groovy library scripts.

Code within a **Groovy Members** step must create an instance of the Groovy library interface definition to enable the interface methods to be invoked. Refer to the topic [Using Groovy Within Scripts](#) for more information.

Debugging Server-Based Scripts

The server can create log entries to help you debug your server scripts.

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.

Refer to the [Debug Mode](#) topic in the Configuration Tools chapter for details of how to execute the application in debug mode.

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**.

Use the **Script Query** portal to search for an existing script. Once a script is selected, you are brought to the maintenance portal to view and maintain the selected record.

NOTE: Script Tips. A context sensitive "Script Tips" zone is associated with this page. The zone provides links to [Edit Data Syntax](#) and [Advanced Schema Topics](#) so that users can quickly access those online help topics to aid in constructing scripts. In addition, the zone provides links to view the [Groovy JavaDocs Viewer](#) and the allowlist of third party Groovy classes so that users can verify the restricted list of classes available for Groovy coding in the script.

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**, **Groovy Library 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. This field does not appear if the script type is **Groovy Library Script**.

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 key information affecting the context and execution of the script.

- Script engine version values of 1, 2 and 3 define the version of the XML Path Language (XPath) to be used for the script. 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.

The default script engine version is 3.0 for plug-in and service scripts. The default version for BPA scripts is 1.0 as higher level versions are not applicable.

There are some additional details to note about script engine version 1.0:

- The XPath library used is Jaxen
- For BPA scripts, it uses the browser's xpath and XML support except for Internet Explorer where the XSWML parser is used.
- 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
- A Script Engine Version value of **Groovy** is only applicable to plug-in scripts. It is used to indicate that only Groovy related step types are used in the script and signals to the system that there is no need to convert the data to and from an XML interface. This allows for greater efficiency in script execution.
- The value **Framework Version 2.1 Compatibility Mode** remains for upgrade purposes. This value should only be applicable to early versions of BPA scripts using syntax that is incompatible with XPath syntax.

NOTE: The **Script Engine Version** field does not appear for **Groovy Library** scripts. The script engine version for these scripts is set to **Groovy** by default and cannot be changed.

Click the **View Script Schema** to view the [script's data areas](#) on the [schema viewer](#) window. This link does not appear if the script engine version is **Groovy**.

Click the **View XSD** hyperlink to view a script's schema definition in XSD format. This link only appears if the script type is **BPA Script** or **Service Script**.

The **View Script Superclass** hyperlink appears only for plug-in scripts using an engine version of **Groovy**. Click this link to view the code of the runtime generated superclass for the related plug-in spot's implementation.

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 [tree](#) summarizes the script's steps. You can use the hyperlink to transfer 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.

Step Sequence 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. Note that some step types are only applicable for BPA scripts and some step types are only supported for server based scripts. Each section indicates whether there are any script type limitations.

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. Only the field types **Temporary Storage** and **Predefined Value** are applicable when the Condition is **In** or **Between**.

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.

Display Text Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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.

Edit Data Steps

Edit data steps provide a free format region where you can specify commands to control your script processing.

Enter your scripting commands in the **Edit Data Text** field. Click the adjacent icon to open a window providing more space for defining the edit data step.

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**.

NOTE: Not all BPA step types are supported using the edit data syntax. Refer to the details below for more information on edit data commands and examples.

The topics in this section provide detail of the syntax supported in the edit data step type.

Contents

- [Comments](#)
- [Temporary Variables](#)
- [Context Variables](#)
- [Move Statement](#)
- [Go To Statement](#)
- [Conditional Branch Statement](#)
- [If Statement](#)
- [For Statement](#)
- [List Processing](#)
- [Functions for Processing a List](#)
- [Declare and Invoke Schema Based Objects](#)
- [System and Global Variables](#)
- [Perform Script and Transfer](#)
- [Navigate Statement](#)
- [Invoke Map Statement](#)
- [Declare BO with BO Group](#)
- [Generate Edit Map Statements](#)
- [Terminate Statement](#)
- [Invoking Groovy Code](#)
- [Debugging a BPA Script](#)
- [Useful Tips and Tricks](#)

Comments

You can place comments into your script using the double slash notation `//` in the first two characters of the edit data step.

Example:

```
//  
// quit with error  
//  
if ("not(customer/securityEnabled)")  
terminate with error (8000, 1001 %1="customer/id" %2='not allowed');  
end-if;
```

Temporary Variables

Temporary variables can be declared within all types of scripts. They should be referenced by a leading single dollar sign (`$`). However, temporary variables behave differently in the various script types:

- In BPA Scripts temporary variables remain persistent from one BPA script to another (refer to the [Perform Script and Transfer Control statements](#)), which means that you can use temporary variables to communicate between BPA scripts.
- Temporary variables cannot be passed from a BPA script to a service or plug-in script. Only data area elements can be passed between these types of scripts.
- Within service and plug-in scripts, temporary variables remain persistent only for the life of the particular script that declared the variable. This means that temporary variables cannot be passed between plug-in scripts and service scripts, only [global variables](#), [context variables](#) and data area elements can be passed between these types of scripts.

Declaring / Initializing / Defaulting Temporary Variables

When using a temporary variable, it should be declared or initialized with an appropriate value before using it. A typical method for declaring a variable is to simply move data to it in a move statement, for example.

```
move "0" to $index;
```

FASTPATH: Refer to [Move to a Temporary Variable](#) for more information on implicit declaration of a temporary variable within a move statement.

For BPA scripts, as mentioned above, temporary variables may be passed from one BPA script to another. As such, it is common to reference a temporary variable in a BPA that should have been initialized by a previous BPA. However, if there is any reason that a temporary variable did not get initialized by a previous BPA, a reference to it will cause an error. It is good practice, therefore, to use the **default** statement that will initialize temporary variables that are not already created / initialized.

- The following statement will initialize the temporary variable \$InputAction – but only if the temporary variable has not yet been initialized:

```
default $InputAction;
```

- The following statement will set the value of the temporary variable \$InputAction to 'read' – but only if the variable has not yet been initialized:

```
default $InputAction as 'read';
```

NOTE: Scripts should take care not to define variables using a reserved keyword. The following table lists the reserved keywords.

Keyword

add

as

asError

bpa

branch

data

declareBO

declareBS

declareDA

declareMap

declareSS

default

delete

edit

element

else

end-edit

end-for

end-if

error

escape

evaluate

fastAdd

fastUpdate

for

goto

Keyword

if

in

invokeBO

invokeBS

invokeMap

invokeSS

label

map

move

navigate

navigateAndReloadDashboard

null

page

performScript

popup

read

readWithoutVersion

replace

suppress

target

terminate

to

transferControl

update

using

warn

with

Context Variables

Context variables are only available within service scripts. The context variable will be available for the duration of the service script and all that it invokes. Therefore, you can use a context variable within a service script to communicate information to a lower level service script or schema. They should be referenced by leading double dollar signs ('\$\$').

NOTE: Because context variables are available for lower level scripts, they may sometimes be referred to as global variables or global context variables. But they should not be confused with [global variables](#).

Declaring / Initializing / Defaulting Context Variables

When using a context variable, it should be declared or initialized with an appropriate value before using it. A typical method for declaring a variable is to simply move data to it in a move statement, for example.

```
move 'context variable' to $$contextVariable;
```

FASTPATH: Refer to [Move using a Context Variable](#) for more information.

Move Statement

The **move** statement copies a source value to a target. The following table highlights various options supported in the move statement.

Statement	Example Description	Example Syntax
Move to Element move "xpath" to "xpath";	Move statement with simple XPath reference.	<pre>move "acct/totalBalance" to "parm/formattedValue";</pre>
NOTE: An XPath expression is surrounded by double quotes.	Move statement with XPath concatenate function.	<pre>move "concat(person/ firstName, ', ', person/lastName)" to "parm/fullName";</pre>
	Move statement with XPath substring-before function.	<pre>move "substring-before(parm/ fullName, ', ')" to "person/firstName";</pre>
	Move statement with XPath substring-after function.	<pre>move "substring-after(parm/ fullName, ', ')" to "person/lastName";</pre>
	Move statement with XPath substring function.	<pre>move "substring(parm/date,1,4)" to "parm/year";</pre>
Move to Element move 'literal' to "xpath";	Move statement using a literal string.	<pre>move 'okay for mailing' to "account/ preferences[type="mail"]/text";</pre>
Move to Element move 'Boolean' to "xpath";	Move statement with Boolean as literal string.	<pre>if ("account/balance > 0") move 'true' to "account/hasDebitBalance"; end-if;</pre>
	Moving an expression, which results in a Boolean. Note that the filter in the example below is located on a group node.	<pre><schema> <account> <hasDebitBalance type="boolean"/ > <balance/> </account> </schema></pre> <pre>move "boolean(account[balance>0])" to "account/hasDebitBalance";</pre>
Move to Group move "xpath" to "xpath";	<p>Move a set of elements from one group with another.</p> <p>The system matches the initial level of element names and the group / list names from the source schema to the target schema. For lists and groups, the default behavior is to move all elements within the source group / list to the target group / list (even if they are not defined in the target group / list).</p> <p>The business service F1-MoveByName may be used for more granular control of the move</p>	<pre>move "account/ custInfo" to "person";</pre> <p>This statement is equivalent to the following statement:</p> <pre>move "account/custInfo/ *" to "person/*";</pre>

Statement	Example Description	Example Syntax
	(without having to define each individual move statement).	
Move using a Temporary Variable	When moving to a temporary local variable, it is not surrounded by double quotes. move "xpath" to \$variable;	<pre>move "count(Person/names/ personName) + count(Person/ids/personId)" to \$PersonChildCount;</pre>
	When moving from a temporary variable, the variable is surrounded by double quotes. move "\$variable" to "xpath";	<pre>move "\$AccountBalance" to "parm/formattedValue";</pre>
Move using a Context Variable move "xpath" to \$\$variable; move \$\$variable to "xpath";	Context variables, source or target, are referenced without any double quotes.	<pre>move 'context value' to \$\$contextVariable; // // here, we move from a context variable. move \$\$contextVariable to "MarketMessage/sender";</pre>
Move using a Dynamic Location move "xpath" 'literal' to evaluate("xpath" \$variable); move evaluate("xpath" \$variable) to "xpath" \$variable;	The evaluate statement allows your move source or target location to be dynamically derived from a variable or schema element location.	<pre>move 'literal' to evaluate("schemaLocation"); // move "schemaLocation" to evaluate(\$Variable); move evaluate("schemaLocation") to \$Variable; // move evaluate(\$Variable) to "schemaLocation";</pre>
Move escape move escape("xpath" \$variable) to "xpath" \$variable;	Move escape is only available for service scripts and plug-in scripts. The escape statement scans your source text value for HTML content and escapes it, i.e. replaces any HTML-like characters with special characters that are escaped from HTML rendering. By doing so the text would be displayed as plain text when displayed as part of an HTML element. NOTE: You should only use this function if the text is to be displayed as part of an HTML element and is suspected to contain HTML-like characters or even malicious HTML that should not be rendered as HTML. If incorrectly displayed using a non HTML element, the special escape characters, if any would be visible as part of your text. Refer to the UI Map Attributes and Functions for more information on how to define an element to display HTML content.	<pre>move escape("schemaLocation") to \$Variable; // move escape(\$Variable) to "schemaLocation";</pre>
Move null move null to "xpath";	You can remove information from the XML instance document through the special syntax of move 'null'. Note that you can specify either a node name in the XPath expression or a group name. If you specify a group then the group and all child elements will be eliminated from processing.	Remove a node and all of its child nodes: <pre>if ("boolean(customer/ securityEnabled)") goto updateInfo; else move null to "customer"; end-if;</pre>

Statement	Example Description	Example Syntax
		Remove all child nodes of a group node with the suffix <code>/*</code> . <pre>if ("boolean(customer/ securityEnabled)") move null to "customer/*"; end-if;</pre>

Go To Statement

The edit data step supports functionality analogous to the [Go To](#) step type. The syntax is **goto label**; where the label represents another location within the edit data text field (identified by this label) or represents another step in the script.

The following is an example of going to another location in the same step identified by the label **addSpouse**.

```
if ( "string(parm/spouse/name) != $BLANK" )
  goto addSpouse;
end-if;
addSpouse: invokeBO 'Person' using "parm/spouse" for add;
```

The following is an example of going to a different step within the same script. The step sequence is the reference used as the label.

```
if ( "string(parm/spouse/name) != $BLANK" )
  goto 110;
end-if;
.
.
.
110: invokeBO 'Person' using "parm/spouse" for add;
```

Conditional Branch Statement

The edit data step supports functionality analogous to the [Conditional Branch](#) step type. The syntax is **branch (“xpath”) goto label else label**; where:

- The XPath condition in the **branch** statement must evaluate to a Boolean value of True or False.
- The targets for the **goto** and **else** statements are labels that represent another location within the edit data text field (identified by this label) or represent another step in the script.

The following example uses labels for **addSpouse** and **addAccount**

```
branch ( "string(parm/spouse/name) != $BLANK" ) goto addSpouse else addAccount;
```

If Statement

The **if** statement is similar to the conditional branch statement. Either can be used to structure the logic of your script. This statement may optionally include an **else** statement but it should always end with an **end-if** statement.

NOTE: This is an example of a statement that is not represented as a separate step type. It is only available within the edit data text.

The syntax is **if (“xpath”) else end-if**; The XPath condition must evaluate to a Boolean value of True or False. The following are some examples.

Example where the XPath contains a simple logical condition.

```
if ( "string(parm/spouse/name) != $BLANK" )
  //
  // Create spouse since spouse name present
  goto addSpouse;
else
  //
  // Create account without spouse
```

```
goto addAccount;
end-if;
```

Example where the XPath contains a complex condition.

```
if ("string(parm/spouse/name) != $BLANK and string(parm/hasSpouse) = true or boolean(parm/requireSpouse)")
//
// Create spouse since spouse name present
goto addSpouse;
end-if;
```

Example of a stacked set of statements used to evaluate multiple possible values of a field.

```
if ("parm/rowCount = 0")
//
// no rows found
goto quit;
end-if;
if ("parm/rowCount = 1")
//
// one row found
goto process;
end-if;
if ("parm/rowCount > 1")
//
// more than one row found
goto quit;
end-if;
quit: terminate;
```

The following XPath shows Boolean based on the existence of the node. In this example, if the node exists in the XML instance document being processed, the statement will evaluate to True. If no element is found, the statement evaluates to false.

NOTE: When treating XPath nodes as Boolean variables be aware that an empty node evaluates to True. Only a missing node return False.

```
if ("boolean(parm/spouse/name)")
goto addSpouse;
else
//
// Create account without spouse
goto addAccount;
end-if;

if ("not(parm/spouse/name)")
//
// Create account without spouse
goto addAccount;
else
goto addSpouse;
end-if;
```

For Statement

The **for** statement creates a list of nodes or values depending on your XPath expression. If you specify a list node then every child node of the list, along with its contents, will be available within the loop. If you specify a child node directly, then a list of values only will be available within the loop.

NOTE: For more information on creating new entries in a list, please refer to the [creating a new list instance](#) example.

NOTE: This is an example of a statement that is not represented as a separate step type. It is only available within the edit data text.

The syntax is **for (\$variable in "xpathList") end-for;**. The XPath condition must evaluate to a Boolean value of True or False.

The following examples are based on this sample schema:

```
<schema>
  <SAList type="list">
    <id/>
    <balance/>
  </SAList>
  <SAContributor type="list">
    <id/>
  </SAContributor>
</schema>
```

Example that specifies the list node in the XPath expression where all child nodes are available for processing.

```
move "0" to $AccountBalance;
move "0" to $index;
for ($SAList in "SAList")
  move "$SAList/balance + $AccountBalance" to $AccountBalance;
  //
  // keep track of each SA contributing to the balance in the SA Contributor list
  move "1 + $index" to $index;
  move "$SAList/id" to "SAContributor[$index]/id";
end-for;
```

Example that specifies a child node within the list node in the XPath expression. Only values of that node are available for processing.

```
move "0" to $AccountBalance;
for ($SABalance in "SAList/balance")
  move "$SABalance + $AccountBalance" to $AccountBalance;
end-for;
```

Example that shows that a filter can be used to limit the rows selected by the **for** loop.

```
move "0" to $AccountDebitBalance;
for ($SAList in "SAList[Balance>0]")
  move "$SAList/balance + $AccountDebitBalance" to $AccountDebitBalance;
end-for;
```

Example that shows the use of a filter when specifying child nodes.

```
move "0" to $AccountCreditBalance;
for ($SABalance in "SAList[Balance<0]/balance")
  move "$SABalance + $AccountCreditBalance" to $AccountCreditBalance;
end-for;
```

List Processing

This section provides details about processing lists. The examples in this section reference the following schema:

```
<schema>
  <parm type="group">
    <name/>
  </parm>
  <Person type="group">
    <names type="list">
      <type/>
      <name/>
    </names>
  </Person>
</schema>
```

Referencing a List Element. You can move a value to a particular list instance by referencing an identifying node in the list within a filter. The syntax is **move "xpath" to "xpathList[filter]/element"**; Example:

```
move "parm/name" to "Person/names[type='main']/name";
```

Creating a New List Instance. A special notation can be used within a move target statement to indicate a new list instance should be created. The "+" indicates to the script processor that a new instance of a list should be initiated for the target element. The syntax is **move "xpath" to "+xpathList"**; Example:

```
move "parm/name" to "Person/+names/name";
```

Deleting a List Instance. An XML list entry can be deleted from the database by moving an action attribute of 'delete' to the element name. To cause a database delete of a list entry requires an attribute of action="delete" in the target node and a subsequent update BO interaction. The syntax is **move 'delete' to "xpathList@action"**; Example:

```
if ("parm/action = 'd'")
  move "0" to $index;
  for ($CCList in "CCList")
    move "1 + $index" to $index;
    if ("CCList/id = parm/id")
      move 'delete' to "CCList[$index]@action";
      goto update;
    end-if;
  end-for;
end-if;
```

The following shows the resulting XML.

```
<root>
  <CCList>
    <id>9876538976</id>
    <balance>309.98</balance>
  </CCList>
  <CCList action="delete">
    <id>4321125899</id>
    <balance>87.45</balance>
  </CCList>
</root>
```

NOTE: Deleting a list instance through use of the action attribute is risky if iterative BO interactions are required. The XML document that contains the list instance to be deleted will not be altered after a successful BO interaction, which means the document will still contain the list instance even though it no longer exists. To solve this problem, it is essential to re-read the BO after any BO update where the action attribute of 'delete' has been used.

NOTE: An alternative to the delete attribute described here, is to use the BO action of [replace](#). Manipulating a list to use the replace action avoids the problem described above concerning stale information in request documents post BO update.

Functions for Processing a List

XPath provides several functions that are useful to process elements of a list including **count**, **sum** and **last**.

The following examples are based on this sample XML document:

```
<xml>
  <ft>
    <type>bill</type>
    <date>20100101</date>
    <amt>30.30</amt>
    <cat>tax</cat>
  </ft>
  <ft>
    <type>adj</type>
    <date>20100301</date>
    <amt>20.20</amt>
    <cat>int</cat>
  </ft>
  <ft>
    <type>bill</type>
    <date>20100201</date>
    <amt>10.10</amt>
    <cat>tax</cat>
```



```
</ft>
</xml>
```

The following is an example of a sum. The syntax is **move "sum(xpathList/element)" to \$variable**; The example sums the total balance.

```
move "sum(ft/amt)" to $TotalBalance;
```

The following is an example of a sum using a filter to get a subtotal. The example sums the balance of the entries that have the 'tax' category.

```
move "sum(ft[cat='tax']/amt)" to $TaxBalance;
```

The following is an example of a count. The syntax is **move "count(xpathList)" to \$variable**; The example finds the count of the number of FT entries in the list.

```
move "count(ft)" to $TranCount;
```

The following is an example of 'last', which is used to locate the last entry. The syntax is **move "last(xpathList)" to \$variable**; The example finds the last amount in the FT list.

```
move "ft[last()]/amt" to $LastAmount;
```

Declare and Invoke Schema Based Objects

You can invoke a business object, business service or service script within the edit data step. To support the dynamic invoke, a dynamic data area name can be declared.

The schema being declared may be a business object (BO) schema, a business service (BS) schema, a service script (SS) schema, data area (DA) schema or a UI map schema. The declare statement will differ based on the type of schema, but the syntax is analogous.

- **declareBO 'BO Name' | \$variable | "xpath" as 'DynamicDataArea';**
- **declareBS 'BS Name' | \$variable | "xpath" as 'DynamicDataArea';**
- **declareSS 'SS Name' | \$variable | "xpath" as 'DynamicDataArea';**
- **declareDA 'DA Name' | \$variable | "xpath" as 'DynamicDataArea';**
- **declareMap 'Map Name' | \$variable | "xpath" as 'DynamicDataArea';**

When invoking a BO, BS or SS, the name of the object can be specified as a literal or it can be a value contained within an element or a variable. For every Invoke, you must supply an XPath reference to a group name.

When invoking a business object, an action must be supplied. The syntax is **invokeBO 'BO Name' | \$variable | "xpath" using "xpath" for action**; The valid actions are as follows:

- **read.** This action reads the current view of the BO data.
- **add.** This action will add the object and read and return the resulting view of the BO.
- **fastAdd.** This action will add the object but does not perform a subsequent 'read' to return the resulting view of the BO. This option is better than add for performance purposes if there is no reason to re-read the record.
- **update.** This action will update the object and read and return the resulting view of the BO. This action executes a 'merge' of the information specified in the invoke statement's request XML document with existing BO data. Using this action allows the script to only indicate the elements that are changing.
- **fastUpdate.** This action will update the object but does not perform a subsequent 'read' to return the resulting view of the BO. This option is better than update for performance purposes if there is no reason to re-read the record.
- **delete.** This action deletes the object.
- **replace.** This action is an alternate to the update action. The replace action completely replaces existing BO data with the information in the request document. Typically, the replace action is used when a BO contains a list because it is easier to simply replace all instances of a list rather than attempt a list merge, which requires special logic to [delete a list instance](#) explicitly.

NOTE: The replace action must be used when using the UI map functionality to [Upload a CSV File](#).

NOTE: The replace action is currently not supported for any maintenance object that is maintained in a 'fixed' page that uses a list metaphor to show all the records in the page at once. Currency is an example of this type of page.

Examples:

```
invokeBO 'BusinessObject' using "dataArea" for fastAdd;
invokeBO $variableBO using "dataArea" for fastUpdate;
invokeBO "daName/boElement" using "dataArea" for replace;
```

The syntax of the invoke statements for both a business service and service script are similar. The BS / SS is specified along with the XPath reference to the group name:

- **invokeBS 'BS Name' | \$variable | "xpath" using "xpath";**
- **invokeSS 'SS Name' | \$variable | "xpath" using "xpath";**

The examples use the **invokeBS** statement but the statements are similar for the **invokeSS** statement.

```
invokeBS 'BusinessService' using "dataArea";
invokeBS $variableBS using "dataArea";
invokeBS "daName/bsElement" using "dataArea";
```

BO Warnings. Note that for BPA scripting, the **invoke** statements may also indicate how to handle warnings.

Syntax	Description	Examples
with warn asError	Indicates that a warning should be treated as an error displayed in the UI map. The text asError is optional.	<pre>invokeBO 'BusinessObject' using "dataArea" for add; invokeSS 'ServiceScript' using "dataArea" with warn asError;</pre>
with warn popup	Indicates that a warning should be presented in the standard framework popup. In this scenario, the user is presented with standard OK and Cancel buttons. If the user clicks OK, it means that the process should continue. If the user clicks Cancel, the processing should discontinue. This is the recommended setting.	<pre>invokeBS "daName/bsElement" using "dataArea" with warn popup;</pre>
with warn suppress	Indicates that a warning should be suppressed. This is the default if no warning syntax is added to the invoke statement.	<pre>invokeBS "daName/bsElement" using "dataArea" with warn suppress; invokeSS 'ServiceScript' using "dataArea";</pre>

NOTE: For service scripts, all objects invoked from the service script will inherit their warning level. Therefore, if the service script is invoked **with warn**, all nested invoke statements will also be invoked **with warn**.

For BPA scripts, there should also be logic following the invocation to handle errors and warnings (if **with warn as popup** is used). The system variable **\$WARNING** is set to **true** if the user has clicked the Cancel button in the warning popup. If a map was shown, the logic should redisplay the map (where the warning message will be redisplayed.) This allows the user to potentially make changes and save again. If no map was shown prior to the warning, then the logic should terminate.

The system variable **\$ERROR** indicates that an error was received. If a map was shown prior to the error, then the logic should redisplay the map where the error will be shown. If no map was shown, then the product provides a BPA Script **F1-HandleErr** that should be used to display the error. The following is an example of typical error handling logic.

```
invokeBO "F1-DetermineBo/output/bo" using "boSchema" for update with warn popup;
```

```

if ("WARNING")
  if ("map_schema/action = 'DEL'")
    terminate;
  else
    goto maintMap;
  end-if;
end-if;
if ("ERROR")
  if ("map_schema/action != 'DEL'")
    goto maintMap;
  else
    transferControl 'F1-HandleErr';
  end-if;
end-if;

```

System and Global Variables

The following tables highlight system and global variables available for script writing.

System Variables - All Script Types

The following system variables are available for all script types (service scripts, plug-in scripts, and BPA scripts).

Variable	Description	Example
\$BLANK	Represents an empty node.	<pre> if ("string(parm/spouse/name) != \$BLANK") goto addSpouse; end-if; </pre>
\$CURRENT-DATE	<p>Represents the current date.</p> <p>For BPA scripts, this is the browser date.</p> <p>For server scripts this is the server date (and is affected by the system date override logic).</p>	<pre> move "\$CURRENT-DATE" to \$tempDate; </pre>
\$CURRENT-STD-DTTM	Represents the current date-time expressed in standard time (meaning without any adjustments for summer time / daylight savings time).	<pre> move "\$CURRENT-STD-DTTM" to \$tempDateTime; </pre>
\$DEVICE-OS	Represents the user's device operating system.	<pre> move "\$DEVICE-OS" to \$tempDeviceOs; </pre>
\$DEVICE-BROWSER	Represents the user's device browser.	<pre> move "\$DEVICE-BROWSER" to \$tempDeviceBrowser; </pre>
\$DEVICE-DISPLAY-TYPE	Represents the user's device screen display type whether it is Desktop size or Medium or Small size. Returned values may be like oraDesktop, oraTablet and oraPhone.	<pre> move "\$DEVICE-DISPLAY-TYPE" to \$tempDeviceDisplayType; </pre>
\$DEVICE-INFO	Provides the combination of all three device properties (DEVICE-OS, DEVICE-BROWSER and DEVICE-DISPLAY-TYPE) and each property value is separated by semi-colon.	<pre> move "\$DEVICE-INFO" to \$tempDeviceInfo; </pre>

System Variables - BPA Scripts Only

The following system variables are only available / applicable for BPA script types.

Variable	Description	Example
\$DOUBLE_QUOTE	Represents a double quote.	<code>move "\$DOUBLE_QUOTE" to \$tempField;</code>
\$SINGLE_QUOTE	Represents an apostrophe.	<code>move "\$SINGLE_QUOTE" to \$tempField;</code>
\$SPACE	Contains a single space value.	<code>move "\$SPACE" to \$tempField;</code>
\$SYSTEM-DATE	Represents the server date. Note that this date is affected by the system date override logic)	<code>move "\$SYSTEM-DATE" to \$tempDate;</code>

System Variables - Server Scripts Only

The following system variables are only available / applicable for service script and plug-in script types.

Variable	Description	Example
\$ADDITIONAL-IP-INFO	An HTTP request includes an "additional IP address" header field. This may be populated by an implementation if there is some information available on the proxy server or load balancer, such as the originating IP address.	<code>move "\$ADDITIONAL-IP-INFO" to "parm/request/headerIpAddress";</code>
\$CURRENT-DTTM	Represents the current date-time.	<code>move "\$CURRENT-DTTM" to \$tempDateTime;</code>
\$F1-INSTALLATION-TIMEZONE	Represents the time zone code defined on the installation options .	<code>move "\$F1-INSTALLATION-TIMEZONE" to \$timeZone;</code>
\$LANGUAGE	Represents the language code the script is using. Typically this is the user's default language.	<code>move "\$LANGUAGE" to \$tempLanguage;</code>
\$PROCESS-DATE	Represents the process date. The process date differs from the current date because the process date will remain consistent throughout the duration of the process being executed. For example, if a service script stores several business objects – the process date is initialized at the start of the service script execution and each business object will have the same process date defaulted. The current date, especially the current date time, will reflect the actual time of processing.	<code>move "\$PROCESS-DATE" to \$tempDate;</code>
\$PROCESS-DTTM	Represents the process date-time. Note that the process date and time is initialized at the start of a particular process and will not reflect the exact date and time of an update.	<code>move "\$PROCESS-DTTM" to \$tempDateTime;</code>
\$REQUESTING-IP-ADDRESS	Represents the IP address from the HTTP request. Note that if the request is routed through a proxy server or load balancer, this IP address is be the IP address of the proxy or load balancer, not the IP address of the end user. Refer to the \$ADDITIONAL-IP-INFO variable for information.	<code>move "\$REQUESTING-IP-ADDRESS" to "parm/request/systemIpAddress";</code>
\$USER	Represents the user ID of the user executing the script.	<code>move "\$USER" to \$tempUser;</code>

Variable	Description	Example
		For BPA scripts that need to know the user ID of the current user, refer to Useful Tips and Tricks .

Global Variables

BPA scripts and service scripts have access to the values defined in [Global Context](#).

When a BPA script is launched from the user interface, these variables will be automatically initialized. They may be referenced with a single dollar sign in front of the field name. For example if PER_ID is a supported global variable, then \$PER_ID can be referenced within the BPA script:

```
move "$PER_ID" to "schema/customerId";
```

For service scripts, global variables may only be referenced if the service script has been invoked directly from a BPA script or a zone on a portal. When a service script is invoked from a BPA script or portal zone, it will have access to the suite of global context variables populated in the UI session. For service scripting, the global fields must be prefixed by two dollar signs (instead of one like in BPA scripting). For example if PER_ID is a supported global context variable, then \$\$PER_ID can be referenced within the service script.

```
move $$PER_ID to "schema/customerId";
```

NOTE: As described in [Context Variables](#), a service script may declare context variables that use the same two dollar sign syntax.

Perform Script and Transfer Control Statements

The edit data step supports functionality analogous to the [Perform script](#) step type and the [Transfer Control](#) step type. These are both applicable only to BPA scripts.

Syntax	Valid Values	Comments
performScript	'BPA Script Name'	Script to perform is explicitly provided.
	\$Variable	Script to perform is found in a variable.
	"XPath"	Script to perform is found in an element, referenced by its XPath.
transferControl	Analogous to the performScript statement	

NOTE: When the script named in the **performScript** statement has finished, control will be returned to the calling BPA script. When the script named in the **transferControl** statement has finished, you will not be returned to the calling script, complete control will be granted to the transferred to script.

Navigate Statement

The edit data step supports functionality analogous to the [Navigate to a page](#) step type. This is applicable only to BPA scripts.

Syntax	Valid Values	Comments
navigate	'Navigation Code'	Navigation option is explicitly provided.
	\$Variable	Navigation option is found in a variable.
	"XPath"	Navigation option is found in an element, referenced by its XPath.

In addition, the edit data step supports the ability to indicate that the dashboard should be refreshed when navigating. This is only applicable to BPA scripts.

Syntax	Valid Values
navigateAndReloadDashboard	Analogous to the navigate statement

Declare BO with BO Group

This statement is specific to BPA scripts that plan to use the base script Main BO Maintenance Processing (**F1–MainProc**) for its Generate Edit Map statements. This script expects that the data used to display in the map is within a **boGroup** tag.

Syntax	Valid Values	Comments
declareBOWithBOGroup	'BO Name'	BO is explicitly provided.
	\$Variable	BO is found in a variable.
	"XPath"	BO is found in an element, referenced by its XPath.

The following table highlights additional syntax for this statement.

Syntax	Valid Values
as	'Dynamic Schema Name'

Examples:

```
declareBOWithBOGroup 'BusinessObject' as 'newMapSchema';
declareBOWithBOGroup $variableBO as 'newMapSchema';
declareBOWithBOGroup "daName/boElement" as 'newMapSchema';
```

Invoke Map Statement

The edit data step supports functionality analogous to the [Invoke map](#) step type. This is applicable only to BPA scripts.

Syntax	Valid Values	Comments
invokeMap	'Map Name'	UI Map is explicitly provided.
	\$Variable	UI Map is found in a variable.
	"XPath"	UI Map is found in an element, referenced by its XPath.

The following table highlights additional syntax for this statement.

Syntax	Valid Values	Comments
using	"Data Area group name"	Indicates the data area to be passed to and from the server when rendering the HTML form associated with the map.
target	bpa	
	page	
	popup	

Refer to the [Invoke map](#) step type for more information about the **target** values.

If the UI map is configured to return a value, then it can be evaluated using the **\$MAP-VALUE** variable.

```
invokeMap 'UI Map' using "dataArea";  
invokeMap $variableMap using "dataArea";  
invokeMap "daName/mapElement" using "dataArea" target bpa;  
  
// $MAP-VALUE is a variable returned by the invoked map.  
if (" $MAP-VALUE='continue' ")  
    goto 300;  
else  
    terminate;  
end if;
```

Generate Edit Map Statements

The 'generate edit map' statements are used to dynamically generate and launch a UI edit map based on a schema definition. The schema used may be a BO schema, a BS schema, an SS schema or a DA schema. This is applicable only to BPA scripts. The generate statement will differ based on the type of schema, but the syntax is analogous.

Syntax

generateBOEditMap

generateBSEditMap

generateSSEditMap

generateDAEditMap

The BO code / BS code / SS code / DA code may be specified using a literal (surrounded by single quotes), as a temporary variable or by referencing an XPath schema location (surrounded by double quotes).

The following table highlights additional syntax for this statement.

Syntax	Valid Values	Comments
using	"Data Area group name"	Indicates the data area to be passed to and from the server when rendering the HTML form associated with the map.
target	bpa page popup	

The target values indicate where the generated map should be displayed as described in the [Invoke map](#) step type. If the UI map is configured to return a value, then it can be evaluated using the **\$MAP-VALUE** variable.

The examples use the **generateBOEditMap** but the statements are similar for the other schema types.

```
generateBOEditMap 'BO Name' using "dataArea";  
generateBOEditMap $variableMap using "dataArea";  
generateBOEditMap "daName/mapElement" using "dataArea" target bpa;  
  
// $MAP-VALUE is a variable returned by the invoked map.  
if (" $MAP-VALUE='continue' ")  
    goto 300;  
else  
    terminate;  
end if;
```

Terminate Statement

The edit data step supports functionality analogous to the [Terminate](#) step type.

The following is an example of a simple **terminate** step that will stop the script.

```
if ( "not (parm/spouse/name) " )
  terminate;
else
  goto addSpouse;
end-if;
```

The **terminate with error** statement is only available in a service script.

NOTE: For issuing an error from a BPA script refer to [Useful Tips and Tricks](#).

Syntax	Attributes	Comments
terminate with error (x, y %n= element=)	'x' represents the message category	Required.
	'y' represents the message number	Required.
	%n="Element XPath" or %n='literal'	Specify the substitution parameters supported by the message using either literal values or XPath references. When the value to substitute is an element within a list, use the XPath of the element within the list qualified by the list occurrence.
	element='Element XPath' or element=\$variable	Optionally specify an element name within a UI map to highlight as part of the error. When the element in error is within a list, use the XPath of the element within the list qualified by the list occurrence.

Example of a simple field:

```
if ("string(customer/lastName) = $BLANK")
  terminate with error (8000, 1001 %1="customer/
lastName" %2='Last name required' element='customer/lastName');
end-if;
```

Example of terminate where the element to mark is in a list:

```
for ($list in "parm/hard/newBusinessObject/listName")
  if /** check some condition for elementName
  terminate with error (11000, 11000 %1="$list/elementName" element='$list/elementName');
  end-if;
end-for;
```

FASTPATH: For more information on presenting errors in a UI map, please refer to [Display Errors](#).

Invoking Groovy Code

If you have a plug-in script or a service script that mixes XPath scripting and Groovy scripting, the edit data step supports the ability to invoke Groovy code using the syntax **invokeGroovy 'method'**; where 'method' is the name of a method defined in a **Groovy Members** step within the script. Only methods that do not receive arguments and return void can be invoked in this way. However, the method invoked from the edit data step can be supported by additional Groovy code in other **Groovy Members** step types.

Example Edit Data step:

```
invokeGroovy 'invoke';
```


Example Groovy Members step:

```
void invoke() {
    initParms()
    readBO()
    initConfig()
    retrieve()
    updateBO()
}
```

NOTE: As mentioned in [Using Groovy Within Scripts](#), if this is a plug-in script and the code is only Groovy, you should use the **Groovy** script engine version rather than the above techniqueLI. This is considered more efficient as it avoids the conversion of the data to and from the XML interface.

Debugging a BPA Script

If a BPA script has height greater than zero, then selected nodes of the script's data area can be displayed at runtime. The XML data is displayed during script execution within the BPA script's display area. Specify the XPath of an XML node from any of the BPA script's data areas, between the paired characters: '%+' and '+%'.

For example, the entire contents of the schema group node named 'input', and the specific contents of the schema element named 'output/status' will be displayed in the BPA script's display area. The debug text must be entered into the BPA script's text area and not within the script's edit data field. Debug text can be declared for any explicit step of the script.

```
display input: %+input+% , and output status: %+output/status+%
```

Useful Tips and Tricks

This section provides some suggestions about how to implement some common logic.

Finding the Logged In User in a BPA

For server based scripts, the **\$USER** variable is populated with the current logged in user. This variable is not available for BPA scripts. The BPA can call the service script **F1-GetUser** if this information is needed.

Issuing an Error from a BPA script

For server based scripts, you use the **terminate with error** statement to return an error message to display to the user. The BPA can display that error by transferring control to **F1-HandleErr** as described in the [Declare and Invoke Schema Based Objects](#). This is not supported in a BPA script. A technique that is available to a BPA for issuing an error is to call the business service **F1-RethrowError** passing the error details. This business service populates the appropriate system error fields needed to issue the error. Then transfer control to **F1-HandleErr**.

```
if ("string($assignedToUser) != string(F1-GetUser/user)")
    declareBS 'F1-RethrowError' as 'errorMsg';
    move '11010' to "errorMsg/messageCategory";
    move '11511' to "errorMsg/messageNumber";
    move "$toDoEntryId" to "errorMsg/messageParameters/+parameters/parameterValue";
    invokeBS 'F1-RethrowError' using "errorMsg";
    transferControl 'F1-HandleErr';
end-if;
```

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.

Height Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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.

Input Data Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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.

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. Refer to [BO warnings](#) for more information about handling warnings when invoking a BO.

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**.

FASTPATH: Refer to [BO Actions](#) for more information about the various actions.

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.

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.

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.

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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).

Invoke Map Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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.

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.

Groovy Imports Steps

NOTE: Server based scripts only. This step type is not applicable to BPA scripts.

Groovy Imports step types provide a free format text area where you list the Groovy classes to be imported for use by the code in **Groovy Member** steps within the script.

Enter a description of the imports step in the **Text** field. Click the adjacent icon to open a window providing more space for entering text.

Enter the list of classes to import in the **Edit Data Text** field using the syntax **import 'class'**; where 'class' is the fully qualified package name of the Groovy class. Click the adjacent icon to open a window providing more space for editing the text.

When saving a script, if the system has been configured accordingly, when the system detects references to a Groovy class that is not explicitly listed in this step, the system warns the user and adds an explicit import for that class. (If no Groovy Import step type exists for the script, one will be added.) Users should review the imported class to ensure it is the expected class, in the case of multiple classes with a similar name.

NOTE: For security, the classes that may be imported are restricted to those allowed by the Framework. The warning and automatic update to the script is controlled by a property setting. Refer to the topic [Using Groovy Within Scripts](#) for more information.

Groovy Library Interface Steps

NOTE: Server based scripts only. This step type is not applicable to BPA scripts.

Groovy Library Interface steps are only applicable to **Groovy Library Scripts**. They provide a free format text area where you can list the Groovy methods defined within the script that are available for use by other scripts.

Enter a description of the interface step in the **Text** field. Click the adjacent icon to open a window providing more space for entering text.

Enter the list of interface methods in the **Edit Data Text** field. Click the adjacent icon to open a window providing more space for editing the list.

NOTE: Every **Groovy Library Script** must have one and only one step of type **Groovy Library Interface**. The supporting code for the available methods is defined using one or more **Groovy Member** steps in the same script. Refer to the topic [Using Groovy Within Scripts](#) for more information.

Groovy Member Steps

NOTE: Server based scripts only. This step type is not applicable to BPA scripts.

Groovy Member steps provide a free format text area where you can enter Groovy code.

Enter a description of the code block in the **Text** field. Click the adjacent icon to open a window providing more space for entering text.

Enter your code in the **Edit Data Text** field. Click the adjacent icon to open a window providing more space for editing the code.

NOTE: While it is possible to set up multiple steps of type **Groovy Member** the system treats these steps as a single class for compilation and execution purposes. Refer to the topic [Using Groovy Within Scripts](#) for more information.

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.

Mathematical Operation Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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**.

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. This is only applicable to Service Scripts and Plug-in Scripts.
- **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**. This is only applicable to BPA Scripts.
- **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 when the source value is a [global context variable](#). This is only applicable to BPA Scripts.
- **Page Data Model.** Use this field type 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**. This is only applicable to BPA Scripts.
- **Portal Context.** Use this field type when the source value is a variable in the portal context. This is only applicable to BPA Scripts.
- **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**. This is only applicable to BPA Scripts.

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. This is not applicable to BPA Scripts.
- **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**. This is only applicable to BPA Scripts.
- **Portal Context.** Use this field type when the destination to be updated is in the current portal context. This is only applicable to BPA Scripts.
- **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**. This is only applicable to BPA Scripts.

NOTE: Conditional field types. The field types **Current To Do Information**, **Page Data Model** and **User Interface Field** are only applicable to BPA scripts.

Navigate To A Page Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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](#).

Perform Script Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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:

Script is the name of the script that is performed.

Press A Button Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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.

Prompt User Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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.

Set Focus To A Field Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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.

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 [Edit Data Syntax](#) the actual syntax of initiating an error message.

NOTE: The ability to terminate a step in error is only supported for server-based scripts.

Transfer Control Steps

NOTE: BPA scripts only. This step type is only applicable to BPA scripts.

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.

Additional Topics

The contents of this section provide additional information about steps.

How To Find The Name Of User Interface Fields

This section is only relevant if you are using the legacy capability of BPA scripts to bring a user to a base fixed page and populate information into fields on that page. Follow these steps to find the name of a field that resides on a fixed page:

- Navigate to the page in question.
- Right click in the body of the page.
- Select **Inspect** from the pop-up menu to display the HTML.
- In the search look for the text "Widget Info". Note that if there are multiple sections on the page, you will find multiple "Widget Info" results. Based on the details shown, you are able to confirm which section you are looking for. For example, the following is an example from the Algorithm - Main page:

```

* Widget Info:
*   Widget_ID , Element Type - label info - label
*   ALG_CD, IT - CI_ALG$ALG_CD - Algorithm Code
*   IM_ALG_CD, IM - $SEARCH_FOR_ALG_LBL - Search for Algorithm
*   OWNER_FLG, IS - $OWNER_FLG - Owner
*   DESCR50, IT - $DESCR50 - Description
*   IM_Main_algtypCont, IM - $F1_SHOW_ALG_CTX - Show Algorithm Context Menu
*   ALG_TYPE_CD, IT - $ALG_TYPE_CD - Algorithm Type
*   IM_ALG_TYPE_CD, IM - $SEARCH_ALGTYPE_LBL - Search for Algorithm Type
*   ALG_TYPE_DESCR50, PL - $DESCR50 - Description
*   ALG_TYPE_LONG_DESC, IA - $DESCRLONG - Detailed Description
*   IM_VerScroll_lftArrow, IM - $PREVIOUS_ALG_LBL - Previous Version
*   ALG_VER$recordCount, SN - $OF_LBL - Of
*   IM_VerScroll_rtArrow, IM - $NEXT_ALG_LBL - Next Version
*   IM_ADD, IM - $ADD_ALG_LBL - Add Version
*   IM_DEL, IM - $DELETE_ALG_LBL - Delete Version
*   ALG_VER$EFFDT, IT - $EFFDT - Effective Date
*   L_OWNER_LBL, LP - $OWNER_FLG - Owner
*   ALG_VER$OWNER_FLG, IS - $OWNER_FLG - Owner
*   resultGrid, GD

```

The field names that you'll reference in your scripts are defined on the left side of the HTML (e.g., ALG_CD, DESCR50, ALG_TYPE_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 version scroll that appears on Algorithm - Main. Notice that the fields in the scroll are prefixed with the name of the scroll plus a \$ sign. For example, the effective date is called **ALG_VER\$EFFDT**.

```

*   IM_VerScroll_lftArrow, IM - $PREVIOUS_ALG_LBL - Previous Version
*   ALG_VER$recordCount, SN - $OF_LBL - Of
*   IM_VerScroll_rtArrow, IM - $NEXT_ALG_LBL - Next Version
*   IM_ADD, IM - $ADD_ALG_LBL - Add Version
*   IM_DEL, IM - $DELETE_ALG_LBL - Delete Version
*   ALG_VER$EFFDT, IT - $EFFDT - Effective Date
*   L_OWNER_LBL, LP - $OWNER_FLG - Owner
*   ALG_VER$OWNER_FLG, IS - $OWNER_FLG - Owner

```

The names of fields that reside in grids are in a slightly different format. The following is an example of the HTML for the algorithm parameters grid that appears on Algorithm - Main. Notice that the fields in the grid are prefixed with the name of the grid plus a :x\$. For example, the algorithm parameter label is called **ALG_PARM:x\$ALG_PARM_LBL**. 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
*   ALG_PARM:x$ALG_PARM_LBL, IL - CI_ALG_TYPE_PRM_L$ALG_PARM_LBL - Parameter
*   ALG_PARM:x$SEQNO, IL - $SEQNO - Sequence
*   ALG_PARM:x$ALG_PARM_VAL, IA - $ALG_PARM_VAL - Value
*

```

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

This section is only relevant if you are using the legacy capability of BPA scripts to bring a user to a base fixed page and perform functionality by executing existing buttons on that page.

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:

Button Name

IM_GOBACK

IM_HISTORY

IM_GOFORWARD

IM_menuButton

IM_USER_HOME

IM_MY_PREF

IM_helpButton

If you want a **Press a button** step to press a button in the page actions toolbar, use one of the following names:

Button Name

IM_SAVE

IM_REFRESH

IM_CLEAR

IM_COPY

IM_DELETE

IM_ScrollBack

IM_ScrollForward

The following buttons are also supported:

Button Name

IM_TO_DO

IM_CurrentTo Do

Comments

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.

Button Name	Comments
IM_MINIMIZE_DASHBOARD	Pressing this will collapse the dashboard. Note that when a script is finished, it will return the dashboard to the state it was when the script was launched.
IM_MAXIMIZE_DASHBOARD	Pressing this will expand the dashboard. Note that when a script is finished, it will return the dashboard to the state it was when the script was launched.

Follow these steps to find the name of other buttons that reside in the object display area of a fixed page:

- Navigate to the page in question.
- Right click in the body of the page
- Select **Inspect** from the pop-up menu.
- In the search look for the text "Widget Info". Note that if there are multiple sections on the page, you will find multiple "Widget Info" results. Based on the details shown, you are able to confirm which section you are looking for.
- 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 Algorithm - Main page (notice the **IM** fields for the iconized buttons).

```
* Widget Info:
*   Widget_ID , Element Type - label info - label
*   ALG_CD, IT - CI_ALG$ALG_CD - Algorithm Code
*   IM_ALG_CD, IM - $SEARCH_FOR_ALG_LBL - Search for Algorithm
*   OWNER_FLG, IS - $OWNER_FLG - Owner
*   DESCR50, IT - $DESCR50 - Description
*   IM_Main_algtypCont, IM - $F1_SHOW_ALG_CTX - Show Algorithm Context Menu
*   ALG_TYPE_CD, IT - $ALG_TYPE_CD - Algorithm Type
*   IM_ALG_TYPE_CD, IM - $SEARCH_ALGTYPE_LBL - Search for Algorithm Type
```

- Transaction-specific actions buttons are represented as switches. The following is an example of the HTML on the Batch Job Submission - 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.

```
*   ACTION_Q_SW, BU - $ACTION_Q_SW - Duplicate & Queue
*   ACTION_CANCEL_SW, BU - $CANCEL_SW - Cancel
```

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:

- `
` causes a line break in a text string. If you use `

` a blank line will appear.
- ` text ` 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.

NOTE: Refer to [Color Contrast](#) for information about the use of the HTML color 'red' and its impact on accessibility.

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

Some explicit 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.

Note that when using the Edit Data step type, the variable available are slightly different. Refer to [Edit Data Syntax](#) for details.

The following global variables exist for BPA scripts:

Variable Name	Comments
<code>%PARAM-<name></code>	This 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.
<code>%PARAM-NOT-SET</code>	This is to be used to compare against <code>%PARAM-< ></code> parameters to check if the parameter has been set or not when the application was launched. A parameter that has not been set would be considered equal to this global variable. It is recommended to compare parameters against this global variable before using them for the first time.
<code>%BLANK</code>	A constant that contains a blank value (no value).
<code>%SPACE</code>	A constant that contains a single space value.
<code>%CURRENT-DATE</code>	The current date as known by the browser, not the server.
<code>%SYSTEM-DATE</code>	The server date. Note that this date is affected by the system date override logic)
<code>%SAVE-REQUIRED</code>	A flag that contains an indication of whether the data on a page has been changed (and thus 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 .
<code>%NEWLINE</code>	A constant that contains a new line character (carriage return). Upon substitution, a line break is inserted in the resultant text. NOTE: This constant 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.

To refer to a [global context](#) variable, use %FIELD_NAME. For example, if the field SP_ID is in the global context, you may reference %SP_ID to reference the ID of the service point currently in context. In addition, the following special values are supported:

Variable Name	Comments
%CONTEXT-PERSONID	A constant that contains the ID of the current person.
%CONTEXT-ACCOUNTID	A constant that contains the ID of the current account.
%CONTEXT-PREMISEID	A constant that contains the ID of the current premise.

In addition, if the script is invoking something else via one of the various “Invoke” step types and an error is returned, the following global variables contain information about the error:

Variable Name	Comments
%ERRMSG-CATEGORY %ERRMSG-NUMBER	The unique identifier of the error message number.
%ERRMSG-TEXT	The brief description of the error.
%ERRMSG-LONG	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_FWDTTM_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 "F1-BatchControl/batchControl" to reference a top-level "batchControl" element in a script data area called "F1-BatchControl".

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 Engine Version 2 and Above Notes

Scripting using the engine version 2 or above requires some extra syntax to take advantage of XPath 2 functionality. In general, any variable declared will be assumed to be a string. This means, that if you intend to construct a mathematical statement then it is necessary to explicitly declare the data type of variables as integers, numbers, or dates.

NOTE: Unless otherwise noted, all XPath examples in this topic are for the Version 1 engine – which means XPath 1. Statements that function using XPath 1 will not necessarily work for XPath 2. This is especially true when executing math, see below for examples.

Date and Time Arithmetic

XPath date/time and interval data types support arithmetic operations ('+', '-', '*' etc.) and functions, which can be used for time calculations in the same way as '1 + xs:integer(value)' is used for numeric calculations.

Compare time duration:

```
if ("xs:dateTime(fn:current-dateTime()) - xs:dateTime($updateDateTimeX))
  ge xs:dayTimeDuration(concat('PT', BO/hoursBetweenStatisticsUpdate, 'H'))")
  goto 60;
end-if;
```

Compare one date to another:

```
if ("xs:date(parm/endDate) < xs:date(parm/startDate)")
  terminate with error (11108, 11507 element='endDate');
end-if;
```

Compare a date against today's date:

```
if ("xs:date(parm/startDate) <= xs:date($CURRENT-DATE)")
  terminate with error (11108, 11507 element='endDate');
end-if;
```

Calculate the end of month:

```
// covert to ISO
move "concat($year, '-', $mon2, '-01T00:00:00')" to $monthStart;

// calculate
move "xs:dateTime($monthStart) + xs:yearMonthDuration('P1M') - xs:dayTimeDuration('P0DT1S')"
  to $monthEnd;

// convert from ISO to OUAF
move "concat($year, '-', $mon2, '-', substring(string($monthEnd), 9, 2), '-23.59.59')" to $endDateTime;
```

NOTE: XPath date/time/interval formats use the ISO standard, which needs to be converted to/from formats supported in the framework.

Comparing Date/Times in String Format

Any ISO-like string format for date/time preserves the YYYY MM DD HH MM SS sequence, which is zero-padded. Regardless of separators, this format will remain appropriate for comparison operations. In particular, date/time values in the framework format “YYYY-MM-DD.HH.MM.SS” can be used with “=”, “!=”, as well as “>”, “>=”, “<”, “<=” operators.

```
// retrieve framework date/time value
invokeBS 'CM-MAXMSRMT' using "CM-MAXMSRMT";
move "string(cm-MAXMSRMT/results[1]/measurementDateTime)" to $lastMsmtDT;

// construct another date/time
move "concat($year, '-01-01-00.00.00')" to $startDateTime;

// compare using string operators
if ("{$lastMsmtDT} >= {$startDateTime}")
    move "substring($lastMsmtDT,1,4)" to $latestMsmtYear;
```

Converting Date/Times Between Framework and ISO

Conversion of date/time from framework format to ISO is only necessary for date/time arithmetic. Comparisons can be done with the framework format directly. The only difference between the framework format and ISO date/time formats is in the separators:

Framework: “YYYY-MM-DD.HH.MM.SS”

ISO: “YYYY-MM-DDTHH:MM:SS”

Example of converting from the framework format to ISO:

```
move "concat(substring($ouafDT, 1, 10), 'T', translate(substring($ouafDT, 12), '.', ':'))" to $isoDT;
```

Example of converting from ISO to the framework format:

```
move "concat(substring($isoDT, 1, 10), '.', translate(substring($isoDT, 12), ':', '.'))" to $ouafDT;
```

Round Money With a Dynamic Currency Scale

Because different currencies support a different number of decimals, the framework provides an API for rounding a monetary amount based on a given currency.

```
move "parm/amount" to $qnty;
move "currency/decimals" to $decimals;
move "fn:round(xs:decimal($qnty) * math:exp10(xs:double($decimals)))
    div math:exp10(xs:double($decimals))" to "parm/roundedAmount";
```

Looping through Sequences

In XPath 2 it is possible to organize a for-loop over a sequence of integers, not only a node list.

This example shows a loop over a range of months. This is a sequence-forming construct in XPath. The XPath node list, which we are familiar with, is just another type of sequence.

```
for ($month in "1 to 12")
```

This example shows a loop over a give range of years in descending order:

```
for ($year in "fn:reverse(parm/startYear to parm/endYear)")
    move "concat($year, '-01-01-00.00.00')" to $startDateTime;
    move "concat($year, '-12-31-23.59.59')" to $endDateTime;
    ...
```

This example shows a loop through a node list using ‘index’, so that other node lists can be accessed:

```
for ($idx in "1 to count(parm/touData/touList)")
```

```
move "parm/touData/touList[$idx]" to $tou; // access any list with this index
```

The above syntax can be used as an elegant alternative to maintaining indices separately, for example instead of the following:

```
move "0" to $idx;  
for ($item in "parm/touData/touList")  
    move "1 + xs:integer($idx)" to $idx;
```

String Padding and Decimal Formatting

This is used with specific input formats or output formatting. It is applicable to zero, space and other types of padding.

This example shows prefixing for date/time components, for example producing “2010-01-02” instead of “2010-1-2”.

```
move "substring(concat('0',string($month)), string-length(string($month)), 2)" to $mon2;
```

This example shows suffixing for adding decimal zero-padded alignment, for example producing “12.30” and “4.00” instead of “12.3” and “4”. The example performs 3 tasks: rounding to 2 decimals, inserting a period if necessary, and zero padding.

```
// round and zero-pad to 2 decimals  
move "$item/amount" to $qty;  
move "fn:round(xs:double($qty) * 100) div 100" to $qty;  
move "string($qty)" to $qty;  
move "concat(substring-before(concat($qty, '.'), '.'), '.', substring(concat(substring-  
after($qty, '.'), '00'), 1, 2))" to $qty;
```

Ternary Operation

This makes a choice between values based on a condition, so that it could be used in a single expression instead of an if/else block. It is known in C/C++ as ‘cond ? value1 : value2’ or in BASIC as ‘IFF(cond, value1, value2)’. In XPath the syntax is: “if (cond) then value1 else value2”. Note this is not the top-level scripting if-statement block.

In XPath this is an expression, which can be combined with other expressions. In scripting it can be used as:

```
move "if (string(D1-UnitOfMeasure/  
measuresPeakQuantity) = 'D1MP') then 'D1MX' else 'D1SM' " to $func;
```

Pipeline Processing

In scripting, it is not easy to create a simple reusable piece of code as there are no local functions, and a separate script call is a coding overhead and requires packing/unpacking parameters. To avoid copying and pasting the same code block between similar script stages, consider ‘pipelining’, which is breaking the overall process into separate top-level steps, some of which could be shared between alternating paths. This is common for parameter preparation and output formatting. An intermediate result between stages can be stored in a “parm” substructure.

Instead of this code:

```
if ("type = A")  
    prepare params ...  
    call services for A ...  
    format output ...  
end-if;  
if ("type = B")  
    prepare params ...  
    call services for B ...  
    format output ...  
end-if;
```

Consider this alternative:

```
prepare params ...  
if ("type = A")  
    call services for A ...  
end-if;  
if ("type = B")  
    call services for B ...  
end-if;  
format output ...
```

XPath 2 Functions

Script engine versions 2 and above support XQuery 1.0 Functions and Operators, and the XQuery 1.0 standard itself with some minor limitations. Below are the URLs to both specifications. The first link has the functions/operators available to use from XQuery.

- <http://www.w3.org/TR/xpath-functions/>
- <http://www.w3.org/TR/xquery/>

The following can only access local file systems. (For other protocols like http they will return an empty sequence):

- fn:doc
- fn:collection

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.

NOTE: Conditional tab page. This tab page does not appear for [Groovy Library scripts](#) or [plug-in scripts](#) using the **Groovy** engine version.

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 only for plug-in scripts using a script engine version. 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: Refer to [Schema Nodes and Attributes](#) for 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. 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.

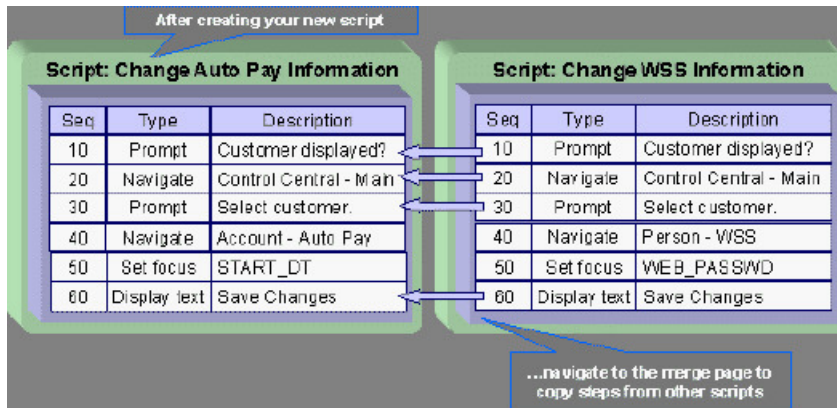
- Some products may also include an option to choose **Characteristic**. Choosing this option displays adjacent fields to define the object on which the characteristic resides and the characteristic type. The objects whose characteristic values may be available to choose from depend on your product.
- 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.
 - 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.




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

	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'
↔	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 a page - userGroupAppService

Remove Merged Step - Script Merge - row 1

	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'
↔	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: Only page services can be invoked.

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 > System > Function** 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.

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: Fields in multiple lists. 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. 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']`.

Maintaining Managed Content

The Managed Content object is used to store various types of text content. Some content is used internally such as Allowlists. Other content may be used by implementations such as XSL files used for communicating messages with an external system, JavaScript include files, and CSS files. These files may then be maintained in the same manner as the HTML in UI Maps.

Open this page using **Admin > System > Managed Content**.

Use the **Managed Content Query** portal to search for an existing managed content entry. Once the managed content is selected, you are brought to the maintenance page to view and maintain the selected record.

Most fields on the page are self explanatory.

Use **Managed Content Type** to indicate the type of content.

- **Allowlist**

is used to capture a list of supported values for given functionality. For example, the product provides allowlists for SQL functions and valid HTML tags.

- **XSLTC** is used to reference XSL files. These may be used internally for user interface referencing. Additionally, the XSLs may be referenced for sending messages to external system using [outbound message](#) and for receiving messages in SOAP as [inbound messages](#).

NOTE: The value FIX (XSLT) is no longer supported and will be deprecated in a future release. Other managed content values are not currently supported by any current framework functionality and are reserved for future use.

Use the **Schema** tab to view and maintain the contents of the managed context text.

Defining Entity Tags

This portal is used to maintain an entity tag along with the list of entities associated with it. Entity tagging supports grouping of configuration, demo and test related entities so they can better describe a logical function. Each group is represented by a unique tag name and entities may be associated with multiple tags.

An entity tag may explicitly restrict the type of entities it can be associated with by referencing a specific maintenance object as part of the entity tag record.

You can access the portal from the **Admin Menu > Implementation Tools > Entity Tag**. You are brought to a query portal with options for searching for a specific tag name. Once an entity tag has been selected you are brought to the maintenance portal to view and maintain the selected record.

A [Tags](#) dashboard zone allows you to review, add and remove tags for the current entity you review. For configuration type of entities, you may use this portal to tag multiple entities at once.

The following zones may appear as part of the portal's **Main** tab page

- **Entity Tag.** This zone provides information about the selected entity tag.
- **Tagged Entities.** This zone lists entities associated with the selected entity tag. You may also use the zone to remove entities from the list.
- **Tag Entities.** Use this zone to add configuration entities to the selected tag.

Advanced Schema Topics

The topics in this section describe some advanced information related to schemas used for business objects, business services, service scripts and UI maps.

Schema Nodes and Attributes

For business object definition, the purpose of the schema is to create a link between the schema and a maintenance object. For business service definition you are specifying the link between the schema and a service (either a general service, search service, or a maintenance object service). For service script definition, you are defining the API for passing information to and from the script. The following documentation is a complete list of the XML nodes and attributes available to you when you construct a schema.

Contents

- [The Four Element Types](#)
- [The Data Type of a Field Element](#)
- [Referencing Other Elements](#)
- [Standard Time Considerations](#)
- [The Mapping Attributes](#)
- [Descriptive Attributes](#)
- [Schema Constants](#)
- [Defaulting and System Variables](#)
- [The Flattening Nodes and Attributes](#)
- [Search Zone](#)
- [Extend Security for Service Script](#)
- [Overriding Action for a Business Service](#)
- [Specifying searchBy for a Search Service](#)
- [Including Other Schemas](#)
- [Compatibility Attributes](#)

The Four Element Types

A schema element can be one of four different structure types. Note that there are two classes of element types: the structural nodes group and list, and the data containing nodes of field and raw.

Mnemonic	Valid Values	Description	Examples
type=	"field"	The field type is the default type for any element not explicitly labeled as something other than a field. Therefore, you virtually never have to explicitly label an element as a field. Note that a field element, unlike group or list, will contain information in its nodes - rather than other nodes.	
	"group"	<p>The group element is typically a structural element of the schema only, in which case it has no mapping.</p> <p>Note that when grouping several elements that are all used to map an XML structure of a CLOB / XML field of a business object driven record, the mapping may be at the group level.</p>	<p>Example where a group is used to create a structure</p> <pre><schema> <input type="group" <userId/> </input> <output type="group"> <firstName/> <lastName/> </output> </schema></pre> <p>Example where the group includes the mapping:</p> <pre><parameters type="group" mapXML="BO_DATA_AREA" mdField= "F1_TODOSUMEMAIL_PARM_LBL" / > <numberOfDays mdField="F1_NBR_DAYS" required="true" /> <frequency mdField="F1_FREQUENCY" / ></pre>
	"list"	The list node is structural node like the group node. The only difference is that the list structure	<p>Example of a schema with a list:</p> <pre><schema> <statesList type="list"></pre>

Mnemonic	Valid Values	Description	Examples
		has the ability to repeat multiple times in an XML document.	<pre><state isPrimeKey="true" /> </statesList> </schema></pre> <p>Example of a schema with a list:</p> <pre><xml> <statesList> <state>AK</state> <description>Alaska</description> </statesList> <statesList> <state>AL</state> <description>Alabama</description> </statesList> ... </xml></pre>
	"raw"	The raw data type is used to capture a chunk of raw text that doesn't have any inherent structure associated with it.	<pre><sendDetail type="raw" /></pre> <p>Example of an XML instance for the above schema:</p> <pre><sendDetail> <messageInfo> <senderAddress>123 W. Main St, </senderAddress> <corpZone>3A</corpZone> </messageInfo> </sendDetail></pre>

The Data Type of a Field Element

Of the four different element types, only a field can have a data type.

Mnemonic	Valid Values	Description	Examples
dataType=	"string"	By default, a field element is a string. Therefore, there is no requirement to specify the string data type.	<pre><schema> <custName dataType="string" /> </schema></pre>
	"number"	<p>Defines an element that is a number.</p> <p>NOTE: UI hints include a setting to Suppress Automatic Number Formatting.</p> <p>NOTE: Use currencyRef attribute for auto-display of currency symbol that is associated with the referenced currency code. The currency decimal positions are ignored by this formatting allowing you</p>	<p>Examples</p> <pre><schema> <count dataType="number" /> </schema></pre> <pre><schema> <taxRate dataType="number" currencyRef="currency" /> </schema></pre>

Mnemonic	Valid Values	Description	Examples
		to display a currency symbol for a unit rate with many decimals.	
"money"	Optional additional attributes currencyRef ="element name"	Defines an element that represents a monetary amount. The currency reference is optional and if left blank the installation currency will be used. Automatic formatting and validation to be applied based on the currency. For example, the currency symbol will be shown when auto-rendering. In addition, the number of decimal places must not exceed the valid number defined for the currency. NOTE: Refer to Referencing Other Elements for supported syntax for referring to other elements.	<pre><schema> <currency default="USD" suppress="true" /> <balance dataType="money" currencyRef="currency" / > </schema></pre>
"lookup"	Required additional attribute lookup ="field name"	Defines an element that has valid values defined using a lookup. The lookup field is required.	<pre><schema> <status dataType="lookup" lookup="STATUS_FLG" /> </schema></pre>
"lookupBO"	Required additional attribute lookupBO ="bo name"	Defines an element that has valid values defined using an extendable lookup. The extendable lookup's business object is required.	<pre><schema> <category dataType="lookupBO" lookupBO="CM- BusinessCategory" /> </schema></pre>
"boolean"		Defines an element that has values of "Y" and "N".	<pre><schema> <allowsEdit dataType="boolean" / > </schema></pre>
"date"		Defines an element that represents a date.	<pre><schema> <startDate dataType="date" / > </schema></pre>
"dateTime"		Defines an element that represents a date and time. NOTE: Refer to Standard Time Considerations for additional configuration available for date / time fields that represent standard time.	<pre><schema> <startDateTime dataType="dateTime" /> </schema></pre>
"time"		Defines an element that represents a time.	<pre><schema> <startTime dataType="time" / > </schema></pre>
"uri"		Defines an element captures a URI. Elements defined with	<pre><schema></pre>

Mnemonic	Valid Values	Description	Examples
		this type enable the support for URI Allowlist and Substitution as described in Referencing URIs .	<pre><exportDirectory dataType="uri" /> </schema></pre>

Referencing Other Elements

There are several attributes that allow for a reference to another element in the same schema. The supported syntax of the XPath reference is the same in every case. This section provides examples using the default reference attribute (**defaultRef**).

Reference a sibling element:

```
<schema>
  <id mapField="ACCT_ID" required="true" />
  <altId defaultRef="id" required="true" />
</schema>
```

Reference an element in a higher group:

```
<schema>
  <id mapField="ACCT_ID" required="true" />
  <msgInfo type="group" mapXML="XML_FIELD">
    <altId defaultRef="../id" required="true" />
  </msgInfo>
</schema>
```

Reference an element in a lower group:

```
<schema>
  <id mapField="ACCT_ID" defaultRef="msgInfo/altId" required="true" />
  <msgInfo type="group" mapXML="XML_FIELD">
    <altId required="true" />
  </msgInfo>
</schema>
```

Reference an element in another group:

```
<schema>
  <acctInfo type="group">
    <id mapField="ACCT_ID" required="true" />
  </acctInfo>
  <msgInfo type="group" mapXML="XML_FIELD">
    <altId defaultRef="../acctInfo/altId" required="true" />
  </msgInfo>
</schema>
```

Standard Time Considerations

Most date / time fields represent "legal" time such that if a time zone changes their clocks for winter and summer time, the date / time field captures the current observed time. However, some date / time fields should always be captured in standard time to avoid confusion / ambiguity. A good example is a date and time related to detailed interval data. Refer to [Designing Time Zones](#) for more information.

When defining an element with **dataType="dateTime"**, you may optionally configure **stdTime="true"** indicating that data captured in the element always represents standard time in the 'base' time zone. The 'base' time zone is specified on the [Installation options](#).

NOTE: If an element is mapped to a table / field with a Standard Time Type of **Physical**, then **stdTime="true"** is implied. Refer to [Fields](#) for more information.

Example:

```
<schema>
  <startTime dataType="Time" stdTime="true" />
</schema>
```

If the time zone that represents the date / time field is not the installation time zone, use the optional setting **stdTimeRef**="XPath to time zone element" on a date / time element to indicate that the element represents standard time and indicates the time zone to use. Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements. .

Example:

```
<schema>
  <alternateTimeZone fkRef="F1-TZONE" suppress="true"/>
  <startDateTime dataType="dateTime" stdTimeRef="alternateTimeZone" />
</schema>
```

NOTE: If an element is mapped to a table / field with a Standard Time Type of **Referenced**, then **stdTime**="XPath" is implied. Refer to [Fields](#) for more information.

NOTE: When schema elements are captured in standard time the UI map supports HTML notation to automatically display the data applying a daylight savings time / summer time correction. Refer to the HTML attribute **oraType**="dateTime; stdTime:true" for more information.

There may be cases where the date / time is captured as standard time in one time zone, but should be displayed using a different time zone. In this case, the attribute **displayRef**="XPath" may be used in addition to the appropriate attribute that identifies the time zone that the data is capture in. Refer to [Referencing Other Elements](#) for supported syntax.

```
<schema>
  <displayTimeZone fkRef="F1-TZONE" suppress="true" />
  <startDateTime dataType="dateTime" stdTime="true" displayRef="displayTimeZone" />
</schema>
```

The Mapping Attributes

When constructing your schema, you can choose from one of the following mapping attributes.

Mnemonic	Valid Values	Description	Examples
mapField=	"field name"	In the case of a business object, the mapField attribute is used to identify the database column the element is related to. For business service schemas, the mapField= attribute is used to link a schema element with a service element.	<pre><schema> <factId mapField="FACT_ID" /> </schema></pre>
mapChild=	"table name"	<p>The mapChild attribute is used only for business object mapping. It is used in two ways:</p> <ul style="list-style-type: none"> First, to create a list in the business object that corresponds to a child table of an MO. Second, you can use mapChild to identify the child table a flattened field lives in. For more information on flattening, refer to flattening section below. 	<p>Example of a list within a child table in a BO:</p> <pre><persons type="list" mapChild="CI_ACCT_PER" <personId mapField="PER_ID" / > </persons></pre>

Mnemonic	Valid Values	Description	Examples
mapList=	"list name"	<p>The mapList attribute is used only for business service mapping. It is used in two ways:</p> <ul style="list-style-type: none"> First, to create a list in the business service that corresponds to a list in the service. Second, you can use mapList to identify the list that a flattened field lives in. For more information on flattening, refer to flattening section below. 	<p>Example of a list within a business service:</p> <pre><selectList type="list" mapList="DE" <value mapField="COL_VALUE"> <row mapList="DE_VAL"> <SEQNO is="2"/> </row> </value> </selectList></pre>
mapXML=	"field name"	<p>The mapXML attribute is typically used to map XML structures into a large character / XML field of the service. Note that when you use mapXML to map either a list or group structure (type="list" or type="group") you don't have to map all the child elements within the structure. It is also possible to map list elements to a large character field associated with the list child.</p>	<pre><enrollmentRequest type="group" mapXML="CASE_CLOB"> <messageID/> <sender/> </enrollmentRequest> <enrollmentKey mapXML="CASE_CLOB" /> <enrollmentInfo type="list" mapChild="CI_CASE_CHILD"> <sequence mapField="CHILD_SEQ" /> <name mapXML="CASE_CHILD_CLOB" / > <value mapXML="CASE_CHILD_CLOB" / > </enrollmentInfo></pre>
isPrimeKey=	"true"	<p>You must specify a primary key for a list defined within a mapped XML element (type="list" mapXML="CLOB"). The primary key is used by the framework to determine whether a list element add, update or delete is required during a business object update.</p> <p>NOTE: You do not need to specify the prime key for a business object list mapped to maintenance object list. When a physical list is mapped, the prime key is derived from existing physical meta-data.</p>	<pre><questionnaire type="list" mapXML="CASE_CLOB"> <question isPrimeKey="true" /> <answer/> </questionnaire></pre>
orderBy=	"XPath asc desc, XPath asc desc"	<p>By default, a list defined within a mapped XML element (type="list" mapXML="CLOB") is sorted by the first element of the list. A different sort order may be specified using the orderBy attribute. The attribute value is a comma separated list of</p>	<pre><questionnaire type="list" orderBy="page/section, page/sequence" mapXML="CASE_CLOB"> <question isPrimeKey="true" / > <answer/> <page type="group"> <section/> <sequence/> </page> </questionnaire></pre>

Mnemonic	Valid Values	Description	Examples
		field XPath(s) (relative to the list element) with an optional sort order (ascending is the default).	
		NOTE: This is only available for lists mapped as XML within business objects.	

Descriptive Attributes

The following attributes can be used to describe a schema element and provide additional configuration related to the element. Typically, these attributes are useful for field elements only.

Mnemonic	Valid Values	Description	Examples
<code><!-- comment --></code>		Use this to add a comment to a schema by using special open and close characters: <code><!--</code> and <code>--></code> .	<pre><schema> <!-- This schema is used to capture business information only, please refer to the 'HUMAN' BO for person information --> </schema></pre>
<code>description=</code>	"text"	The description of an element may be used to provide an internal description of the element to help a reader of the schema to understand the business reason for the element.	<pre><schema> <active type="boolean" description="active account" label="Active" /> </schema></pre>
<code>label=</code>	"text"	The label of an element is meant to be a short bit of verbiage that would typically precede the element in a user interface.	<pre><schema> <active type="boolean" description="active account" label="Active" /> </schema></pre>
<code>required=</code>	"true"	Used to require the existence of an element during object interaction. NOTE: For included schemas, the <code>required="true"</code> attribute is not processed on business object and business service schemas when they are included within a service script schema. This is to support the ability of the service script to populate required elements before an embedded business object or business service is invoked from the service script.	<pre><schema> <logDate mapField="LOG_DT" default="%CurrentDate" required="true" private="true" </schema></pre>
<code>mdField=</code>	"field code"	The meta-data field attribute is used to associate an element with a field's metadata. The field defines data type, as well as its	<pre><schema> <active mdField="CM_ACTIVE_SW" /> </schema></pre>

Mnemonic	Valid Values	Description	Examples
		<p>label and help text. If you link an element with a meta-data field, you don't need to specify any of these attributes.</p> <p>NOTE: For a business object schema, the mapField attribute is used to derive the data type and label. An mdField attribute may be provided to override these attributes, if needed. If the element is mapped to an XML column, the mdField is needed to provide the appropriate data type and label.</p>	
fkRef=	"FK Reference Code"	<p>If the element is a foreign key, defining its FK Reference will enable framework validation of the element during schema interaction and automatically provide descriptions and navigation capability.</p>	<pre><schema> <person fkRef="PER" mapField="CHAR_VAL_FK1"> <row mapChild="CI_SA_CHAR"> <CHAR_TYPE_CD is="PER"/ > </row> </person> </schema></pre>
private=	"true"	<p>Marking an element as private will prevent it from being exposed in schema interaction.</p> <p>NOTE: A private element requires a default.</p>	<pre><schema> <type mdField="SA_TYPE_CD" default="E1" private="true"/ > </schema></pre>
suppress=	"true"	<p>This setting prevents an element from appearing in automatically generated user interfaces.</p> <p>NOTE: This attribute can be specified on a group, in which case all elements of the group will be suppressed.</p> <p>NOTE: Refer to Defaulting and System Variables for advice related to default values for suppressed elements.</p>	<pre><schema> <ls mdField="LIFE_SUPPORT_FLG" default="N" suppress="true"/ < </schema></pre>
	"blank"	<p>This setting means that automatic UI rendering in display mode will hide the element if its value is blank. The element will still</p>	<pre><schema> <email mdField="EMAIL" suppress="blank"/> </schema></pre>

Mnemonic	Valid Values	Description	Examples
		<p>be modifiable on the input map whether blank or not.</p>	
	<p>"input"</p>	<p>This setting means that automatic UI rendering will suppress the element for input, although it may still be displayed if it is not blank.</p> <p>NOTE: Elements marked as suppress="input" will behave as with suppress="blank". If the value is blank, no value will be displayed on either the display or input map. If the element's value is present, then the element will be displayed on both the display and input map.</p> <p>NOTE: Refer to Defaulting and System Variables for advice related to default values for suppressed elements.</p>	<pre data-bbox="1143 275 1430 359"><schema> <email mdField="EMAIL" suppress="input" /> </schema></pre>
<p>noAudit=</p>	<p>"true"</p>	<p>This setting prevents an element from appearing as a changed element in the business object's audit plug-in spot. If specified on a group or list node it applies to the whole node. You cannot specify it on the schema root node, only on schema elements.</p> <p>NOTE: This attribute is only applicable to business object schemas.</p>	<pre data-bbox="1143 940 1536 1171"><schema> ... <version mapField="VERSION_NBR" noAudit="true" /> ... <workFields type="group" noAudit="true" <lastProcessedId/> <lastProcessedTime/> </workFields> </schema></pre>
<p>storeEmptyNodes=</p>	<p>"true"</p>	<p>By default, empty nodes are removed from a business object instance when saved. This setting allows a group or a list element to explicitly keep its empty nodes.</p> <p>This attribute may be useful in situations where business object data is exchanged with an external system and there is a need to distinguish between an empty value for an element that participates in the synchronization and an element that does not participate and therefore omitted from the message altogether.</p>	<pre data-bbox="1143 1350 1620 1455"><schema> <message type="group" storeEmptyNodes= ... </message> </schema></pre>

Mnemonic	Valid Values	Description	Examples
		<p>NOTE: This is only available for group and list elements.</p>	
emitEmptyGroups=	"true"	<p>By default, empty nodes are not included in the output of a business object or a business service call. This attribute is set at the top level schema node, allowing empty nodes of all groups and list elements to be included in the output.</p> <p>This attribute may be useful in the same situations that may require the use of the storeEmptyNodes= attribute.</p> <p>NOTE: This is only available for business object and business service schemas.</p>	<pre><schema emitEmptyGroups="true"> <message type="group"> ... </message> </schema></pre>
emitEmptyElements=	"true"	<p>This attribute is similar to the emitEmptyGroups= attribute but may also be specified at the field level. The attribute is applicable to all schema types.</p>	<pre><schema> <messageNumber type="field" emitEmptyE... ... </message> </schema></pre>
adheresToDA=	"comma separated list of data area names."	<p>This attribute is applicable to the schema, group and list nodes. It claims that this sub-schema node is intended to "look like" the schema of one of the listed data areas, for polymorphism reasons in general.</p> <p>NOTE: The recommended approach for enforcing the structure of a shared data area is by including it in the schema. This attribute should be limited to use cases where the ability to include the data area is not possible.</p>	<pre><schema> <messageNumber type="group" adheresToD... ... </message> </schema></pre>

Schema Constants

There are some product owned schemas where the design warrants a value to be defaulted in the schema, but where the value is implementation specific and therefore cannot be defined by the product. For these situations, the product may use a technique called a schema constant. The design of the schema will include a declared constant. At implementation time, a configuration task will include defining the appropriate value for the constant.

For example, imagine the product delivers an algorithm that will create an outbound message when a certain condition occurs. The outbound message type to use must be configured by the implementation. To use a schema constant to define the outbound message type, the base product will configure the following:

- An option type lookup value for the lookup **F1CN_OPT_TYP_FLG** is defined. For example, **M202 - Activity Completion Outbound Message Type** with a Java Value Name of **outmsgCompletion**
- The base schema that is used to create the "complete activity" outbound message references the schema constant using the Java Value Name of the option type's lookup value

```
...
<outboundMessageType mapField="OUTMSG_TYPE_CD" default="%Constant(outmsgCompletion)"/>
...
```

At implementation time, the administrative users must configure the appropriate outbound message type for "activity completion". Then, navigate to [Feature Configuration](#), choose the **Schema Constants** feature type, choose the option type **Activity Completion Outbound Message Type** and enter the newly created outbound message type in the option value.

Schema constants may also be used in the flattening syntax to define [the row elements required for flattening](#).

Defaulting and System Variables

The default attribute can be used to default values into field elements as well as [the row elements required for flattening](#). You can default a field to a constant or to one of several system variables.

When an element is required, the default value is applied at the server level when adding or changing the record where no value is provided for this element.

When an element is optional, a default value may be configured to provide a suggested value to the user when populating the record on the user interface in Add mode. Users should be able to remove the default value for this type of element and the system will not attempt to populate it at save time.

NOTE:

The standard for our 'list' maintenance is to show an empty row for a new list that a user may or may not choose to populate. As such, UI default values are not shown for list elements.

It is not recommended to configure a default attribute for an optional element that is not editable or visible on the user interface in Add mode (for example an element with **suppress="true"** or **suppress="input"** or a list element whose default value is not shown per the previous point). The default for this type of element is still applied but the user doesn't have the ability to reset the value. This may result in inconsistencies as defaulting does not occur for optional fields added by a use case other than the UI interaction.

Mnemonic	Valid Values	Description	Examples
default=	"value"	Use this attribute to default an element to a specified value. The values that are valid depend on the dataType setting.	<pre><schema> <perType mapField="PER_OR_BUS_FLG" default="P" required="true"/ > </schema> <schema> <frequency dataType="number" default="1" required="true"/ > </schema></pre>
	"%CurrentDate"	Used to default the element to the current date. This is only applicable to date elements.	<pre><schema> <logDate mapField="LOG_DT" default="%CurrentDate" required="true"/ > </schema></pre>
	"%CurrentDateTime"	Used to default the element to the current date / time. This is	<pre><schema> <logDateTime mapField="LOG_DTTM" default="%CurrentDateTime" required="true"/></pre>

Mnemonic	Valid Values	Description	Examples
		only applicable to date / time elements.	</schema>
"%StandardDateTime"		Used to default the standard date and time. The standard date and time is identical to the current date and time, unless daylight savings time / summer time is in effect for the base time zone. This may be used with the stdTime attribute. NOTE: Refer to Standard Time Considerations for more information.	<schema> <startDateTime mapField="START_DTTM" default="%StandardDateTime" required="true"/> </schema>
"%ProcessDate"		You can default the process date. The process date differs from the current date because the process date will remain constant throughout the duration of the process being executed. The current date will reflect the actual date of processing. This is similar to the batch business date that is a standard batch parameter .	<schema> <billDate mapField="BILL_DT" default="%ProcessDate" required="true"/> </schema>
"%ProcessDateTime"		This is similar to " %ProcessDate " but for date / time fields.	<schema> <calcDateTime mapField="CALC_DTTM" default="%ProcessDateTime" required="true"/> </schema>
"%CurrentUser"		Used to default the element to the current user.	<schema> <logUser mapField="LOG_USER" default="%CurrentUser" required="true"/> </schema>
"%CurrentUserTimeZone"		Used to default the element to the current user's time zone. If the current user's time zone is not found, the installation time zone is used.	<schema> <timeZone default="%CurrentUserTimeZone" required="true"/> </schema>
"%CurrentUserLanguage"		Used to default the element to the current user's language.	<schema> <custLanguage mapField="CUST_LANG" default="%CurrentUserLanguage" required="true"/> </schema>
"%InstallationCurrency"		Used to default the currency from the installation record.	<schema> <currency mapField="CURRENCY_CODE" default="%InstallationCurrency" required="true"/> </schema>
"%InstallationCountry"		You can default the country from installation record.	<schema> <country mapField="COUNTRY" default="%InstallationCountry" required="true"/> </schema>

Mnemonic	Valid Values	Description	Examples
			<code></schema></code>
	<code>"%InstallationLanguage"</code>	You can default the language from the installation record.	<pre><schema> <language mapField="LANGUAGE" default="%InstallationLanguage" required="true"/> </schema></pre>
	<code>"%Constant()"</code>	You can default an element value using a schema constant .	<p>The following is an example of a schema constant used as a default value, where the Java Value Name of the Lookup Value is 'customerLanguage'.</p> <pre><language mapField="CUSTOMER_LANG" default="%Constant(customerLanguage)" required="true"/></pre>
	<code>"%Context()"</code>	<p>You can default a value contained in a context variable.</p> <p>WARNING: Context variables must be initialized within a server script before the schema context default can be applied. Refer to Context Variables for more information.</p>	<p>An example of a context variable used as a default value:</p> <pre><source mapField="PER_ID" default="%Context(source)" required="true"/></pre> <p>NOTE: When defining a context variable in scripting, it should be prefixed with \$\$. When referring to the variable in the <code>%Context()</code> syntax, the prefix is not included.</p>
<code>defaultRef=</code>	<code>"XPath"</code>	Use this attribute to default the value of one element to the value of another one.	Refer to Referencing Other Elements for supported syntax for referring to other elements.

The Flattening Nodes and Attributes

The term "flattening" is used to describe the act of defining one or more single elements for a schema that are actually part of a list within the maintenance object. Flattening is possible if there are other attributes of the list that can be defined to uniquely describe the element or elements. A common use case for flattening is a characteristic. Rather than defining the characteristics of an object using a collection where the user must choose the characteristic type and then define the value, the characteristics are defined as actual elements with the appropriate label already displayed. This technique enables the designer of the schema and the user interface to display each separate characteristic in the logical place in the user interface rather than all lumped together.

NOTE: A flattened element represents a unique row in the database. This row is inserted when the flattened values are created. The row is updated when any of the flattened values are changed. The row is deleted when all the flattened values are removed. The behavior of effective dated elements is slightly different - please see [Flattening an Effective Dated List](#).

NOTE: The flattening feature can also be used to define a list, see [Flattened List](#).

Identifying the List or Child Table

When flattening a child table, the row node is required to identify the list / child table that the element comes from. Within the row node, at least one element must be defined with an `is=` definition that ensures that the element is a unique row in

the database. It may also define elements or fields in the row that are suppressed and are populated using default value configuration.

- For a business object, the row node defines the child table the flattened field belongs to.

The syntax is `<row mapChild="table name">`. This example is for the list of persons for an account in the customer care and billing product. One person may be marked as the "main" person. This illustrates how to define an explicit element for the main person ID to simplify references to that field. It is part of the CI_ACCT_PER child table. What makes it unique is that the MAIN_CUST_SW is **true** (and only one row may have that value)

```
<custId mapField="PER_ID" mdField="CM-MainCust">
<row mapChild="CI_ACCT_PER">
  <MAIN_CUST_SW is="true" />
  <ACCT_REL_TYPE_CD default="MAIN" />
</row>
</custId>
```

NOTE: The above example illustrates that the row node may also define elements within the list that are suppressed and assigned a default value. This syntax is never used to identify a particular row. Note that a default value can either be a literal string, or a [system variable](#).

- For a business service, the row node identifies the list name the flattened field belongs to.

The syntax is `<row mapList="list name">`. This example shows two entries from a list being flattened to a field value and description.

```
<selectList type="list" mapList="DE">
  <value mapField="COL_VALUE">
    <row mapList="DE_VAL">
      <SEQNO is="2" />
    </row>
  </value>
  <description mapField="COL_VALUE">
    <row mapList="DE_VAL">
      <SEQNO is="3" />
    </row>
  </description>
</selectList>
```

Uniquely Identifying the Flattened Field or List

The `is=` syntax within a row or rowFilter element is used to uniquely identify the row.

Mnemonic	Valid Values	Description	Examples
<code>is=</code>	"value"	Use this attribute to reference a value directly.	<pre><tdTypeCd mapField="CHAR_VAL_FK1"> <row mapChild="F1_EXT_LOOKUP_VAL_CHAR"> <CHAR_TYPE_CD is="CM-TD-TYPE" /> </row> </tdTypeCd></pre>
	"%Constant()"	You can configure a flattened element using a schema constant . During a service interaction the value of the schema constant will be used to identify the flattened element in its child row.	<p>An example of a schema constant used in flattening syntax where the Java Value Name of the Lookup field value is 'cmRate'.</p> <pre><unitRate mapField="CHAR_VAL" dataType="number"> <row mapChild= "F1_EXT_LOOKUP_VAL_CHAR"> <CHAR_TYPE is ="%Constant(cmRate)" /> </row></pre>

Mnemonic	Valid Values	Description	Examples
	%n	The %n substitution value is used to reference a relative list instance. A relative list instance is typically used to configure a flattened element for a child table keyed by sequence number. The value of n should be a positive integer value. During a business object read interaction the relative list instance (position) specified by the integer will be returned.	<pre></unitRate></pre> <p>An example with a relative list instance - where the first instance of the row is returned.</p> <pre><schema> <mainPhone mapField="PHONE"> <row mapChild="CI_PER_PHONE"> <PHONE_TYPE_CD is="%Constant(mainPhoneType)"/> > <SEQ_NUM is="%1"/> </row> </mainPhone> </schema></pre>

FASTPATH: Additional values for `is=` are used when [Flattening an Effective Dated List](#). Refer to that section for more information.

Flattening a Pre-defined Characteristic Type

If the flattened field is in a characteristic collection and the characteristic type is a predefined characteristic, automatic UI rendering will generate a dropdown for the list of valid values. For example, the schema below will generate a dropdown for the Usage element showing the valid values of the Status Reason Usage (**F1-SRUSG**) characteristic type.

```
<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>
```

Defining Multiple Elements from the List

When attempting to include multiple columns from the same list, the system provide shorthand notation for copying the flattening rules defined on another element so that the flattening rules do not need to be repeated. To do this, the row node includes the **rowRef** attribute and it indicates the other element name that defines the mapping information. The following example illustrates flattening the fields Customer ID and Receives Copy of Bill from the same list of Persons for an Account (where the `MAIN_CUST_SW` is `true`).

```
<custId mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true" />
    <ACCT_REL_TYPE_CD default="MAIN" />
  </row>
</custId>
<copyBill mapField="RECEIVE_COPY_SW" rowRef="custId"/>
```

Note that the above notation also illustrates that the **rowRef** may be defined directly in the element's attribute definition.

NOTE: Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements.

Flattening Two Layers Deep

If your maintenance object or service has nested lists two layers deep, the system supports flattening an element within a flattened element. This technique also uses the **rowRef** attribute. The flattening of the second level refers to the flattened element of the first level. The following example illustrates flattening a characteristic into an element called `legalContact` for the "main" customer. Notice that the `legalContact` element has two row nodes: one to refer to the flattening information for its parent record and one to define its child table

```
<custId mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
```

```

    <MAIN_CUST_SW is="true" />
    <ACCT_REL_TYPE_CD default="MAIN" />
  </row>
</custId>
<legalContact mapField="CHAR_VAL_FK1">
<row rowRef="custId">
  <row mapChild="CI_ACCT_PER_CHAR" >
    <CHAR_TYPE_CD is="LEGAL" />
  </row>
</row>
</legalContact>

```

Note that the above notation also illustrates that the **rowRef** may be defined as an attribute of a row node rather than directly in the element's attribute definition.

Defining a Flattened List

There are times that a list or child table supports multiple values of the same "type" and these should be presented as a list. To continue with the example above, the list of persons for an account may identify one person as the "main" person. This person has been flattened to a single element (with the account relationship type defaulted and suppressed). To maintain the other persons related to an account, you can define a list where each row captures the Person Id and the Account Relationship Type.

Rather than a row node, the flattened list is configured with a **rowFilter** element. The following schema illustrates the described example. The list node defines the child table. The **rowFilter** includes the criteria that identify the rows within the table to include. The elements of the list are defined within the list node outside the **rowFilter** element.

```

<custId mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true" />
    <ACCT_REL_TYPE_CD default="MAIN" />
  </row>
</custId>
<miscPersons type="list" mapChild="CI_ACCT_PER">
  <rowFilter>
    <MAIN_CUST_SW is="false" />
  </rowFilter>
  <relType mapField="ACCT_REL_TYPE_CD" />
  <personId mapField="PER_ID" />
</custId>

```

Note that the system will validate that if a schema contains flattened single elements and flattened lists from the same child table, the criteria that defines what makes them unique must be analogous.

Flattening an Effective Dated List

There are some lists in the application that are effective dated (and still others that have effective date and time). For example, there are some effective dated characteristic collections. In these collection, the design is to capture a single value for a characteristic type that may change over time. It is not meant to support multiple characteristic values in effect at the same time. The following highlights some information regarding effective dated characteristic functionality:

- The most recent dated row is returned when invoking a BO for read.
- No new row added when all of the values are unchanged on a change to the BO.
- The flattened row value is updated when any of the flattened values are changed and the most recent date is equal to the current date (or the referenced effective date);
- A new row value is inserted when any of the flattened values are changed and the most recent date is different than the current date (or the referenced effective date);

NOTE: Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements.

When flattening an effective dated list, the date column must include information regarding the date to use. The following table highlights the possible values.

Mnemonic	Valid Values	Description	Examples
is=	"%effectiveDate"	<p>Use this configuration to indicate that current date should be used for processing. Any value added or updated using this schema will be for the current date.</p> <p>With this option, if the maintenance object allows for the characteristic value to be blank, then setting the flattened value to blank during the BO update will result in updating the existing record with an empty value, or adding a new row with an empty value in case the current date effective dated record is not found.</p>	<pre><schema> <price mapField="CHAR_VAL" dataType="number"> <row mapChild="CI_SA_CHAR"> <CHAR_TYPE is="PRICE" /> > <EFFDT is="%effectiveDate" /> </row> </price> </schema></pre>
	"%effectiveDate()"	<p>Use this configuration to indicate that the date to use the value of another element.</p> <p>NOTE: Refer to Referencing Other Elements for supported syntax for referring to other elements.</p>	<pre><schema> <price mapField="CHAR_VAL" dataType="number" required="true"> <row mapChild="CI_SA_CHAR"> <CHAR_TYPE is="PRICE" /> > <EFFDT is="%effectiveDate(priceEdate)" /> </row> </price> <priceEdate mapField="EFFDT" rowRef="price" /> </schema></pre>
	"%effectiveDateTime"	<p>Use this configuration to indicate that current date /time should be used for processing. Any value added or updated using this schema will be for the current date / time.</p>	<pre><schema> <price mapField="CHAR_VAL" dataType="number"> <row mapChild="RATE_CHAR"> <CHAR_TYPE is="PRICE" /> > <EFFDT is="%effectiveDateTime" /> </row> </price> </schema></pre>
	"%effectiveDateTime()"	<p>Use this configuration to indicate that the date / time to use the value of another element.</p> <p>NOTE: Refer to Referencing Other Elements for supported syntax for referring to other elements.</p>	<pre><schema> <price mapField="CHAR_VAL" dataType="number"> <row mapChild="RATE_CHAR"> <CHAR_TYPE is="PRICE" /> > <EFFDTTM is="%effectiveDateTime(priceEdatetime)" /> </row> </price> <priceEdatetime mapField="EFFDTTM" rowRef="price" /> </schema></pre>

Search Zone

A UI Map schema element can be configured to enable an automatic search dialog when the schema is included within a maintenance UI map.

NOTE: Please note that an fkRef can be configured with a search zone. If a schema element has an fkRef but no explicit search attributes (as described here) then the fkRef search information will be used in the UI map. In other words, if the schema element already has an fkRef, then these explicit search attributes in the schema are only used to override the fkRef search information.

NOTE: Refer to the [UI Map Attributes and Functions](#) for more information on search zone configuration.

search="search zone"

The search attribute can be used within a UI map schema and is used to automatically generate the oraSearch UI map attribute. The search zone is an explorer zone configured as a search.

```
<person fkRef="PER" search="C1_PSRCH" />
```

searchField="search field:element['literal'];"

The searchField attribute can only be used in conjunction with the search attribute. The searchField attribute is used to build the oraSearchField UI map attribute. The searchField value is used to populate a search zone filter with an initial value when the search zone is launched. The initial value can be a literal also. The searchField value is used to match to the filter mnemonic also named searchField.

Search field: element['literal']. The search field represents the search zone filter to populate on launch. The element is the map's schema element used to populate the filter. The element is optional, if left blank, it will default to the element that this attribute is specified on. The searchField also takes 'literal' as input value

NOTE: Multiple filters can be populated within the search zone at launch, but multiple search field pairs must be constructed within the attribute value. The value specified here will be used to directly build the HTML attribute oraSearchField within the UI map where this schema is specified.

NOTE: Note that the element reference is *relative*. Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements.

```
<person fkRef="PER" search="C1_PSRCH" searchField="PERSON; PER_TYPE:personType;" />
```

searchOut="search field:element;"

The searchOut attribute can only be used in conjunction with the search attribute. The searchOut attribute is used to build the oraSearchOut UI map attribute. The searchOut value is used to capture a selected value from the search zone and move it to a UI map element. The searchOut value specified should match the ELEMENT_NAME mnemonic within the search result zone parameter.

Search field: element. The search field represents the search zone result brought back to the UI map. The element is the map's schema element to be populated. The element is optional, if left blank, it will default to the element that this attribute is specified on.

NOTE: Multiple elements can be populated as a result of search zone selection, but multiple search field pairs must be constructed within the attribute value. The value specified here will be used to directly build the HTML attribute oraSearchOut within the UI map where this schema is specified.

NOTE: Note that the element reference is *relative*. Refer to [Referencing Other Elements](#) for supported syntax for referring to other elements.

```
<person fkRef="PER" search="C1_PSRCH" searchField="PER_ID" searchOut="PER_ID; PRIMARY_PHONE:personPhone;" />
```


Extend Security for Service Script

Application service security will be enforced when either a business object or a service script is invoked from a BPA script or a UI map, but not from a service script. If you want security to be enforced when the business object or a service script is invoked from a service script, you must add the following attribute to the service script's schema.

appSecurity="true"

The appSecurity attribute is only available for service script schemas. If specified, any business object or service script directly invoked by the service script will have their application service evaluated for access.

```
<schema appSecurity="true">
  ...
</schema>
```

Overriding Action for a Business Service

If you want to invoke a business service with an action other than 'read', you need to specify the action attribute on the primary node business service schema.

pageAction="add, change, delete"

The action attribute is used to override the default action of read on a business service schema. Valid values are:

- add
- update (only allowed for maintenance object service)
- change (not allowed for maintenance object service)
- delete
- read (this is the default action if no pageAction specified)

Example:

```
<schema pageAction="change">
  <parm type="group">
    <ele1/>
    <ele2/>
  </parm>
</schema>
```

Specifying searchBy for a Search Service

If you want to invoke a search service then you must explicitly specify the searchBy attribute appropriate for the elements mapped in the schema.

searchBy="MAIN"

The value values of the searchBy attribute can be found by viewing the XML schema linked to the business service, use the View XML URL. Typical values are:

- MAIN
- ALT
- ALT2
- ALT3
- etc.

```
<schema searchBy="MAIN">
  <AccountID mapField="ACCT_ID" />
  <Results type="list">
    <AccountID mapField="ACCT_ID" />
  </Results>
```

```
</schema>
```

Including Other Schemas

There are no limitations on your ability to include a schema into another schema - all types can be included in all other types. Nested includes are also allowed - and at present there is no limitation on the depth of the nesting.

Including a schema requires two parts:

1. The include node
2. The name attribute

The following table highlights the supported include statements.

Mnemonic	Description	Examples
<code><includeBO name=" "/></code>	Including a business object schema into another schema is allowed. Note that the mapping rules of a business object or business service schema may or may not make sense in the context of the parent schema. Include other schemas at your own risk. However, a very useful aspect of XML processing is that the framework ignores non-pertinent attributes. In other words, it will not hurt to have mapping attributes included into a script schema.	<pre><schema> <cust type="group"> <includeBO name="Cl-Person" /> </cust> <spouse type="group"> <includeBO name="Cl-Person" /> </spouse> </schema></pre>
<code><includeBS name=" "/></code>	Used to include a business service schema.	<pre><schema> <includeBS name="F1-ReadMOLog" /> </schema></pre>
<code><includeDA name=" "/></code>	Used to include a data area schema.	<pre><schema> <includeDA name="F1CommonSchemaFieldData" / > </schema></pre>
<code><includeMP name=" "/></code>	Used to include a UI map schema.	<pre><schema> <includeMP name="F1-DisplayRecordActions" / > </schema></pre>
<code><includeSS name=" "/></code>	Used to include a service script schema.	<pre><schema> <includeSS name="F1-ActShowZn" /> </schema></pre>

Compatibility Attributes

These attributes were added as part of upgrades from previous versions of the Framework.

fwRel="2"

This attribute has been added to schemas created in Framework 2 as part of a Framework 4 upgrade. New schemas will not need this attribute. It is not advisable to modify this attribute without understanding the following behavior differences as improper changes could result in errors:

NOTE: Schemas created in Framework 2 with the fwRel="2" attribute will store any XML mapped fields under groups as top-level XML elements in the mapXML field. This means that if two or more fields, in different group structures,

were to have the same field name, their storage would conflict with one another resulting in errors. The new behavior, without the fwRel="2" attribute, will preserve the group structure and avoid the conflicts.

```
<schema fwRel="2"
  . . .
</schema>
```

UI Hint Syntax

Contents

- [Working Examples](#)
- [Technical Notes](#)
- [Format an Input Map Title](#)
- [Create a Section](#)
- [Include a Map Fragment](#)
- [Build Dropdown](#)
- [Conditionally Hide Elements](#)
- [Conditionally Protect Elements](#)
- [Trigger Dependent Behavior](#)
- [Control Rendering Target](#)
- [Generate a Text Area](#)
- [Modify FK Reference Defaults](#)
- [Suppress Automatic Number Formatting](#)
- [Auto Capitalize the Input Data](#)

Working Examples

For working examples of uiHint functionality, refer to the following business objects:

BOs with User Assigned Keys

The following examples illustrate the patterns used to enable uiHints on an object with a user specified key.

- **F1-OutcomeStyleLookup.** This extendable lookup BO does not require state transition, but does allow duplicate and delete actions.
- **F1-TodoSumEmailTyp.** This request type illustrates the hints required to support state transition on a display map.
- **F1-WebSvc.** This web service BO is a good example for management of complex JavaScript requirements. Both display and input maps have functionality that requires specialized javascript.

BO with System Generated Key

The following example illustrates the pattern used to enable uiHints on an object with a system generated key.

- **F1-GenericAttachment.** This attachment BO has a system assigned key, which entails the following special handling:
 - **F1-AttachmentMain.** This is the main section data area contains the elements common to all attachments, including the key, bo, and version. Because this data area is used to define the main section of the generated maps, the main section of the map can be extended by an implementation via data area extension functionality.
 - **F1-AttachmentActions.** This record actions map contains the standard actions, Edit and Delete, plus custom actions used only by attachments, View and Upload.
 - **F1-AttachmentIDFrag.** This record information map contains the primary key of the attachment.

Display Map Service Script

Display map service scripts can be fully supported via dynamic HTML generation. However, to help eliminate the need for a display service script, self-contained uiHint functionality has been developed to write the business object status and determine valid state transitions. So the two most common reasons to craft a display service script have been eliminated.

A typical reason to use a display pre-script is if you have an embedded map fragment that contains a business service schema. The display service script can be used to invoke the business service. Both the map fragment and the display service script must declare the business service schema to support this scenario.

WARNING: The zone used to display the object's map must have a derivation script, like **F1-GncDsMpDZ** or **F1-GenDss**, that will invoke a display service script for the business object if it has been defined as a BO option - but not require an explicit display map BO option. In addition, the display service script's schema must be enabled for uiHint functionality - as the script's schema will be dynamically rendered by the zone - and not the BO schema.

- **F1-ExcelSpreadsheet.** This attachment BO has a display service script used to manipulate the attachment business object before displaying it:
- **F1-AttchDtU.** This display map service script's schema has been defined with the uiHint namespace, and will have a display map generated for it.

Maintenance Pre-Processing Service Script

Maintenance pre-processing service scripts can be used with uiHints.

- **F1-ExcelSpreadsheet.** This attachment BO has a maintenance pre-processing service script used to manipulate the attachment business object before rendering the maintenance map:
- **F1-AttchPre.** This pre-processing service script's schema mimics a maintenance map schema with embedded boGroup and action elements. It will be invoked before the maintenance map is rendered.

Maintenance Post-Processing Service Script

Maintenance post-processing service scripts can be used with uiHints.

- **F1-ExcelSpreadsheet.** This attachment BO has a maintenance post-processing service script used to manipulate the attachment business object after rendering the maintenance map:
- **F1-AttchPost.** This post-processing service script's schema mimics a maintenance map schema with embedded boGroup and action elements. It will be invoked after the maintenance map is rendered.

Technical Notes

The following prerequisites are required to support dynamic HTML generation:

Schema Requirements

To support automated UI generation, the business object schema must contain the following:

- `<schema xmlns:uiHint="http://oracle.com/ouafUIHints">`. The schema node must name the uiHint namespace.
- `isPrimeKey="true"`. Every element of the business object schema that is part of the primary key must be identified.

Maintenance Script Requirements

The maintenance script for the MO must be enabled for dynamic generation.

CAUTION: The business object maintenance BPA script must be declared as an MO Option for uiHint maintenance functionality to work!

If the script performs **F1-BOProc** then it is likely no special functionality is needed. However, if the maintenance script contains its own call to **F1-GetValOpt** then the following statement is required prior to that call:

```
move 'false' to "F1-GetBOOpts/input/maintenanceMapRequired";
performScript 'F1-GetValOpt';
```

After the call to **F1-GetValOpt** the following logic must be included to dynamically declare the map schema if the business object does not have a maintenance map of its own:

```
// Perform Main Processing
if ("F1-GetBOOpts/output/maintenanceMap = $BLANK")
  declareBOWithBOGroup "$bo" as 'map_schema';
else
  declareMap "F1-GetBOOpts/output/maintenanceMap" as 'map_schema';
end-if;
```

Format an Input Map Title

NOTE: Throughout this topic the term "field" to refer to both the generic concept of displaying and capturing data in a 'field' as well as referring to the meta-data object supplied in the product to define [Fields](#). When referring to the latter, the term "MD Field" (meta-data Field) is used.

A uiHint element can be used to build a title for a maintenance map. The title will only print on the maintenance map, not on the display map. It will be printed as the first line in the map, centered, with a heading style.

Syntax	Description	Examples
<code><uiHint:title mdField=" " /></code>	Displays the label of a referenced MD field as the title.	<pre><schema xmlns:uiHint="http:// oracle.com/ouafUIHints"> <uiHint:title mdField="STATUS_RSN_LBL" /> ... </schema></pre>
<code><uiHint:title text=" " /></code>	Displays the indicated text as the title. (Do not use this mechanism when multiple languages are supported.)	<pre><schema xmlns:uiHint="http:// oracle.com/ouafUIHints"> <uiHint:title text="Status Reason" /> ... </schema></pre>

Create a Section

The uiHint namespace supports the definition of a UI map section. Note that sections are currently created in generated UI Maps when the schema has a group or list node with a label or mdField. The functionality described here enables the creation of a section without requiring a labeled group or list node within the schema. Every section must be bounded by **startSection** and **endSection** element pair.

Syntax	Supporting Attributes	Description
<code><uiHint:startSection .../></code>	<code>sectionColumn="left right fullWidth float"</code>	The default is that the section will be the full width in display maps. To override that setting, specify if you want a half-width section to appear in either the left (left) or right (right) column or to float (float). Sections that are marked as 'float' will display half-width and be aligned according to whether prior sections are displayed or conditionally hidden. For example, if a left-aligned section is followed by a floating section, the floating section will appear in the right column if the left section is

Syntax	Supporting Attributes	Description
		populated but will display in the left column if the left section is hidden / collapsed.
	<code>editColumn="left right fullWidth float"</code>	By default a section appears as full width in maintenance maps. To override that setting, specify if you want a half-width section to appear in either the left (left) or right (right) column or to float (float). The behavior is the analogous to the sectionColumn behavior.
	<code>sectionOpen="false"</code>	By default a section is open on initial display. Specify this attribute to initially display the section as closed (collapsed).
	<code>mdField=" "</code>	Specify the name of a MD field whose label should be used as the section heading.
	<code>label=" "</code>	Specify the explicit text to use as the section heading.
	<code>visibleOn="displayMap inputMap"</code>	By default a section appears on both the display and the input maps. Use this attribute to limit the display of the section to either the display map (displayMap) or input map (inputMap).

The syntax for the end section attribute is `<uiHint:endSection/>`

Examples:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <uiHint:startSection label="Main" sectionColumn="left"/>
  ...
  <uiHint:endSection/>
</schema>
```

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <uiHint:startSection mdField="F1-ADD-
INFO" sectionColumn="fullWidth" editColumn="float" sectionOpen="false" visibleOn="displayMap"/>
  ...
  <uiHint:endSection/>
</schema>
```

NOTE: The `sectionColumn`, `editColumn` and `sectionOpen` attributes are available for group and list nodes as well.

Include a Map Fragment

You can specify a UI map fragment to inject HTML into a generated map using the **includeMap** element name. This allows for you to support more sophisticated behavior on your user interface. For any element that is included for rendering in the map fragment, be sure to suppress the element in its schema definition, otherwise HTML will automatically be generated for the element.

Syntax	Supporting Attributes	Description
<code><uiHint:includeMap .../></code>	<code>map=" "</code>	Specify the name of the map.
	<code>visibleOn="displayMap inputMap"</code>	By default the details from the map fragment appear on both the display and the input maps. Use this attribute to limit the display of the section.

Example:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <uiHint:includeMap map="StandardActionButtons" visibleOn="displayMap"/>
  ...
</schema>
```

NOTE: Important note on the map fragment schema: If a map fragment contains a schema, then the fragment schema structure will be injected into the dynamically generated schema when the business object is rendered for input. Technically, the fragment schema will be inserted after the boGroup structure within the map's schema. This method may be used to support the implementation of maintenance pre and post script processing for a business object and oraInvokeBS function calls within embedded JavaScript.

If JavaScript is required within an XHTML UI Map fragment, it is necessary to bound it within a **![CDATA[]]** tag to ensure a valid XML document. Note that the tags themselves may need to be commented out to promote compatibility with older browsers. For example:

```
<script type="text/javascript">
/*  */
//
//javascript
//
/* ]]&gt; */
&lt;/script&gt;</pre></div><div data-bbox="89 395 896 457" data-label="Text"><p><b>Flush the cache:</b> For performance reasons, the Framework automatically caches business object schemas, data areas, and UI maps. When you update a business object, the cache is automatically flushed. However, if the business object includes either a data area or embedded UI map fragment, the cache must be manually flushed in order for your changes to be recognized. Refer to <a href="#">Server Cache</a> for more information.</p></div><div data-bbox="89 475 257 492" data-label="Section-Header"><h2>Build A Dropdown</h2></div><div data-bbox="89 496 896 526" data-label="Text"><p>Syntax is provided to build a dropdown list in an edit map. The dropdown may be built using data returned from a service script, a business service or a table.</p></div><div data-bbox="89 532 906 621" data-label="Table"><table border="1"><thead><tr><th>Syntax</th><th>Description</th></tr></thead><tbody><tr><td>uiHint:select="ss: "</td><td>Specify the name of the service script after the colon.</td></tr><tr><td>uiHint:select="bs: "</td><td>Specify the name of the business service after the colon.</td></tr><tr><td>uiHint:select="table: "</td><td>Specify the name of the table after the colon.</td></tr></tbody></table></div><div data-bbox="89 636 885 666" data-label="Text"><p>When specifying a service script or a business service, extra mapping information is needed to pass data to and from the service.</p></div><div data-bbox="89 672 906 836" data-label="Table"><table border="1"><thead><tr><th>Syntax</th><th>Values</th><th>Description</th></tr></thead><tbody><tr><td rowspan="2">uiHint:selectIn=" "</td><td>serviceXPath:element</td><td>Used to pass the value of another element into the service (mapping to the service's XPath).</td></tr><tr><td>serviceXPath:'Literal'</td><td>Used to pass a constant or literal to the service (mapping to the service's XPath).</td></tr><tr><td>uiHint:selectOut="valuePath: ; descPath: "</td><td>See examples below.</td><td>Used to indicate which element in the service's output holds the values and which one holds the descriptions.</td></tr></tbody></table></div><div data-bbox="89 852 163 868" data-label="Text"><p>Examples:</p></div><div data-bbox="89 873 734 897" data-label="Text"><pre>&lt;schema xmlns:uiHint="http://oracle.com/ouafUIHints"&gt;
  &lt;boStatus mapField="BO_STATUS_CD" uiHint:select="bs:F1-BOStateReasonList"</pre></div><div data-bbox="147 942 913 960" data-label="Page-Footer"><p>Oracle Utilities Work and Asset Management and Operational Device Management Administrative User Guide • 348</p></div>
```

```

    uiHint:selectIn="boStatusBO:boStatusBO" uiHint:selectOut="valuePath:results/status;
    descPath:results/description"/>
    ...
    <algorithm mdField="ALG_CD" uiHint:select="bs:F1-RetrieveSysEvtAlgorithms"
    uiHint:selectIn="algorithmEntity:'F1AA';" uiHint:selectOut="valuePath:results/algorithm;
    descPath:results/description"/>
    ...
    <outboundMsgType mdField="OUTMSG_TYPE_CD" required="true" fkRef="F1-
    OMTYP" uiHint:select="table:F1_OUTMSG_TYPE"/>
</schema>

```

Conditionally Hide Elements

The **displayNone** attribute is used to suppress elements on the map based on conditions.

Syntax	Values	Description
uiHint:displayNone=	"XPath','value','!' '='"	Used to conditionally hide this element based on the value of another element (referenced using its XPath). Enter a value of '' to interrogate a blank value. By default the operator is '='. This may be overridden using '!='. <hr/>
	"function name, true false"	Used to indicate a JavaScript function, which must return a Boolean.

WARNING:

Embedded spaces are not supported within the comma separated string values of this attribute.

This setting may be used on group nodes, list nodes, and elements - except for elements within a list. Elements within a list cannot be hidden conditionally.

The following example illustrates that two elements (currency reference and lookup) that will be hidden or displayed based on the value of the data type element. Note that this example also illustrates [Trigger Dependent Behavior](#) because the data type element's value may change and if it does, the condition for hiding the subsequent elements should be re-evaluated.

```

<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
    ...
    <dataType mdField="F1_SE_DATA_TYPE" dataType="lookup" lookup="F1_SE_DATA_TYPE"
    uiHint:dependents="currencyRef;lookup; "/>
    <currencyRef mdField="F1_SE_CURR_REF_LBL" uiHint:displayNone="'dataType','F1MO','!='/>
    <lookup mdField="F1_SE_LOOKUP_LBL" fkRef="F1-LKUPF" uiHint:displayNone="'dataType','F1LP','!
    ='"/>
    ...
</schema>

```

The following example illustrates referring to a function where the function receives parameters:

```

<uiHint:startSection mdField="F1_SE_DEFAULT_SECT"
    uiHint:displayNone="isApplicableForSchemaType(item,'F1MP'),true"/>

```

Conditionally Protect Elements

The protect attribute is used to protect elements on the map based on other factors.

Syntax	Values	Description
uiHint:protect=	"XPath','value','!' '='"	Used to conditionally protect this element based on the value of another element (referenced using its XPath). Enter a value of '' to interrogate a blank value. By default the

Syntax	Values	Description
		operator is '='. This may be overridden using '!='. <hr/>
	"function name, true false "	Used to indicate a JavaScript function, which must return a Boolean. <hr/>
	" action ','A' 'C','!=' '='"	Use the ' action ' setting to protect the element based on the current action. For example, certain elements may only be specified when adding a record. Any subsequent changes to the record should protect the element from being changed. When using this option, the valid values for the ' <i>value</i> ' are A (add) and C (change). <hr/> <hr/>

WARNING:

Embedded spaces are not supported within the comma separated string values of this attribute.

The protect UI Hint may be used on group nodes, list nodes, and elements - except for elements within a list. Elements within a list cannot be protected conditionally.

The following UI Hint will protect the statistics category when the action is 'C'.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
...
<statisticsCategory dataType="lookup" mapField="STAT_CATEGORY_FLG"
  lookup="STAT_CATEGORY_FLG" uiHint:protect=" 'action','C','!=' />
...
</schema>
```

Trigger Dependent Behavior

The dependents attribute is used to trigger behavior on a child element when a parent element is changed.

Syntax	Values
uiHint:dependents=" "	A list of one or more dependent elements separated by semicolons. <hr/> <hr/>

The following example illustrates that the dropdown list of one element is driven by the value of another element. In this example, when the Country changes, the list of States to choose from should change to only show the states for the indicated country.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
<country label="Country" uiHint:select="table:CI_COUNTRY" uiHint:dependents="state" />
<state label="State" uiHint:select="ss:CM-RetrieveCountryStates"
  uiHint:selectIn="input/country:country;" uiHint:selectOut="valuePath:output/state/stateCode;"
  descPath="output/state/stateDesc" />
...
</schema>
```

NOTE:

Dependent targets may only name elements, not group or list nodes.

Do not modify the "id" attribute value of dependent and parent element. Data population in dependent is done based on the "id" attribute value.

Control Rendering Target

By default all elements that are not suppressed are visible on both the display map and the input map. Use the **visibleOn** attribute to limit the inclusion of an element to either the display or input map.

Syntax	Values
uiHint:visibleOn=	"displayMap"
	"inputMap"

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <uiHint:includeMap map="StandardActionButtons" visibleOn="displayMap"
  ...
</schema>
```

Generate a Text Area

By default, a standard text box is rendered in an input map for any string element. If the field is larger and you wish to have a bigger text area (with a scroll bar), use the **textArea** attribute.

Syntax

uiHint:textArea="true"

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <message label="Message" uiHint:textArea="true"/>
  ...
</schema>
```

Modify FK Reference Defaults

By default, when an element with **fkRef** is displayed, an info string, context menu, navigation, and search are enabled (if the FK reference has been configured accordingly). Syntax is provided to allow you to selectively turn off any of these features.

Syntax

uiHint:fkRef="info:false;context:false;navigation:false;search:false;"

Only the feature that you wish to turn off needs to be specified. The following example illustrates turning off the navigation capability, meaning the text will not be rendered as hypertext.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <attachmentID fkRef="F1-
ATTCH" primeKey="true" suppress="input" uiHint:fkRef="navigation:false;"/>
  ...
</schema>
```

FASTPATH: Refer to [FK Reference Formatting](#) in the UI Map Attributes section for more information on each FK reference setting.

Suppress Automatic Number Formatting

By default numeric fields (**dataType="number"**) are formatted as numbers. An attribute is provided to instead apply alphanumeric formatting.

Note: If **dataType** is not specified explicitly, it is derived from **mdField** or **mapField**.

Syntax

uiHint:alphaFormat="true|false"

By default, its value is **false** (and therefore can be left out altogether).

Examples:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  ...
  <numberCount mdField="" dataType="number" uiHint:alphaFormat="true"/>
  ...
</schema>
```

Auto Capitalize the Input Data

The uiHint provides syntax to automatically capitalize input data.

Syntax

uiHint:capitalize="true|false"

By default, its value is **false** (and therefore can be left out altogether).

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <toDoTypeCd mdField="TD_TYPE_CD" uiHint:capitalize='true' isPrimeKey="true"/>
</schema>
```

NOTE: This attribute is ignored if **uiHint:textArea="true"** is configured.

The attribute is only available in the schema designer when the **isPrimeKey** is set to **true**. The attribute may be added to any string element when using the source viewer.

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 two view modes:

- **Text** mode shows the schema elements and their attributes written in the proper syntax and allows for direct text entry.
- **Tree** mode is a view showing the elements in a tree format with many of the key attributes of each element in tabular form. While in this mode, the **Node Display** controls allow the user to choose between displaying the elements by their internal identifiers or their associated screen labels while viewing or editing the schema.

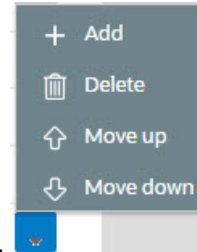
NOTE: Schemas that are not owned by the current installation owner are protected in both view modes.

The following sections provide more information about functionality available in the Tree mode.

- If the schema definition refers to another schema using an 'include' statement, a triangle is visible to the left of the tree area. One can expand that schema by clicking the triangle.
- Detailed information about an element's definition can be displayed and if applicable, updated from the **Tree** view by clicking the edit icon for that element, which appears on the right.  Note that the element attributes are only editable

if the element is defined in this schema. When viewing the attributes of an element that is part of an 'include' of another schema, the attributes are display only.

- Context-sensitive embedded help is provided for fields and controls in the edit pane by clicking the Help icon ⓘ
- There is a menu dropdown icon visible to the left of the tree for any element that is defined in the current schema being viewed. It is not visible for elements included from another schema. New elements may be added in the **Tree** mode.



Clicking that dropdown displays options to Add, Delete or Move an element. Note that not all options are visible, based on what is allowed for the element adjacent to the menu. For example, on the main "schema" node, only the Add option is visible as deleting or moving the "schema" node is not applicable.

When adding a new element, you are prompted first for the element position, which is either a sibling node to the current element or a child node of that element. For either option, the new element is added below the one adjacent to where the menu is clicked. You are then prompted for the element type.

The following lists the possible element types. Most are self explanatory and represent standard schema options.

- Characteristic List. This is a special type of Flattened List element that is used to map list elements that include a sequence and a characteristic value for a given characteristic type. This element is only applicable for maintenance objects that have one or more characteristic child tables where the primary key is the maintenance object's key, characteristic type and sequence. Effective dated characteristic collections are not supported. The user defines the list name, the characteristic value element name and the characteristic type. The system will configure the remaining flattening information accordingly.
- Characteristic. This is a special type of Flattened Field element that is used to map a single element to the characteristic for a given characteristic type. This element is only applicable for maintenance objects that have one or more characteristic child tables where the primary key is the maintenance object's key, characteristic type and sequence. Effective dated characteristic collections are not supported. The user defines the element name and the characteristic type. The system will configure the remaining flattening information accordingly.
- 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 a column that supports data defined in an XML structure. (This is either a column with the character large object data type (CLOB) or the XML data type).

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.

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 the [Admin menu](#) are suppressed, the menu itself is suppressed in the application toolbar. This is not expected to ever occur for the Main menu as the user preferences is on that menu and the expectation is that all users have access to their preferences.
- Menu items that are suppressed are not visible in the [Menu Item Search](#) results.

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.

Accessibility Considerations

This section provides information about topics related to accessibility that should be considered when extending the system via configuration tools.

Color Contrast

The HTML color **red** (RGB code #FF0000) when used for text on a white background does not comply with the accessibility color contrast guidelines. When it is desired to use red for text for emphasis in a zone, map or in a text string, the product recommends the shade of red with the RGB code #E0292F.

Note that in addition, the product provides a special CSS class **textColorRedOnWhite** which may be used when HTML references are used for applying the red color. For example, UI map, zone help text and scripting.

Referencing URIs

There are some configuration objects that require a reference to a URI, including file path URIs. The following sections highlight some functionality supported by the product with respect to defining / accessing URIs.

NOTE: In order for the functionality described below to occur, specific APIs must be used by the underlying code related to the fields that capture or process the file path or URL information. If you find that there is a URI-related field that does not provide the functionality described here, please contact customer support.

NOTE: For schema elements in a business object that reference a [Field](#) with the **URI** data type or define the element with the **URI** data type configured in the [schema](#) will automatically use the appropriate API that validates the value that may reference a substitution variable or must be checked against the allowlist if applicable.

Validation Against an Allowlist

Based on a property setting, your implementation may be configured to define a allowlist of URIs. If this setting is enabled, the system will issue an error if the URI is not defined in the allowlist. Consult your system administrator to verify if this setting is enabled or not for your implementation.

FASTPATH: Refer to [URI Substitution](#) for information about how defining substitution tokens for URIs using the properties file technique automatically adds the defined URI to the allowlist.

URI Substitution

The system supports the ability to define substitution variables (sometimes referred to as tokens) for both URL values and file path values that reference native file storage locations. For URLs, the system supports defining variables in a Substitution Variable properties file. For native file storage paths, there are two options: the Substitution Variable properties file or via a File Storage configuration extendable lookup. More details on both these options are found below.

- Substitution variables properties file. A substitution variable for all or part of the URI definition may be configured in a properties file. This allows the system administrators to define the proper URI locations in a properties file whereas the configuration users only need to know the variable name. For example, when defining a location for an extract file in an extract batch job, instead of typing a file path of `h:\oracle\serverName\1.0.0.0\batch\extract\`, the batch user can enter `@FILE_EXTRACT@`, assuming there is an entry in the substitution variables file with a name of `FILE_EXTRACT`, and a value of `h:\oracle\serverName\1.0.0.0\batch\extract\`. Another example is that the batch user could enter `@BATCH_FILES@\extract\`, assuming that the URI variable for `BATCH_FILES` is defined as `h:\oracle\serverName\1.0.0.0\batch\`.

NOTE: The product automatically populates the value of `SPLOUTPUT` in the properties file so that this may be used in URI configuration. In addition, the product may supply some pre-defined variable names for other common references. As part of this, the 'advanced' menu in the system installation steps may prompt for installers to define the values of these pre-defined variables, if desired. Installations may opt to define additional substitution variables for various URI references. Refer to the *System Administration Guide* for more information.

- File Storage extendable lookup. Specifically for file paths that reference the native file system, the system also supports the ability to define a path using the **File Storage Configuration** extendable lookup. The following points highlight the steps to take for this option.
 - Navigate to [Extendable Lookup](#) and search for the **File Storage Configuration** lookup.
 - Click **Add** to create a new entry. Define a lookup value name. This will be used when configuring a file path that uses this value. Choose the File Adapter value of **Native File Storage**. Enter the desired file path value. The values defined here may in turn refer to values defined in the Substitution Variables properties file. For example, the file path can reference `@SPLOUTPUT@`.
 - To reference this value in system configuration, use the syntax `file-storage://XXXX`, where `XXXX` is the extendable lookup value. Using the same example from above, if you define an extendable lookup value of `CM-FileExtract` with a file path of `h:\oracle\serverName\1.0.0.0\batch\extract\`, then when configuring the extract file path for an extract

batch job, enter **file-storage://CM-FileExtract**. Another example is that if the extendable lookup value's file path is configured as **h:\oracle\serverName\1.0.0.0\batch**, then the user configuring the file path on the batch extract can enter **file-storage://CM-FileExtract/extract**

When should you define substitution variables in the properties file and when you define them in the extendable lookup? The following points highlight differences between the options that may help this decision.

- Substitution variables for URLs are only supported via the properties file.
- The properties file can typically only be modified by a system administrator. If there are values that are set at installation time and don't change, then defining the values in the properties file may be beneficial.
- When defining additional environments, such as test environments or production, the values in the extendable lookup may be copied using CMA. Ideally the values are defined such that they are the same between various regions.

External File Storage

The system supports using Oracle Cloud Object Storage for managing files in cloud implementations and can be configured to read files from or write files to this external location. Refer to Oracle Cloud Object Storage documentation for more information about obtaining an account and defining appropriate file locations or "buckets".

Once your cloud storage information is defined, the following points highlight the configuration steps required in the application.

- Define a [signature key ring](#). When the system tries to communicate with cloud object storage, it must provide a signature key so that cloud storage can confirm that the request is from a trusted source.
 - Navigate to [Key Ring](#) in add mode and select the **Signature Key Ring** business object.
 - Define a key ring code, which will be used in the File Storage configuration (below) along with a description.
 - Once the key ring is added, click **Generate Key** to generate a private / public key pair.
 - Click View Public Key to launch a pop-up that displays the public key, allowing the user to copy the key.
- At this point, the user should navigate to cloud object storage and register the public key. Once this is done, the key ring is now ready to be configured in file storage configuration.
- Use a File Storage extendable lookup to define the location and connection information so that you may reference this location in system configuration. The following points highlight the steps to take for this option.
 - Navigate to [Extendable Lookup](#) and search for the **File Storage Configuration** lookup.
 - Click **Add** to create a new entry. Define a lookup value name. Note that the lookup value should not have a slash or backslash in its name. This will be used when configuring a file path that uses this value. Choose the File Adapter value of **Oracle Cloud Object Storage**. Provide the following information that identifies the cloud storage options: User, Tenancy, Compartment, Namespace and Region. For the Key Ring, choose the value defined above.
- To reference this value in system configuration, use the syntax **file-storage://XXXX/...**, where XXXX is the extendable lookup value and any additional path information that is appropriate. For example, if you define an extendable lookup value of **CM-CloudStorage** and you have a bucket defined in Cloud Storage for processUpload, when configuring the file path for an upload batch job, enter **file-storage://CM-CloudStorage/processUpload**.
- You may optionally use a Bucket Name Prefix on the file storage configuration to support defining different Bucket Names for different environments. For example, imagine you want to have a separate "processUpload" bucket for a development region, a test region and a production region. In the Cloud Storage configuration you can define separate buckets for "DEV_processUpload", "TEST_processUpload" and "PROD_processUpload". If you needed to define the full bucket on each file path reference on various batch jobs, migrating the batch job configuration from one environment to another would require a manual step to change all the bucket references. Instead, the system supports defining a Bucket Name Prefix, which if populated will be prepended to bucket reference before connecting to Cloud Storage. This means that the prefix is defined once and all the references throughout the system can keep the common part of the

Bucket name (such as "processUpload"). Then only the File Storage configuration needs to differ between the different regions.

NOTE: The initial key pair is generated in the **Active** state. Once the key ring is defined on a File Storage lookup, it is recommended to practice key rotation and generate new keys periodically. To do this, use the **Generate Key** button on the Key Ring. New key pairs are generated in the **Inactive** state. The user should register the public key of the new key pair with cloud storage. Once that is done, the new key should be activated. The system generates the signature for connecting to cloud storage using the private key of the **Active** key pair for the key ring.

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 and in the batch thread pool worker. These are referred to as the "application caches". Some examples of application caches include

- System messages
- Field label and other field information
- Security Information

The framework product provides many specific caches for commonly used (and infrequently changed) data. In addition, specific edge applications may introduce additional caches as appropriate.

Information may also be cached on each user's browser.

The following topics highlight information about refreshing the various caches.

Server Cache

The server cache refers to data that is cached on the web server. An important use of this cache is for users' online access to the application. The caches aid in better performance while navigating throughout the system, allowing for data to be accessed from the cache rather than by always accessing the database. Besides user access to the web server cache, other functionality deployed to the web server uses caches in a similar way. For example, web services are deployed to the web server and access their own version of the cache.

The contents of the cache are cleared whenever the web server is restarted. This means that fresh values are retrieved from the database once users and web services start using the application again.

The product also supplies a flush command that one can issue in the browser's URL to immediately clear the contents of the cache. The command **flushAll.jsp** flushes every cache.

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

You would issue the following command in your browser's address bar: **http://OU-Production:7500/flushAll.jsp**. Notice that the command replaces the typical `cis.jsp` that appears after the port number.

If your system has been configured correctly, the **flushAll** command will submit a request to do a "global" flush of caches (including the web services cache and the thread pool worker cache). This functionality uses a JMS Topic to publish the flush request. Refer to the *Server Administration Guide* for details on how to configure the JMS topic.

Also note that the system provides a business service **F1-FlushAllService**, which is equivalent to the `flushAll` command (except that it does not restart your browser session). This may be useful for any processes that are built to orchestrate updates to configuration data where you want to ensure that the cache is immediately flushed. This business service is also exposed as a REST web service.

Batch Cache

When submitting a batch job, the batch component uses a Hibernate data cache to cache administrative data that doesn't change very often. The tables whose records are included in this cache are configured using the Caching Regime value of **Cached for Batch**. This is often referred to as the "L2 cache". Refer to [Tables](#) 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.

In addition batch jobs may also access application caches when applicable. When starting a thread pool worker, application data that is cached is loaded and cached for as long as that thread pool is running.

If there are any batch jobs that update any tables that are marked as **Cached for Batch**, you must run that batch job using a special thread pool that is configured with the caching disabled. Contact your system administrator to confirm the thread pool that is configured this way.

If there is a change in cached data that should be available for the next batch job, the following points highlight how the cache can be refreshed:

- By default the system is configured to automatically refresh the Hibernate cache every 60 seconds. However, an implementation may override the configuration to either change the number of seconds between intervals or to disable the automatic caching altogether. Application caches used by the batch jobs are not impacted by this refresh.
- Restart the thread pool workers.
- Run the **F1-FLUSH** (Flush all Caches) background process. This background process will flush the application data cached for all thread pool workers for all thread pools.
- If your the region has configured the thread pool workers to "listen" to requests for global flush as described in the [Server Cache](#) section, the thread pool worker caches are also refreshed when a **flushAll** command is issued.

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 (note that depending on the browser you use, the steps may vary slightly):

- 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.

Expression Parser

The product provides support for defining expressions that may be of a mathematical or logical/boolean nature. The expression may include variables and functions.

The data explorer [column parameter](#) is an example of where this may be used. That parameter supports the definition of a formula. Edge applications may include support for a formula or expression using this parser as well. For example, several application include a type of 'rule' object (calculation rule, form rule or usage rule) that is used for validation or calculation that may support applying a formula.

The following tables highlight what is supported in the expressions that use this parser.

Category	Supported in Expression	Description
Data types	Number	
	String	
	Boolean	
	List	
Literals	Numbers	
	Strings surrounded with either single quote or double quote. NOTE: 'Escaping' special characters is not currently supported.	
	Boolean values: true and false .	
Operations	+	Plus
	—	Minus
	/	Division
	*	Multiplication
	^ or **	Power
	%	Modulus
Logical operations	=	Equal
	>	Greater than
	>=	Greater than or equal to
	<	Less than
	<=	Less than or equal to
	!= or <>	Not equal to

This table identifies the functions that are supported. Note that several of the functions are applicable to a list of values. Note that although the functions are listed in lower case, the column parameter syntax in data explorer indicates referencing the functions as all capital letters. The system converts the data explorer column formula to lowercase before being evaluated.

Function	Parameter	Results	Comments
size()	List element	Number of elements in the list.	
isEmpty()	List element	Returns true if the list is empty.	
sum()	List element of type 'number'	Returns the sum of the numbers in the list.	
avg()	List element of type 'number'	Returns the average of the numbers in the list.	
	One or more numbers separated by commas	Returns the average of the number arguments.	
max()	List element	Returns the largest value in the list.	
	One or more comparable elements.	Returns the largest value of the number arguments.	
min()	List element	Returns the smallest value in the list.	
	One or more comparable elements.	Returns the smallest value of the number arguments.	
abs()	Number	Returns the absolute value.	
ceiling()	Number	Rounds the number to the ceiling.	
exp10()	Number	Raises 10 to the number power.	
acos()	Number	Returns the arc cosine of the number in radians.	The result will lose precision, as it uses double float based functions.
asin()	Number	Returns the arc sine of the number radians.	The result will lose precision, as it uses double float based functions.
atan()	Number	Returns the arc tangent of the number radians.	The result will lose precision, as it uses double float based functions.

Function	Parameter	Results	Comments
cos()	Radian	Returns the cosine of the radian angle input.	The result will lose precision, as it uses double float based functions.
exp()	Number	Raises e to the number power.	The result will lose precision, as it uses double float based functions.
log10()	Number	Takes the log, base 10, of the number.	The result will lose precision, as it uses double float based functions.
log()	Number	Takes the natural log (base e) of the number.	The result will lose precision, as it uses double float based functions.
sin()	Radian	Returns the sine of the radian angle input.	The result will lose precision, as it uses double float based functions.
sqrt()	Number	Returns the square root of the number.	The result will lose precision, as it uses double float based functions.
tan()	Radian	Returns the tangent of the radian angle input.	The result will lose precision, as it uses double float based functions.
floor()	Number	Rounds the number to the floor.	
round()	Number	Assumes a scale of 0. The default rounding mode of "round half up" is applied.	
	Number, Scale	The default rounding mode of "round half up" is applied.	
	Number, Scale, Mode	The mode must be set to one of the following: <ul style="list-style-type: none"> • "ROUND_CEILING" • "ROUND_DOWN" • "ROUND_FLOOR" • "ROUND_HALF_DOWN" • "ROUND_HALF_UP" • "ROUND_HALF_EVEN" • "ROUND_UP" • "ROUND_UNNECESSARY" 	
negate()	Number	Returns the negative value of the number.	Only available in data explorer.

The following are special functions supported in the application for a list of values. In each case, the syntax is *function* [*indexVariable* in *listName* | *expression using indexVariable*], where the *indexVariable* is chosen by the formula writer to represent each entry in the list and the expression used to evaluate each entry must reference that variable.

NOTE: The syntax supported for a given use of the formula in a functional area is driven by that particular functional area. For example, in Oracle Public Sector Revenue Management, a formula is supported in the "conditional element validation" form rule. In that form rule all variables including lists are declared in the form rule using letters and the formulas in turn use these letters. In that scenario, the functions below would reference the declared variable letter as the "listName". Other specific functional area that use this expression parser may support different syntax for referencing elements or lists.

Function	Description	Examples
any []	This function returns the value true if any of the entries in list satisfies the expression.	The following returns true if any entry in the Balance list is greater than 0. any [i in list/Balance i > 0]
all []	This function returns the value true if all of the entries in the list satisfy the expression.	The following returns true if all phone numbers are populated. all [i in list/phoneNumber i != ' ']
collect []	This function returns a new list of elements from the referenced list where the value of each entry of the new list is the result of the expression applied to each original value.	The following returns a new list with the tax rate applied to each amount. collect [i in list/amount i * taxRate]
select []	This function returns a list of all the values of the original list that satisfy the Boolean expression.	The following returns a new list with only the amounts that are negative numbers. select [i in list/amount i < 0]

Function	Description	Examples
<code>reject []</code>	This function returns a list of all the values of the original list that do not satisfy the Boolean expression.	The following returns a new list with only the amounts that are not negative numbers. <code>reject [i in list/amount i < 0]</code>

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.

The following are only applicable in a non-Cloud environment:

- **Start Debug** starts a logging session. During this session the processing of various configuration tools objects 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.

NOTE: The buttons are secured. An application service **F1CONFIGLOG** has been provided for this functionality to allow implementations to restrict user access to these buttons. An additional application service **F1DEBUG** is checked when the environment is not a 'development' environment. This allows implementations to provide an additional layer of security for their production users.

The following are available to view log entries captured while using the system.

- **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.
- **Show Log** allows administrative users to view logs of other users. The following are examples of when this capability is needed:
 - Batch and integration logs are captured by special user codes created for this type of processing. These users typically do not represent a human and therefore wouldn't access logs using "show user log". Instead someone else would be accessing these logs.
 - Customer admins would like to retrieve logs for an end user possibly for support reasons. An end user might not have technical skills to gather or assess the logs nor would they be given the privilege to access the logs for security reasons
- Checking the **Global Debug** indication starts various tracing options that are added to the user log.

NOTE: Show User Log button is secured. An application service **F1USERLOG** has been provided for this functionality to allow implementations to restrict user access to this button. Only users with access to the **Administrative** access mode of the application service are allowed to view logs of other users. Such restrictions may be called for in production environments.

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.

Prepare Issue Details

When a user encounters an error with the product and submits a service request, information about the environment, the release and the configuration of the user is often very helpful in understanding the cause of the problem. The system provides a script to capture the environment details and issue description for reporting purposes.

To access the script use **Menu > Tools > Prepare Issue Details**. The option is also available in the help menu in the toolbar.

A pop-up window is provided to display the details that are being captured, including a generated Issue ID to uniquely identify the report. The pop-up allows the user to add additional information describing the specific issue. When the user presses Save, the details are added to the system log with the issue ID, which can be searched when investigating the problem. A message will be displayed to either confirm the details were successfully logged or indicate that an error occurred.

NOTE: The script and menu item are secured. An application service **F1-PREPISSUEDTLS** has been provided to allow implementations to restrict user access to this functionality.

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 [User — Characteristics](#) page, 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.

Additional 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 [SQL Statement](#) in the zone parameter details section.

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 page into which a user is taken when they drill down to the related object on an entry.
- The message associated with the To Do that appears on various pages. 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 entry [sort keys](#). Note that the processes that create To Do entries are responsible for populating the sort key values.
- 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 entry. Note that this value may be overridden by a **Calculate Priority** algorithm.
- An indication of whether a To Do of that type may be created manually by a user.
- The algorithms used to perform specific business rules for To Do entries of this type.
- The characteristics applicable to the To Do.

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.

The base package determines the initial To Do Role as follows:

- The system checks if an entry's message category / number is rerouted to a specific role. If so, it defaults this role. Refer to [Adjusting the To Do Role](#) for more information.
- There may be some other condition related to the object linked to your To Do entry that warrants an override role. A [To Do pre-creation plug-in](#) would be used to check the conditional logic and override the role, if applicable.
- If a Role wasn't determined in one of the previous steps and a Role is provided by the initiating process (algorithm or background 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.

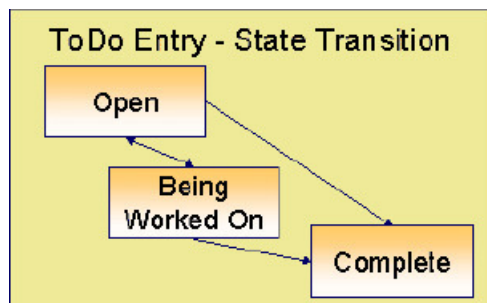
Supervisors for a Role

Most organizations have the notion of a supervisor who is responsible for all entries assigned to a given role. A user that is considered a supervisor should be a valid user in a role. In addition, application security is used to provide supervisors access to additional capabilities, namely the ability to review To Do entries assigned to other users and the ability to assign To Do entries to other users.

The pages listed in [To Do Supervisor Functions](#) are meant for supervisors only. So security for these pages should be restricted to those users that are considered supervisors. The [To Do Management](#) portal and the [To Do Search](#) page are available to all users. There is a special access mode for Supervisors to allow those users the ability to assign To Do entries to other users.

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. This may also be referred to as "reopening" or "unassigning" a To Do entry.

When an entry becomes **Complete**, it is no longer visible in various To Do 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.

- The To Do monitor process may be used to evaluate whether or not the To Do entry can be closed automatically. Refer to [Monitoring a To Do Entry](#) for more information.

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 [Suppress the Creation of the To Do Entry, Adjusting the To Do Role and Working on a To Do Entry](#) for more information.

Suppress the Creation of the To Do Entry

The system provides the ability to indicate that a To Do Entry should not be created based on conditions. When the system detects that no To Do should be created, there is no error. The system simply returns to the caller with an indication that no To Do was created. This functionality only applies to automatic To Do Entries created by an algorithm or background process. It does not apply to manually created To Do Entries.

Suppress Based on To Do's Message

There are some types of To Do Entries that are generated with a message that is specific to a condition that was found for a record. For example, business errors found when trying to update a record through a system process will generate a To Do entry using the error message as the To Do Entry's message. There may be some use cases where the error reported is not

something that a user needs to address and it may be beneficial to your organization to suppress the To Do Entry. Note that this is only appropriate if the message generated is for informational or warning purposes and doesn't require action or if you know that the error will be resolved by a subsequent step.

The configuration to suppress a To Do entry based on its message is on the [To Do type's message overrides](#) page.

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.

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 testing the system. During this period, compile a list of the messages that warrant special behavior described here.

Note that if the To Do type is configured to reference a specific message category / number, it means all To Do entries have the same message. If your organization does not want To Do entries created for this To Do type, then rather than using this suppression technique, you should identify the background process or algorithm that generates the To Do entries of this type and change the configuration there such that it does not try to create the To Do in the first place.

Suppress Based on a Condition

There may be use cases where a To Do entry should be suppressed based on some conditional logic. For example, consider the billing process in Oracle Utilities Customer Care and Billing. There is typically a bill cycle schedule that allows for attempting to bill all the accounts for the bill cycle over a multi-day window. There are some errors that occur on day 1 of the window that may be resolved by day 2 or 3. An implementation may choose to suppress To Do entries for this condition if this is day 1 or day 2 of the bill cycle. If the error still occurs on day 3, then the To Do entry should be generated. This type of condition needs to be determined using an algorithm. The To Do pre-creation plug-ins support returning an indication to suppress the To Do entry. Refer to [To Do Pre-creation Plug-ins](#) 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 that the [Plug-in Driven batch program](#) cannot be used for this purpose. This program assumes that any To Do Type that references it as a creation process is using the link for the [Object-Specific Error Condition](#) use case. If you want to use the plug-in driven batch process to create To Do entries based on a given condition, then you need to follow the pattern described in [To Dos Created by Background Processes for Specific Conditions](#).

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. This is applicable for processes that do not have built in error handling, for example where there is an explicit "error" state or where the record has an explicit "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. Do 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](#).

The functionality will only create a new To Do entry if there is not already an existing (non-complete) To Do for the same To Do type, drill key and message category / message number. It will also check for an existing To Do for a successfully processed record and complete that To Do.

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

Refining a To Do Entry's Details at Creation Time

When a To Do Entry is created, the key information provided to the system are the To Do Type and the drill key, which is the identifier of the object related to the To Do entry. With this information, system logic along with algorithms called at creation time can add details to the To Do entry or update important attributes so that the To Do can be worked on by the right team in the right priority.

The topics in this section will highlight the features provided to further refine a To Do Entry at creation time.

Linking Additional Information to a To Do Entry

Additional information may be linked to a To Do entry using characteristics. This may include master data so that the To Do entry is associated with one or more specific main objects in the system. It may also include other information that is helpful to any user that may be working on the To Do entry.

For **manual** To Do entries, 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. When create a manual To Do entry, the user populates the related characteristics as part of creating the To Do.

For **automatic** To Do entries, the program that generates the To Do may link related data to the To Do using characteristics.

In addition, the system [To Do Pre-creation plug-ins](#) can be used to link additional characteristics. These algorithms would use the drill key, which is the related object for the To Do entry. Based on the drill key, it can look up other information and create appropriate characteristics. Your specific product may be providing one or more algorithms to determine key master data records associated with the related object. For example, Oracle Utilities Customer Care and Billing provides an algorithm that attempts to link a related person, account, premise, service agreement and service point for that drill key. Other edge applications provide a similar algorithm for their relevant entities.

Any information that your user may need to have to help work a To Do entry should be linked to the To Do via a characteristic. For example, if the Rate Class of an account is important or the amount of the previous bill will help determine the priority of the To Do entry, this information may need to be linked as a characteristic.

NOTE: With the volume of To Do entries in the system, it is recommended to carefully consider what information you are linking to the To Do as a characteristic. Users can drill down on details when investigating a To Do entry. Only very important and critical information that would help users in reviewing To Do entries in the [To Do Management](#) portal and make decisions about prioritization and assignments should be linked as characteristics.

Note, before you can set up these algorithms, 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.

If your To Do entries reference characteristics that are related to your global context data, you may want to configure an alert algorithm to display an alert if a related entry is **Open** or **Being Worked On**. Refer to [Installation Options](#) for more information.

NOTE: Count of related To Do entries. There are some user interfaces that display a count of related To Do entries. This logic looks at foreign key characteristics linked to the To Do entry where the underlying object is a master or transaction object. It counts the distinct list of other non-closed To Do entries that reference the same characteristic(s).

Adjusting the To Do Role

When creating a To Do entry, for a To Do type, there may be attributes of the To Do entry that require that a special set of users work on it. This is sometimes referred to as 'rerouting the To Do entry' to a different role. This section highlights ways to override the role based on conditions specific to a To Do entry.

The information in this section only applies to automatic To Do Entries created by an algorithm or background process. It does not apply to manually created To Do Entries.

Refer to [To Do Entries Reference A Role](#) for the hierarchy used to determine the role .

Reroute Based on To Do's Message

As described in [Suppress the Creation of the To Do Entry](#), To Do entries created for a given To Do type may have different messages based on the specific condition for a record. You may decide that based on the message, the To Do should be routed to a specific Role that is more knowledgeable about a specific error condition. The same [message override](#) configuration that may be used for suppression of a To Do entry, may alternatively be used to assign a specific Role. For example, if certain bill segment errors are always resolved by the rate specialist team, you can indicate the role for that team on the To Do type, message override.

Reroute Based on a Condition

The [To Do Pre-creation plug-ins](#) may be used to assign a specific role based on some condition related to the To Do entry's specific data.

The product provides configuration on [access group](#) to define To Do assignment overrides. The override list defines a To Do type and To Do role. This is useful if your product associates access groups with specific master data for row level security purposes and you would also like to direct To Do entries related to this data to a specific role.

Oracle Utilities Customer Care and Billing also defines similar To Do assignment override configuration on CIS division and Account Management Group.

A pre-creation algorithm is responsible for checking if data linked to the To Do entry is related to a configuration object that supports an assignment override and then looks for an override for the To Do type. You can decide if this algorithm should be plugged in at the To Do Type level or the Installation level. The decision is based on how many To Do types you define overrides for. If it's a small number, then it makes more sense to plug the algorithm in for each To Do type that you define an override for. This ensures that the configuration is only checked for To Do entries that will have an override. However, if the number of To Do types that are eligible for overrides is larger or if the list is changing or growing often, it may be better to plug the algorithm in at the Installation level to ensure that every To Do is checked for a possible override.

NOTE: Note that your product may already provide a base delivered pre-creation algorithm that is checking for this override. If so, that can be used by your implementation. If not, your implementation can provide an appropriate algorithm.

To Do Pre-creation Plug-ins

The system provides two To Do Pre-creation plug-in spots that allow for functionality to add or update information on the To Do Entry that is about to be created. The two plug-in spots are [To Do type](#) and [installation options](#). In general, the installation option To Do pre-creation algorithms would be used for logic that applies to many To Do types and the To Do type pre-creation algorithms would be used for logic that is specific to that To Do type.

If at least one To Do pre-creation algorithm is plugged in on the To Do type, only those algorithms are executed when creating the To Do entry and the installation option algorithms are ignored. The system provides a base delivered To Do type pre-creation algorithm that executes the Installation option To Do pre-creation algorithms. This allows you to control whether the installation algorithms should also be executed for a To Do type that has specific algorithms and if so, when.

The following points highlight functionality supported by both To Do pre-creation plug-in spots.

- Indicate that the To Do entry should be suppressed (not created). The expectation is that this type of algorithm is configured on the To Do type rather than at the installation options level, as this allows for more granular conditions to be checked for a given To Do entry. Refer to [Suppress the Creation of the To Do Entry](#) for more information.
- Linking context specific data to the To Do entry using characteristics. Refer to [Linking Related Information to a To Do Entry](#) for more information.
- Overriding the Role of a To Do entry based on specific configuration related to the To Do's characteristic data. Refer to [Adjusting the To Do Role](#) for more information.

Assigning a Priority

As described in [The Priority of a To Do Entry](#), a To Do type may reference a Calculate Priority algorithm. If so, this algorithm is called as part of the steps to create a To Do entry. The algorithm may use the drill key or any of the related information linked to the To Do as described above to determine priority information.

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 assigned as follows:

- If one or more **Calculate Priority** plug-ins are defined on the To Do entry's type, the system calls them to determine the entry's priority. They are called initially when a To Do entry is created and each time it gets updated. 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. The system also provides a batch process (**F1-TDCLP**) that calls the calculate priority algorithms "at will" for non-closed To Do entries. This is useful when the priority should be 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. When the priority is determined by one of the algorithms, a log entry is created indicating that the priority was calculated.
- 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. When a user overrides the priority, a log entry is created indicating that the priority was overridden. 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 maintenance page associated with the entry (the To Do entry's drill key). The page to which the user is taken is configured on the To Do type.

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 specific page. The script can check conditions related to the To Do entry to determine a more appropriate page or could be configured to walk the user through the steps required to resolve the To Do entry.

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.

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.

Monitoring A To Do Entry

A To Do type may reference a To Do monitor [algorithm](#). The algorithms are executed by the To Do monitor background process (F1-TDMON). The process executes the monitor algorithms for any non-complete To Do entry whose To Do type references at least one monitor algorithm.

Monitor algorithms may be useful if there are business requirements where a To Do could be automatically completed based on criteria that the algorithm can review. For example, an algorithm may be used to detect if the situation that caused the To Do Entry to be generated has been remedied in the meantime, allowing for the To Do Entry to be completed.

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
- The To Do entry's priority was updated as a result of a calculate priority algorithm.

FASTPATH: For information about the contents of log entries for each of the events, refer to [Log Entry Events](#).

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 Routingalgorithms** 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.

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](#).

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

There are a number of installation options that may be configured to govern various aspects of To Do processing.

- A **To Do Information** algorithm may be plugged in on the Installation record. Refer to [To Do Entries My be Formatted By An Algorithm](#) for more information.
- 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. Refer to [To Do Pre-creation Plug-ins](#) for more information.
- A **Next To Do Assignment** algorithm must be plugged into the Installation record if your organization opts to use the next assignment feature supported by the Current To Do dashboard zone. The algorithm is responsible for determining 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. Refer to the [Current To Do](#) zone for more information.
- 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 an 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.

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-TDSK** has the ability to assign To Do entries to users that have the proper skills to work on them. This plug-in is 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.

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 > General > To Do Role** 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.

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 attributes that may be used to sort To Do entries.

The To Do Information zone on the To Do Management portal provides the ability to search for To Do entries using sort keys and includes up to five sort key values as columns in the results, which may be used for sorting as required.

The To Do list page includes an **Extra Information** option to allow To Do entries to be sorted by a selected sort key. Note the default sort key for this type of sorting is also defined on To Do type.

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 various pages.

Use as Default indicates the default sort key that is initially used when a user opens the To Do List page. 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 on the To Do List page.

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 [Suppress the Creation of the To Do Entry and Adjusting the To Do Role](#) 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 > General > 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**.

System Event	Optional / Required	Description
Calculate Priority	Optional	<p>Algorithms of this type may be used to calculate a To Do entry's priority. Refer to The Priority of a To Do Entry for more information on when this plug-in is called.</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>
External Routing	Optional	<p>Algorithms of this type may be used to route a To Do entry to an external system.</p> <p>The base package F1-TDEER background process invokes the algorithms for every To Do entry that its type references the process as the Routing Process 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>
To Do Information	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>
To Do Monitor	Optional	<p>Algorithms of this type are executed by the To Do Monitor background process and may be used to periodically review a To Do entry and perform actions, if needed. Refer to Monitoring a To Do Entry for more information.</p>
To Do Post-Processing	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>
To Do Pre-Creation	Optional	<p>Algorithms of this type are called when a To Do entry is being added. Refer to Refining a To Do Entry's Details at Creation Time for more information. To Do Type pre-creation algorithms override the Installation level pre-creation algorithms.</p>

Implementing The To Do Entries

To enable the To Do entries visible in the To Do Type page, 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.

Understanding 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 Overview

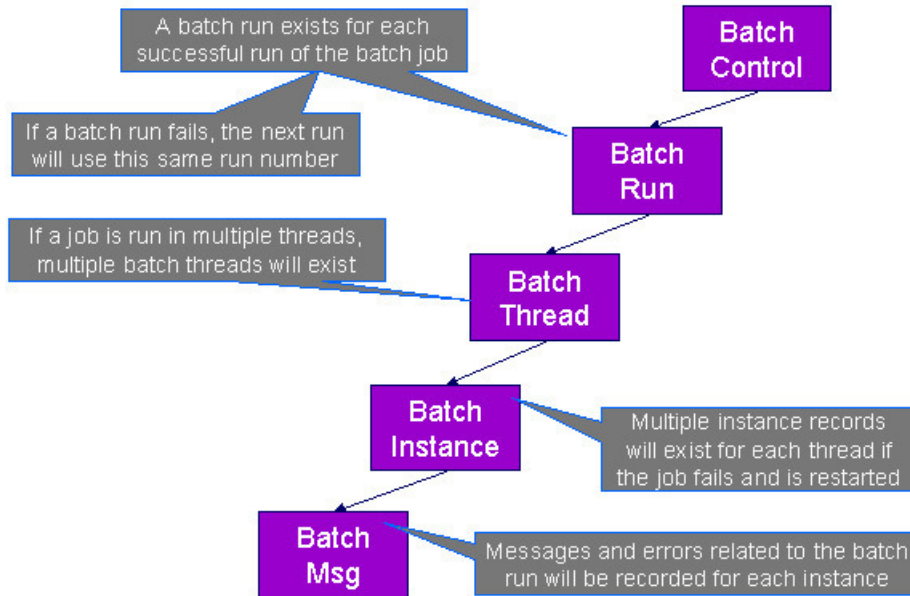
While the system relies on a scheduler to secure and execute its background processes, there are additional issues that you should be familiar with:

- 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.

Refer to [Batch Scheduler Integration](#) for information about the integration with the Oracle Scheduler.

Parallel Background Processes

Many processes have been designed to run in parallel in order to speed execution. This is referred to as running the process with multiple “threads”.

The system provides two strategies for distributing the data to the multiple threads.

- **Thread Level SQL Select.** This strategy is sometimes referred to as the “thread iterator” strategy. In this strategy, the batch job uses the primary key to figure out how to evenly distribute key ranges to each thread. Each thread is then responsible for selecting the records. In this strategy, the threads should also re-select the data periodically to release the cursor, which aids in performance. Note that this strategy is preferred but may only be used under the following conditions:

- The data from only one maintenance object is being processed.
- The primary key for the maintenance object is a single, numeric system generated key.

NOTE: Parameters may be used to override the low and high id. Refer to [Parameters Supplied to Background Processes](#) for more information.

- **Job Level SQL Select.** This strategy is sometimes referred to as the “standard commit” strategy. In this strategy, the keys for the records to be processed by the batch job are all selected first and stored in a temporary table. The batch job then supplies each thread with a range of keys that it should process. This strategy is used if multiple maintenance objects are being processed by the batch job; if the primary key of the maintenance object has multiple parts or if the primary key is non-numeric.

The multi-threading logic relies on the fact that primary keys for master and transaction data are typically system generated random keys. In addition, if the data is partitioned, it is expected to be partitioned based on the primary key.

NOTE: The detailed description in the metadata for each batch control provided with the system should indicate if it may be run in parallel. Note that the strategy used is not typically indicated in the detailed description.

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.

Parameters Supplied To Background Processes

This section describes the various types of parameters that are supplied to background processes.

General Parameters

The following information is passed to every background process.

- **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, a billing process may use 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.
- **User ID.** Please be aware of the following in respect of user ID:
 - Both the user submitting the job and the user ID recorded on the batch submission should have access to the application service for the batch control that secures execution.
 - Any batch process that stamps a user ID on a record it creates or updates uses this user ID in applicable processing.
 - 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 Start, Trace Program Exit, Trace SQL and Trace Output.** These switches are only used during QA and benchmarking. If trace program start is checked, a message is displayed whenever a program is started. If trace program exit is checked, a message is displayed whenever a program is exited. If trace SQL is checked, a message is displayed whenever an SQL statement is executed. If trace output is checked, special messages formatted by the background process are written.

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.

Common Additional Parameters

Each batch control supports the definition of additional parameters. There are some additional parameters that are common to all batch processes or common to a specific type of batch process. The batch control should be delivered with the appropriate additional parameters. However, when new additional parameters are introduced, existing batch controls may not be updated with the new additional parameter.

The following table highlights the common parameters that may be linked to a batch control. Note that for batch parameters, although there is a sequence number that controls the displayed order of the parameter, the batch process does not use the sequence to identify a particular parameter but rather uses the parameter name. In some cases multiple parameter names are supported (a 'camel case' version and an 'all caps' version).

Parameter Name	Description	Additional Comments
MAX-ERRORS / maxErrors	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 that constant using this parameter.	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.
DIST-THD-POOL	Each batch process executes in a thread pool. This parameter is only necessary if the batch process should execute in a different thread pool than the default thread pool.	The default thread pool name is DEFAULT .
emailMode	When the batch job is submitted with an associated email address, the default logic is to send an email when the job completes regardless of success or failure. Use this parameter to limit the email based on the status of the job when it ends.	Valid Values <ul style="list-style-type: none"> • ERROR — send an email only when the job ends in Error status. • SUCCESS — send an email only when the job ends in successfully. • ALL — always send an email only when the job ends. (This is the default.)

The following parameters are only applicable to jobs that use the **Thread Level SQL Select** method of distributing work to threads as described in [Parallel Background Processes](#).

overrideLowIdValue	Specifies a new low id to use in calculating the range for a thread. The framework by default assumes that the Id is between 0's (e.g. 00000000) and 9's (e.g. 9999999999), but this parameter will override the low value.	The parameter value can be an actual number or it can be set to auto . If auto is configured, it is set to the lowest current value on the database table associated with the background process.
overrideHighIdValue	Specifies a new high id to use in calculating the range for a thread. The framework by default assumes that the Id is between 0's (e.g. 00000000) and 9's (e.g. 9999999999), but this parameter will override the high value.	The parameter value can be an actual number or it can be set to auto . If auto is configured, it is set to the highest current value on the database table associated with the background process.
idRangeOverrideClass	Use this parameter to specify a custom class to do thread range calculation. During batch execution, this override class is instantiated and the setter methods called to initialize the Ids as required. The low and high getter	The class name specified must implement interface <code>com.splwg.base.api.batch.BatchIdRangeOverride</code> .

Parameter Name	Description	Additional Comments
	methods are called to retrieve the high and low ids to be used for the run.	
The following parameters are only applicable to jobs that perform a single commit, for example for extract batch jobs.		
numRecordsToFlush	This parameter defines how frequently to flush the Hibernate cache to prevent high heap consumption and Out Of Memory Errors.	

Specific Batch Parameters

Some background processes define 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.

Indicating a File Path or File Name

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.

For the file path parameter, refer to [Referencing URIs](#) for several points related to populating the file path and the supported syntax.

For the file name parameter, there are extract processes that support defining a file name that ends in ".zip" or ".gz", which will automatically result in writing the file with zip or gzip compression (based on the suffix). Similarly, there are upload processes that support decompression of files using the appropriate method based on the file name ending in ".zip" or ".gz". If the extract or upload process supports compression / decompression, it should be indicated accordingly in the description of the File Name parameter. If a batch control doesn't explicitly indicate that it supports compression or decompression in its description, the assumption should be that it is not supported.

Manifest File

Batch processes that create files, for example data export related batch processes, may create one or more files in each processing thread. External applications that consume these files need to process them only when the entire batch process is successfully completed. To automate the consumption of such files, the external application may leverage a function to be triggered upon the creation of a manifest file to process all the files referenced by the manifest. The manifest file includes details such as content hash value, file size information etc that assist in confirming that file content is completed and has not been tampered with prior to processing the files.

The following sections provide additional information about manifest files.

The Manifest Option Batch Parameter

Manifest file creation is supported on certain base product extract related batch controls only. These batch controls include the **Manifest Option** batch parameter that controls whether a manifest file should be created or not upon successful completion of the batch process. You may use the Batch Control Query portal to find all batch controls supporting this option.

Manifest File Information

Manifest file content is provided in JSON format and includes the following information:

- The version of the manifest details structure.
- The batch control code, run number and rerun number identifying the batch run producing the files

- Total number of files created by the batch run.
- Total number of records processed by the batch run.
- The manifest creation timestamp in ISO format.
- A list of files created by the batch run that includes the following details for each file: the file's name, its size in bytes and the SHA-256 hash value of the file's content.

The manifest file name is constructed as follows:

```
[batch control]_[batch run number]_[batch rerun number]_manifest.json
```

The following is an example of the manifest file structure.

```
{
  "version": 1,
  "batchCode": "HD-PDE",
  "batchRunNumber": 106,
  "batchRerunNumber": 0,
  "totalFiles": 3,
  "recordsProcessed": 10,
  "timeStamp": "2020-05-02T16:55:43.000353Z",
  "fileDetails": [
    {
      "fileName": "SOME_FILE_NAME_001.json.gz",
      "fileSize": 1543,
      "hashAlgorithm": "SHA-256",
      "hashValue": "ff7d4e466f93cadc4a18ab009080916baff631e03b728e2f2684c4a4f92ccbe1"
    },
    {
      "fileName": "SOME_FILE_NAME_002.json.gz",
      "fileSize": 1182,
      "hashAlgorithm": "SHA-256",
      "hashValue": "78869e7ce971f28a70f5b35693a05726ba66109fecf492edb60291a62ae1ba9f"
    },
    {
      "fileName": "SOME_FILE_NAME_003.json.gz",
      "fileSize": 1182,
      "hashAlgorithm": "SHA-256",
      "hashValue": "78869e7ce971f28a70f5b35693a05726ba66109fecf492edb60291a62ae1ba9f"
    }
  ]
}
```

Verifying The Manifest File

The **Verify Manifest File** web service may be used by an external system to validate that the manifest file it has received for a specific batch run is not corrupted or tampered with. The service returns the manifest file name and the hash value of its content for the requested batch run allowing the caller to compare their file details with the returned values.

Sharing Customer Exported Data

Some data export related batch processes support an option that streamlines the process by which customer extracted data may be routed to and shared with Oracle so it can be uploaded to Oracle's data lake for analytic purposes. These batch controls include the **Copy to Oracle** batch parameter that controls whether a copy of the exported file should be sent to Oracle's data lake. You may use the Batch Control Query portal to find all batch controls supporting this option.

NOTE: The batch parameter is set to not share any content by default. Customers may opt in to share their data with Oracle by submitting such a batch process with the parameter explicitly set to do so.

NOTE: This option is only available in a cloud installation.

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.

Error Post-Processing Logic

The product supports executing one or more algorithms when a batch process encounters an error that causes execution to stop. This allows for some special processing to occur to handle the failure of the batch job. Algorithms for this plug-in spot receive the batch control, batch run number, batch processing business date, number of threads and the list of the ad hoc parameters of the batch job.

NOTE: This plug-in spot is available for all Java based batch programs. For programs of type **Java (converted)**, the individual batch programs need to explicitly support this plug-in spot. Unless otherwise noted, assume that a **Java (converted)** program does not support it.

The following are some examples of functionality that may be executed when a batch job fails:

- Another object or record that is monitoring the batch job may have its status updated to reflect the batch status.
- An outbound message service may be invoked to perform a task related to the failure.

Note that the units of work for all threads are committed prior to executing the error post-processing logic.

Post-Processing Logic

The product supports executing one or more algorithms after all the threads of a given batch job have completed. This allows for some special processing to occur at the end of a batch job. Algorithms for this plug-in spot receive the batch control, batch run number, batch processing business date, number of threads and the list of the ad hoc parameters of the batch job.

NOTE: This plug-in spot is available for all Java based batch programs. For programs of type **Java (converted)**, the individual batch programs need to explicitly support this plug-in spot. Unless otherwise noted, assume that a **Java (converted)** program does not support it.

The following are some examples of functionality that may be executed at the end of a batch job:

- Another dependent batch job can be kicked off. Note that this use case is only needed when the multiple dependent jobs are not part of a scheduler (which can also detect the successful end of one batch job so as to submit the next job).
- Statistics for the batch run may be analyzed and based on results, a To Do Entry may be sent to an administrator.
- If the current batch job is processing a large number of child records in multiple threads, a parent record could be updated to a different status or with some other audit information.

Note that the units of work for all threads are committed prior to executing the post-processing logic. The algorithm should perform standard error handling. If an error occurs in one of the post-processing algorithms, the overall batch job's status is set to Error so that it can be re-submitted to retry the logic in the finalize step.

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.

Monitor Background Processes

In many areas of the system, functionality is driven from business object configuration as a BO driven record progresses through its lifecycle. Refer to [Business Object Lifecycle](#) for details. As part of that functionality, it is possible that a background process, called a [monitor batch process](#), is used to execute functionality for the record. A single program is provided for the BO monitor functionality. The product also provides a batch control template (**F1-MNTPL**) that should be used as a starting point. Parameters are used to limit the records processed by maintenance object and other optional parameters that may further limit the records. The product typically provides at least one monitor batch control for each maintenance object that supports a configurable lifecycle on its business object.

This topic highlights the parameters supported by the monitor batch job. Not all parameters are applicable to all maintenance objects and therefore may not be configured on a given base monitor batch control.

Parameter Name	Description	Comments
maintenanceObject	Maintenance Object	For most base delivered batch controls, this parameter is delivered already populated with the value of the maintenance object value. Note that it is supported to leave this value blank, at which point, the program will determine the maintenance object (objects) to process by looking for an MO that refers to this batch control record as an option.

Parameter Name	Description	Comments
isRestrictedByBatchCode	Restrict by Batch Code	Set this to true to indicate whether the process should only select records that explicitly refer to this batch control on its current BO state. This is also referred to as "deferred" mode. If set to false , the program includes all records that refer to the current batch control in its BO state <u>and</u> records that don't refer to any batch control in its current state (but monitor algorithms exist in the current state). This is commonly referred to as "periodic" mode. Note that if the value is not set at all, the program will determine whether to run it as "deferred" or "periodic" based on whether the batch code is configured on the MO option as a State Monitor Process ("deferred") or a Periodic Monitor Process .
restrictToType	Restrict by Related Type	This parameter is only applicable to maintenance objects that have a related 'type' object and the maintenance object has configured an option indicating the field for the related type column. This parameter may be used to limit the processing to records that are in the indicated type.
restrictToBusinessObject	Restrict by Business Object	This parameter may be used to limit the processing to records that are in the indicated business object.
restrictToBOStatus	Restrict by Status	This parameter may be used to limit the processing to records that are in the indicated status.
restrictToDate	Restrict By Date	Enter a valid date on the record's primary table to limit processing to records that are have a value in this field that is on or before the batch business date. Optionally, enter "+" or "-" followed by a number to shift the comparison date to the batch business date plus or minus the given number. For example, entering REVIEW_DT -3 will retrieve all records whose review date is on or before the batch business date minus 3 days.
sampleRecordNumber	Sample Record Number	This is not a commonly used parameter. It is only applicable when the monitor is used for a business use case that supports processing a subset of the records during a testing phase. For example, if the process is validating a large number of records, it may be an option to only validate every 100 records to determine if there are repeated validation errors that may indicate a common problem that may be solved to fix many errors.

Also note that when submitting a monitor process with multiple [parallel threads](#), the program will use a **Thread Level SQL Select** strategy unless any of the following are true (in which case it will use the **Job Level SQL Select** strategy:

- The input maintenance object is left blank and the program finds more than one maintenance object that refers to this batch control in its options.
- A single MO is applicable but it has a multi-part primary key.
- A single MO is applicable and it has a single primary key, but it is a user defined key instead of a system generated key.
- The sample record number parameter is populated.

Plug-in Driven Background Processes

Although the product is delivered with a rich library of background processes, implementations may have business requirements that require new processes to be introduced. It is possible for an implementation to write a background process from scratch using a base process as a template. However, the product also provides base background processes that call algorithms to do the work that is needed. These are called plug-in driven background processes. There are two major types of plug-in driven batch processes:

- Processes that act on records that are stored in the database in the system. These types of processes require SQL to select the records along with the logic to process the records.
- Processes that import data from a file and store new records in the system as a result. These types of processes require an algorithm that is able to map the data from the file to appropriate new records in the system

The subsequent sections provide more detail about the two types of plug-in driven background processes.

Processing System Records

Processes that retrieve records in the system and do some action on them require an algorithm to select the records to be processed and another algorithm to process the records. The base processes implement standard background process functionality including parallel background process logic and the ability to create To Do entries for errors. This allows for an implementation to take advantage of the pre-built support and provide plug-ins that include the logic that is unique to the specific use case.

The system provides the following processes that support plug-ins for selecting and processing the records:

- Ad-hoc Process. This background process is provided for implementations that have some custom business logic that needs to be performed on a group of records. The base batch control Plug-in Driven Generic Template (**F1-PDBG**) may be used as a template.
- Extract Process. This background process is provided for implementations that have extract files to produce for integration with external systems. The process includes parameters to configure the file path and file name for the created file along with other parameters to control how the file is formatted. The base batch control Plug-in Driven Extract Template (**F1-PDBEX**) may be used as a template.

The following sections provide more information about the functionality that is common for both ad-hoc and extract processes. Refer to [Extract Record Processing](#) for additional information specific to extract processes.

Select Records Algorithm

The first important algorithm to design when implementing a plug-in driven batch process is the Select Records algorithm, plugged in on the [batch control](#) page. This algorithm type must define the first parameter as the SQL. The batch job will directly access the SQL parameter value in the metadata (rather than invoking the algorithm). All other parameters are available for the algorithm to use for its own logic.

In addition, when invoking the algorithm, it must return the strategy to use (**Thread Level SQL Select** or **Job Level SQL Select**). Refer to [parallel background processes](#) for more information about the two strategies and when to use each. When choosing the **Thread Level SQL Select** strategy, the algorithm should return the name of the primary key in the Key Field parameter. In addition, the SQL should include a **BETWEEN** clause that includes the bind variables for the low and high ID for the ranges. See below for the bind variable syntax.

If the SQL statement includes variables that are determined at execution time, it must use bind parameters. Bind parameters are referenced in the SQL statement using a colon and a parameter name (for example **:parameter**). There are some variables provided by the system that are populated by the batch job at execution time. These have **f1_** as its prefix.

The system supports the following pre-defined bind parameters:

- **:f1_lowID** and **:f1_highID** - these should be used in the **BETWEEN** clause for the **Thread Level SQL Select** strategy. The batch job will substitute the appropriate ID range as required.
- **:f1_batchCode** and **:f1_batchNumber** - these are common attributes of the batch control that are referenced on a record for selection purposes. Note that the batch run number is set according to whether the batch job is a re-run of a previous run or not.
- **:f1_businessDate** - the batch job will populate the input batch business date, if populated otherwise the current date.

NOTE: The system supports both "f1." and "f1_" as a prefix for the bind variables. However, the "f1." prefix will result in an error if the SQL security property setting is turned on. As such, the underscore syntax is recommended. The period syntax remains for backwards compatibility.

For any other custom parameters, the Select Records algorithm may return one or more sets of field name / variable name / value where the variable name matches a bind variable in the SQL. The field name provides information about the length that assists the SQL binding logic to properly substitute the values. Note that the variable name cannot start with **f1.** as its prefix. The batch job will use the value returned by the algorithm to set the bind parameter in the SQL statement. Also note that the logic binds all variables as a string, so for date variables the SQL must include the "TO_DATE" function.

The plug in spot receives a list of the ad hoc parameters for the batch job as name / value pairs. If the list includes parameters whose values are to be used in selecting records, your algorithm may be used to identify the relevant batch parameter passed as input and populate the field name and output bind variable appropriately.

The product provides a base algorithm type for this plug-in spot that simply defines a parameter for the SQL. It also includes parameters for the strategy and the key field name. This algorithm type may be used by any custom batch process where the SQL does not rely on any special bind variables that must be determined. Simply create an algorithm for the algorithm type and provide the appropriate SQL. Refer to the algorithm type Select Records by Predefined Query (**F1-PDB-SR**) for more information.

Process Records Algorithm

The other important algorithm to design when implementing a plug-in driven batch process is the Process Record algorithm, plugged in on the [batch control](#) page. This algorithm is called for each record selected for the process. It receives all the information that was selected from the Select Records plug-in.

For the ad-hoc processing batch process, algorithms plugged into this spot are responsible for doing the work for each record based on the desired logic.

For the extract batch process, algorithms plugged into this spot are responsible for returning the data that should be written to the file. Refer to [Extract Record Processing](#) for more information on the functionality specific to extract processes.

Also note that algorithms for this plug-in spot will be passed two Booleans, `isFirst` and `isLast`, to indicate if the current work unit is the first and/or last for that thread. This allows for the plug-in to do additional work if needed. Note that the `isFirst` indication is available for both types of batch processes, ad-hoc and extract. However, the `isLast` indication is only applicable for the file extract batch. For the ad-hoc batch process this value will always be set to **false**. Extracts will always execute in a single database transaction. In a single transaction run, any error causes the run to be aborted so that it restarts from the beginning when resubmitted. This is done to avoid partial files being written along with inaccurate setting of the `isLast` element.

Configuring a New Process

The following points summarize the steps needed to implement a new background process that acts on records in the system using plug-ins for the specific functionality:

- Verify the SQL that the background process should execute. Keep in mind that all the data selected in the SQL is available to pass to the plug-in that processes the records. If the performance of the background process is important, be sure to consult with a DBA when designing the SQL.

- If the SQL does not require any custom variables to substitute at runtime, create an algorithm for the base algorithm type **F1-PDB-SR** and configure the SQL. In addition, configure the strategy and the primary key name (for the **Thread Level SQL Select** strategy).
- If the SQL requires custom variables, a new [plug-in script](#) must be designed and coded to populate the variable names and values using the algorithm entity **Batch Control - Select Records**. Besides defining the variables, the algorithm must also indicate the strategy and the primary key name (for the **Thread Level SQL Select** strategy). Define the algorithm type for the newly created script. The first parameter of the algorithm type must be the SQL as illustrated in the base algorithm type. Note that the other parameters are available for use by this algorithm type if needed. Define the algorithm, populating the SQL as appropriate (using the custom variables).
- Design the logic required to process each record. If your edge product does not deliver a suitable plug-in, create a [plug-in script](#) where the algorithm entity is **Batch Control - Process Record**. Note that the plug-in receives all the information selected in the SQL defined in the Select Records plug-in
 - For an ad-hoc process, the algorithm should perform whatever process is required based on the business use case. Note that the background process is responsible for committing the records.
 - For an extract process, the algorithm is responsible for returning one or more schema instances populated with information that should be written to the file. Refer to [Extract Record Processing](#) for more information and examples of extract algorithms.

In either case, if a new processing script is required, define the algorithm type and algorithm for the newly created script.

- Create a batch control by duplicating the appropriate base template as per the type of background process needed. Plug in the algorithms created above and configure the parameters as appropriate. Note that you may configure custom ad hoc parameters on the batch control if required. Both base and custom batch parameter values are available to the Select Records and Process Records plug in algorithms.

Extract Record Processing

For an extract batch process, the Process Records algorithms are responsible for returning the data that should be written to the file in one or more XML instances along with the schema name(s) that describes the XML instance(s). The program will write the data to the file as per the format indicated in the File Format batch parameter. By default the service uses the OUAF format for date and time. To override this and use XSD format, configure the Date Time Format batch parameter to 'XSD'.

If an existing schema satisfies the output requirements, it may be used. Otherwise, you should define a data area to indicate the output format of the records as appropriate.

There are two options for designing and configuring an extract process using a plug-in driven background process:

- Create a **Process Records** algorithm type that encapsulates the extract logic. The product provides a base algorithm type that illustrates the technique to follow. Refer to the algorithm type General Process - Sample Process Record Extract (**F1-GENPROCEX**) for more information.
- Create a file integration type that defines the records to be included in the extract and configure a plug-in driven batch control that references that integration type. The following sections provide more information about implementing this type of functionality.

File Integration Type Configuration

Your organization may have many types of extract file integrations. Some can be simple, containing a single record type and possibly a header or footer record. Others can be more complex, involving multiple record types and parent/child relationships between the records.

The framework supplies two objects that work together to define the processing needed to create a specific type of extract:

- The file integration type which describes and classifies the extract configuration.

- The file integration record where the sequence, format and logic to extract the data required for a given record type are captured.

File Integration Type

The file integration type defines the integration classification. Currently, only **Extract** is supported. In addition, the file integration type can be assigned to a category. The category may be used to identify extracts with a common purpose or target such as collection agency referrals. Your edge product may deliver a number of categories or you may define your own.

The system provides a business object for extract file integration types (**F1-ExtractFileIntegrationType**) that is expected to be used for most use cases

File Integration Record

The file integration record is the object that is responsible for defining a record's format and capturing the record details. A separate integration record must be defined for each record type included in the extract.

The record sequence defines the order that records are processed by the system. The parent sequence defines the record type of which this record is a child. Child records may also be referenced as a parent of lower level child records. The base Process Records algorithm provided by the framework processes all child records of a record type before processing the next record type in sequence.

For extracts that are XML format, the system supports creating a grouping node surrounding all the output that belongs to the same entity. File integration records define a record XML node that will be used as the grouping node. Each record type in the list that belongs within the same group would indicate the same value in the record XML node. Refer to the [Grouping by Record XML Node](#) section for more details.

NOTE: The system does not support nested record XML nodes. All child records of a record type must define the same record XML node as the parent.

Each file integration record references a data area that defines the extract record's format, allowing the data to be extended. The **Extract Records** algorithms are responsible for populating the data area. The file integration record supports multiple extract processing algorithms so that the custom information can also be populated.

The system provides a business object for extract file integration records (**F1-ExtractFileIntegrationRec**) that is expected to be used for most extract record use cases.

Configuring an Extract Process Using File Integration Type

The following points summarize the additional steps needed to implement a new extract background process that uses a file integration type to define the specific functionality:

- Choose or create data area objects to define the extract record formats.
- Define the logic required to populate each extract record type. Your edge product may deliver suitable plug-ins or you may need to create a [plug-in script](#) where the algorithm entity is **File Integration Record - Extract Record**. Note that the Extract Record plug-in receives most of the information defined in the Batch Control - Process Records plug-in.
- Define the algorithm type and algorithm for each newly created script.
- Choose or create a category for the file integration type using lookup **FILE_INT_OBJ_FLG**.
- Create the file integration type and configure the file integration records using the defined data areas and algorithms.
- Create a batch control by duplicating the appropriate base template for an extract background process. Include the File Integration Type parameter on the batch control and reference the appropriate file integration type.
- Plug in a Process Records algorithm that reads the file integration type configuration and processes the record types. The system provides a base file integration record extraction algorithm (**F1-FILEX-PR**) that is expected to be used for most file integration plug-in extract processes.

Extending a File Integration Extract Process

Your edge product may deliver file integration types and batch controls designed for standard extract processes. However, extending the delivered extracts to add custom data is common practice for most implementations. You can customize the output from a delivered extract by:

- Identifying the record types that need additional details
- Creating a data area that defines the additional details and using that to extend the data area configured on the record type
- Creating a File Integration Record — Extract Record [plug-in script](#), algorithm type and algorithm that populates the extra details
- Adding the algorithm to the integration record's extract algorithms collection

In some cases, you may wish to extend the extract by adding additional record types. To do this, add your custom file integration records to the file integration type, following the steps described in [Configuring an Extract Process Using File Integration Type](#).

Grouping by Record XML Node

When configuring an extract process that should produce the output in XML format, your Process Record plug-in may return multiple schemas with information that is all part of the same overall record. For example, the output could include account information and related service agreement details:

```
<account type="group"
  <accountId>1234567890</accountId>
  ...
</account>
<sa type="group"
  <saId>123457665</saId>
  ...
</sa>
<sa type="group"
  <saId>1234588913</saId>
  ...
</sa>
```

In this case, it may be desired to wrap all that information in a grouping XML tag so that all the information for one record is grouped together. Because the Process Record plug-in could be returning header or footer records that are not part of a given selected record's extract information, the batch process doesn't know what schemas returned by the Process Record plug-in belong together logically. The output parameter 'record XML node' in the schema collection should be used to indicate the outer XML node to use to group related information together. For example:

```
<SchemaInstance>
<recordXMLNode>record</recordXMLNode>
<schemaName>CM-AccountRecord</schemaName>
<schemaType>FlDA</schemaType>
<data><account><accountId>1234567890</accountId>...</data>
</SchemaInstance>
<SchemaInstance>
<recordXMLNode>record</recordXMLNode>
<schemaName>CM-SAInfo</schemaName>
<schemaType>FlDA</schemaType>
<data><sa><saId>123457665</saId>... </data>
</SchemaInstance>
<SchemaInstance>
<recordXMLNode>record</recordXMLNode>
<schemaName>CM-SAInfo</schemaName>
<schemaType>FlDA</schemaType>
<data><sa><saId>1234588913</saId>... </data>
</SchemaInstance>
```

All schemas returned from a single call to the Process Record plug-in that have the same record XML node will be grouped in the written output within that XML tag, as per the example below:

```
<root>
<record>
```

```

<account>
  <accountId>1234567890</accountId>
  ...
</account>
<sa>
  <saId>123457665</saId>
  ...
</sa>
<sa>
  <saId>1234588913</saId>
  ...
</sa>
</record>
<record>
<account>
  <accountId>987654320</accountId>
  ...
</account>
<sa>
  ...
</sa>
</record>
</root>

```

Skipping Records

By default, the extract process expects one or more schemas to be returned by each call to the Process Record plug-in. (Ideally when designing any process, the Select Records algorithm only selects records that should be processed and the Process Record algorithm does not need to confirm whether the record should be part of the processing or not.) If however there is a need to check a condition in the Process Record plug-in and it is found that no data needs to be extracted for this record, the process record algorithm should return the 'is skipped' output parameter to **true** so that the program doesn't write an empty row. This parameter is not applicable to the Adhoc Process batch program. In that case, the Process Record can simply not perform any action. No information needs to be returned to the program.

Note that the base process records algorithm for file integration types (**F1-FILEX-PR**) will set the 'is skipped' parameter to **true** if the file integration record extract record algorithm returns no data.

Uploading Records

The base product provides a background process to upload data from a file. The batch control Plug-in Driven File Upload Template (**F1-PDUPL**) may be used as a template. The process includes parameters to configure the file path and file name for file to upload along with other parameters to control how to handle missing files and how to rename the file once processed. Refer to the description of the batch control and its parameters for more information.

This background process requires an algorithm plugged into the plug-in spot **File Upload**. This plug-in is called once for a given file. The batch process opens the file and passes to the algorithm the file reader element. The algorithm associated with the batch control is responsible for using provided APIs to read the content of the file and store the data in appropriate table(s) (for example, an appropriate staging table). The base provided process supports uploading multiple files and may be run mutli-threaded to process multiple files. Each file is processed by one call to the File Upload algorithm and supports a single commit for the records uploaded in a given file.

NOTE: Plug-in scripts written to implement this type of algorithm must use the Groovy script engine version as the APIs are not accessible using the XPath scripting language. Refer to [Additional Coding Options For Server Scripts](#) for more information.

Note that this step in the upload of data is only one part of a typical upload end-to-end process. This step is sometimes referred to in the product as "Process X". The goal of this step is to get records from a file into database records, with minimal validation and processing. The data should be stored in records that are then processed by a second step (often referred to in the product as the "upload" step, for example "Payment Upload"). The second step, independent of the plug-in driven batch process described here, is responsible for validating the data and should be able to be threaded by individual records and have proper error handling for individual records. Note that depending on the type of data being

uploaded, the product may already supply appropriate tables that the plug-in driven upload process may populate. These could be staging tables, such as payment upload staging. Or they may be records with business objects that have a lifecycle designed to handle uploaded data, for example Business Flag. In such cases, the product will typically supply out-of-the-box background processes to validate and further process the data and finalize the upload. If the data to upload does not already have a base provided staging table, be sure to work with your implementation team to identify an appropriate table to use for the plug-in driven batch upload. In addition, confirm the design for the second step that is responsible for the detailed validation and finalization of the data.

The product supplies sample algorithms to illustrate calling the supplied APIs for processing different types of source data: comma delimited, fixed position and XML. In every case, the sample data supported by the upload uses 'degree day' information to illustrate the process. The system provides sample target records (based on the Fact maintenance object) in order to illustrate the step to store records based on the input data. Note that only sample plug-in scripts have been provided. No algorithm, algorithm type or batch control is configured to use the sample plug-in scripts. To view the scripts, navigate to the [Script](#) page, search for a Script Type of **Plug-in Script** and Algorithm Entity of **Batch Control - File Upload** and look for the 'Sample' scripts.

Note that the sample plug-in scripts provided by the product are supplied to illustrate use of the provided APIs. They should not be considered a template for how to implement a plug-in script for a real use case. The following highlight some points to consider when designing a file upload algorithm:

- **Error handling / resolution.** The sample plug-in scripts do some basic error handling related to the data to illustrate error handling. However, any errors found in this step require processing of the whole file to stop. As such, this plug-in should only report errors that are not possible to fix, but where the whole file should be rejected. If there are errors that can be adjusted in the data, then the recommendation is to not check for those errors at this step. Rather, this plug-in should simply populate the appropriate staging tables and let the next step check for validity. As described above, the next step should include the ability to mark individual records in error, allowing for users to fix the data and retry.
- **Target tables.** The sample plug-in scripts use Fact as the target for the resulting insert statements. As mentioned above, the decision of where to store the uploaded data must be carefully considered. There may already be existing tables that are specific to a given use case. If the data being uploaded does not have existing tables to use, review the product to verify what existing tables may be useful, such as Inbound Sync Request or Service Task. Be sure that the tables chosen support error handling, either out-of-the box or via designing an appropriate business object with a lifecycle that supports an error status and the ability to resolve the error. Also note that the Sample Flat File Upload plug-in illustrates a header record / detail record scenario. In this case, the header record is linked to the child record via a CLOB element. This is not the recommended technique. In a real use case, the header record should be linked to the child record via a separate database column to allow for searching.

Configuring a New Process

The following points summarize the steps needed to implement a new file upload background process:

- Verify the details of the data in the upload file and map the data to fields in one or more appropriate tables in the system.
- Design the logic required for reading the record details and identifying each record to properly create the insert statements for storing the data. The sample plug-in scripts provided by the product illustrate using the various APIs available for use. Create a [plug-in script](#) where the algorithm entity is **Batch Control - File Upload**. Create an appropriate algorithm type and algorithm for this plug-in script.
- Create a batch control by duplicating the base template. Plug in the algorithm created above and configure the parameters as appropriate. Note that you may configure custom ad hoc parameters on the batch control if required. Both base and custom batch parameter values are available to the File Upload algorithm.

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. Refer to [Batch Scheduler Integration](#) for information about the integration with the Oracle Scheduler.

Batch jobs may be configured as [Timed](#), which means they will automatically be run based on the timer frequency.

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.

The system provides a [REST web service](#) (using the inbound web service **F1-SubmitJob**) to create a batch job submission request from an external system. Note that no parameter defaulting occurs when submitting a record using a web service. All the desired parameter values must be provided.

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 one or more algorithms that check some conditions to assess whether or not the process is performing as expected. An algorithm 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 algorithms for a given background process, if desired. Each 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 are any level of service algorithms configured, the logic is executed and the results are displayed. The level of service is also part of the [Health Check](#) service.

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**.

Use the **Batch Control Query** portal to search for an existing batch control. Once a batch control is selected, you are brought to the maintenance portal to view and maintain the selected record.

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.

Enter the **Application Service** that is used to provide security for submission requests for the batch control. The application service must have an Access Mode of **Execute**. Refer to [Granting Access To Batch Submission](#) for more information.

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.
- **Purge.** Processes of this type are used to purge historical records from certain objects that generate a large number of entries and may become unwieldy over time.
- **To Do Entry.** Processes of this type are used to detect a given situation and create or complete a To Do Entry. Refer to [To Do Entries Created by Background Processes](#) for more information.
- **Upgrade.** Processes of this type are typically run only one time and are used to populate data in a table based on information that can be determined elsewhere in the system. These are processes that could have been done through an upgrade script, but perhaps due to data volume or timing, the upgrade logic warranted a separate stand-alone process.
- **Upload.** Processes of this type are used to upload data from a file into the system.
- The following categories are related to the data conversion / migration processes:
 - **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](#) algorithms for the batch control. If one algorithm is plugged into the Batch Control, the level of service lookup value along with a message indicating the reason for the output value is shown. If multiple algorithms are plugged in, the text **See Results for Details** is displayed. There is an icon provided to expand the details returned by each algorithm. 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. Any value entered here is displayed to a user when submitting a job through online [Batch Submission](#). Users may override the value before submitting the job. This includes resetting the value for an optional parameter. Regardless of how a batch job is submitted, if a value is not supplied for a required parameter, the default value define here, if found, is used. Optional parameters with no value supplied at submission time will not be defaulted to a value defined here. 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.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

The following table describes the **System Events**.

System Event	Optional / Required	Description
Error Post-Processing	Optional	Algorithms of this type are called if the batch process fails to complete due to an error. Multiple algorithms are allowed and are executed in sequence order. Refer to Error Post-Processing Logic for more information.
File Upload	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to upload all records for a file. Note that only one algorithm is allowed.
Level of Service	Optional	Algorithms of this type are called to determine the Level of Service provided by a batch control. Multiple algorithms are allowed. Refer to Assessing Level of Service for more information.

System Event	Optional / Required	Description
Post-Processing	Optional	Algorithms of this type are called after all threads are complete. Multiple algorithms are allowed and are executed in sequence order. Refer to Post-Processing Logic for more information.
Process Record	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to process a specific record.
Select Records	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to define the SQL to use to select the records to process. Only one algorithm is allowed.

NOTE: You can add new system events. Your implementation may want to add additional batch control oriented system events. To do that, add your new values to the customizable lookup field **F1_BATCH_CTRL_SEVT_FLG**.

Defining File Integration Types

This section describes the functionality supported for viewing and maintaining file integration types.

Refer to [Extract Record Processing](#) for an overview of the file integration type functionality provided with [plug-in driven background processes](#).

To view or add a file integration type, open **Admin > Integration > File Integration Type**.

Once a file integration type record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **File Integration Records** zone lists the record definitions for this integration, in hierarchical order. Use the zone actions to add, maintain or view a file integration record. Press the **Record Layout** icon to view a pop-up display of the record's schema.

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.

The email includes information from the product installation to help users that may be working in and submitting background processes in more than one environment.

- It includes the product name and product version for the product in the [installed products](#) list that is marked as the one to display.
- If a domain name message is configured in the [installation messages](#), this information is also included.

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.

NOTE: A batch control parameter may be used to indicate if the email should only be sent in the case of an error or success or both. Refer to [Parameters Supplied to Background Processes](#) for more information.

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. 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**. If a previous run ended with one or more threads in error, a new submission may be created to restart all the threads that were in error. This submission would result in new Batch Thread Instance records added for each thread that previously ended in error. Note that if the previous run was marked to "Do Not Restart", then a new submission will result in a new Batch Run.

- In the situation where a previous run resulted in one or more threads in error, you may use the Thread Number to indicate a specific thread to restart. This is only valid if there is a previous run found that is not marked "Do Not Restart". This will result in a new Batch Thread Instance created for the specific thread in the previous run.
- To run a process as a single thread, enter Thread Count = **1** and Thread Number = **1** (or **0**). 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

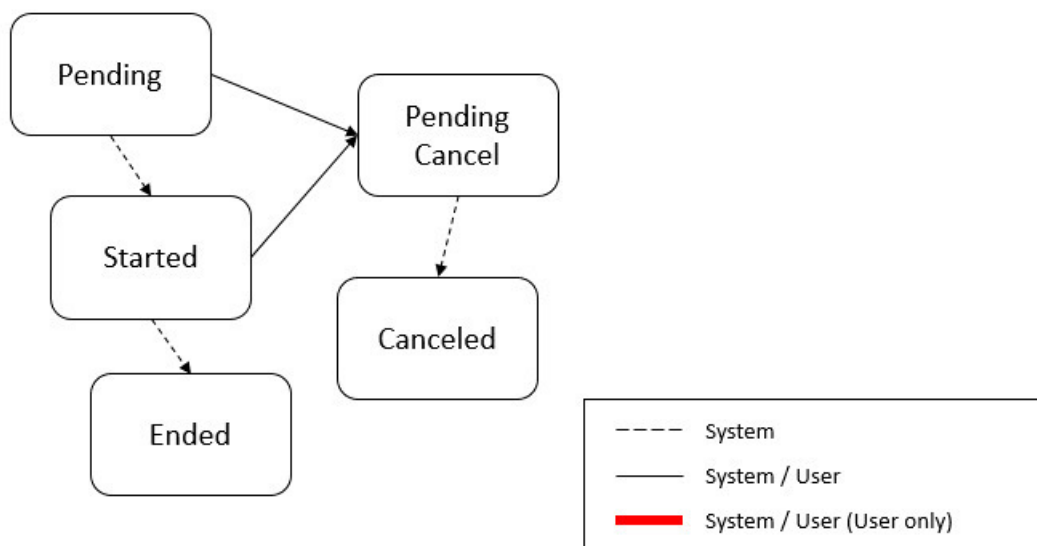
It is possible for your background process to end with an error. When this occurs, your batch job submission record will still be marked as **Ended**. You will need to navigate to the [Batch Run Tree](#) page to determine the status of the batch run.

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.

Started – Once a pending record has been submitted for processing, its status will be changed to **Started**.

Pending Cancel – Users may request the cancellation of a pending record to prevent the batch job from being submitted. In addition, users may request to cancel a started batch job that was not initiated by DBMS scheduler. The system will look for records in this status and look for batch runs that may need to be canceled for this batch job. Once it is confirmed that any batch runs are stopped, the status changes to **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.

Granting Access To Batch Submission

Every batch control must reference an [application service](#). When you link a batch control to an application service, you are granting all users in the group the ability to submit the associated batch job for execution.

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

When the batch control is added to the batch job table, the system checks if both the user submitting the job and the user ID recorded on the batch submission record have access rights. Application security does not apply when viewing a batch control or an associated batch run.

NOTE: Base batch controls will be associated with base owned application services by default. Implementations will need to ensure the appropriate user groups are linked to these application services.

Batch Job Submission - Main

This page allows you to submit a batch job on-line. Navigate to this page using **Menu > Tools > 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.

The following parameters are provided with each background process:

Thread Number should typically be set to 0. Refer to [Running Multi-threaded Processes](#) for situations where populating a value other than 0 are warranted.

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.

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.

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. It may differ from the user who submitted the batch, as described below. 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.

Language 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 **Submission Method** indicates how the batch job was initiated. Valid methods include:

- **Online** - populated when a user manually creates or manually duplicates a batch job submission.
- **Generated** - populated by algorithms that submit a batch job and by 'initiator' batch jobs that submit other batch jobs.
- **Scheduled** - populated by the DBMS scheduler. Note that for other schedulers, it is not advised to use this value as there may be logic and validation related to records created with this value that may not apply to other schedulers.
- **Timed** - populated by the batch daemon that submits timed batch jobs.
- **Other** - populated when no other value is provided.

Additional values may be added by your implementation. Refer to the lookup **SUBMIT_METH_FLG** for a list of the submission methods supported by your system.

The **Submission User** indicates who initiated the batch job. This value is not populated for timed batch jobs or batch jobs submitted via a scheduler. It is only displayed if the field is populated.

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** or a **Started** record that does not have the **Scheduled** submission method, use the **Cancel** button. The button is disabled for all other status values.

Tracking Batch Processes

The topics in this section cover the tools provided for reviewing the status and performance of batch processes. Refer to [Defining Batch Controls](#) for more information on how batch control codes are defined.

Batch Run Tree

The batch run tree pages show the execution status of a selected batch process. 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.

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 > Tools > 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 **Start Date/Time** and **End Date/Time** of the batch run are shown.

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. Within each thread, a node appears for each batch instance for the thread. A thread will only have more than one instance in the case of a thread that ended in error where a user subsequently restarted the run. The **Start Date/Time** and **End Date/Time** of the batch run are shown, if populated.

The messages that appear under a thread instance 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.

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.

If the application properties file has been configured with the location of the log files and the log files associated with the batch thread are still available, the links **Download stdout** and (if applicable) **Download stderr** are visible. Clicking either link allows you to view or save the log files.

NOTE: Compression. The log files are compressed and include a *.gz extension. Different browsers treat this type of file differently. Some browsers may automatically decompress the file as part of the download and they are viewable in any text viewer by changing the extension to one the text viewer recognizes. However, some browsers download the compressed file and a user will need to unzip the file prior to viewing.

NOTE: Security Access. The 'download' hyperlinks are only visible for users that have security access to the **Download** access mode for the batch run tree application service.

Batch Run Tree - Run Control

By default, if a batch process fails, it will restart. This tab allows you to modify the restart status of a failed run.

Navigate to this page using **Menu > Tools > 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:

- **Last Update Timestamp** contains the date and time the most recent batch run started or 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**.

Batch Day Tracking

The batch day tracking pages provide information to quickly determine the status and performance of the daily batch workload for a selected business date. The portals contain zones to provide details about run status and individual thread performance to assist in detecting data and performance issues with individual batch runs.

Batch Day Dashboard

This portal provides a high level summary of the status, duration and other details of the batch jobs that ran on a given business date. An implementation may introduce additional zones to this portal as needed.

Use **Menu > Dashboards > Batch Day Dashboard** to navigate to the portal.

Batch Runs By Business Day

This zone provides a summary by business date of the number of batch runs completed, in error, and in progress, together with the earliest start time and latest end time of the jobs on that day. The zone is sorted in descending order by business date.

NOTE: For performance reasons, the zone is configured to restrict the number of rows retrieved by the query. The 'Results' number will indicate how many rows have been retrieved. Use the Next and Previous buttons to retrieve additional dates or return to a prior set of dates.

The following filters are provided:

- **Batch Business Date** allows you to select a particular business date.
- **Process Date** allows you to restrict the batch runs included in the summary to those where the processing date is within the start and end dates for the run.
- **Exclude Reruns** allows you to exclude batch reruns from the summary.

NOTE: The values selected in these filters will be applied as hidden filters to the other zones in the portal.

When a specific business date is broadcast from the summary, the following zones will appear as part of the portal's **Main** tab page.

Batch Day By Status

This zone summarizes the batch runs for the business date by their status, displayed in the form of a pie chart. Selecting a specific slice of data on the chart broadcasts its value to the other zones on the portal allowing them to filter their data accordingly. Use the **Reset** action to reset the filter set by this zone.

Batch Day By Category

This zone summarizes the batch runs for the business date by their batch category, displayed in the form of a pie chart. Selecting a specific slice of data on the chart broadcasts its value to the other zones on the portal allowing them to filter their data accordingly. Use the **Reset** action to reset the filter set by this zone.

Batch Jobs Summary

This zone displays details of the individual batch runs for the broadcast business date. The zone is sorted by batch control, batch run number and batch rerun number.

NOTE: For performance reasons, the zone is configured to restrict the number of rows retrieved by the query. The 'Results' number will indicate how many rows have been retrieved. Use the Next and Previous buttons to retrieve additional rows or return to a prior set of rows.


The following filters are provided:

- **Batch Control** allows you to select a particular batch control to review. This filter supports wildcard searching.

- **From Start Date/Time** restricts the batch runs displayed to those where the start date time is on or after this date and time.
- **To Start Date/Time** restricts the batch runs displayed to those where the start date time is on or before this date and time.
- **From End Date/Time** restricts the batch runs displayed to those where the end date time is on or after this date and time.
- **To End Date/Time** restricts the batch runs displayed to those where the end date time is on or before this date and time.
- **Batch Control Type** restricts the batch runs displayed to the selected type.

The following columns are displayed initially:

- **Batch Control** is the batch control ID of the batch run.
- **Description** is the description of the batch control. Click the hyperlink to navigate to the [Batch Run Threads](#) portal.
- **Batch Run** is the run number of the batch job. Click the hyperlink to navigate to the [Batch Run Tree](#).
- **Status** is the status of the batch run.
- **Start Date/Time** is the start time of the run.
- **End Date/Time** is the end time of the run. The current date and time may be displayed if the batch job is still in progress.
- **Duration** is the time taken for the run, formatted as Days:Hours:Minutes:Seconds.
- **Threads** is the total threads for the batch run.

Click the **Column Configuration** icon  in the zone header to show the [Column Configuration Area](#) in which additional columns can be added to the results. The additional columns include the following:

- **Batch Category** is the category of the batch run.
- **Batch Control Type** is the type of the batch run.
- **Do Not Attempt Restart** indicates whether the run has been marked not to be restarted.
- **Rerun Nbr** is the rerun number of this batch job.

Batch Run Threads

This portal provides details and analysis of the threads for a given batch run. It is accessed via a link from the batch jobs summary zone on the [Batch Day Dashboard](#). An implementation may introduce additional zones to this portal as needed.

The following zones will appear as part of the portal's **Main** tab page

- **Thread Summary.** This zone displays details of the individual threads for the batch run in context. It can be filtered by thread status.
- **Threads by Status.** This zone displays a count of the number of threads by status for the selected batch run, in the form of a pie chart.
- **Threads by Volume.** This zone displays a count of the records processed and records in error for each thread in the selected batch run, in the form of a bar chart.
- **Threads by Elapsed Time.** This zone displays the elapsed time for each thread in the selected batch run, in the form of a horizontal bar chart.
- **Threads by Records Per Second.** This zone displays the number of records processed per second for each thread in the selected batch run, in the form of a horizontal bar chart.

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 > Tools > Request** in add mode. 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.

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 > General > Request Type**.

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 > Tools > 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.

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 > System > 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.

External Reference

Attachment records may be added from an external system in which case, the external system could have a reference ID to capture. The attachment allows for including an external reference ID when adding an attachment internally.

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. To do this, you may use the generic attachment zone provided by the product: **F1-ATTCHOWN**.

Note that your product may already have support for viewing and maintaining owned attachments on one or more of its base delivered portals.

Configuring Size Limit

The system allows for configuration of a size limit for uploading attachments. This is optional but the recommendation is to set this to a reasonable value so as to limit the ability to upload unreasonably big files. To configure this, navigate to [Feature Configuration](#). Search for an existing feature configuration with the Feature Type **General System Configuration**.

If one does not exist, create a feature configuration of this type. Select the **Maximum Attachment Size** option and define an appropriate value.

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.

Supporting File Name Exceptions

By default, the system prevents attachment file names from having special characters. If there is some reason that an implementation needs to define an attachment with special characters, there is a system property setting that may be configured to relax this validation. Contact your system administrator for support.

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 chosen, 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 > Implementation Tools > 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. Note that only service programs that are Java based or Java (converted) are applicable here. There are some Services in the application generated for portals. There is no XML for those services.



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.

Groovy Java Docs Button



The **Groovy Java Docs** button switches to the Groovy 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



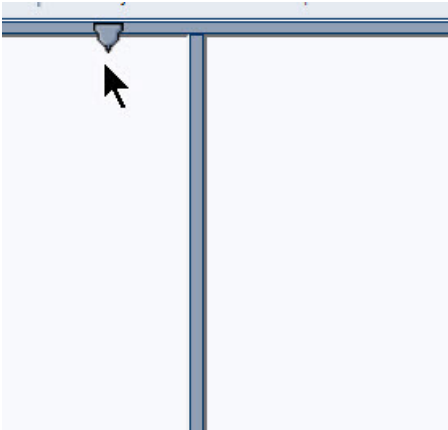
The **About** button opens a window that shows when was each Application Viewer data component recently built.

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.

Groovy Java Docs Viewer

The Groovy Java Docs viewer is an interactive tool that allows you to browse Java documentation files (Javadocs) for the Java classes that are accessible to Groovy code within scripts.

NOTE: For system protection, only a subset of the base Java classes are available for use by Groovy code.

To open the Groovy Java Docs viewer from within the application viewer, click the [Groovy Java Docs button](#). You can also access the viewer via the 'View Groovy Javadocs' link in the context sensitive Script Tips zone. Refer to the additional topics in the [Java Docs Viewer](#) section for details of how to navigate the viewer panels.

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.

Monitoring and Analytics

This chapter describes various tools provided in the product to support system monitoring including tools related to reporting, monitoring and analytics.

Reporting Tool Integration

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. Note that the product includes an out-of-the-box integration with BI Publisher. However it is possible to use the reporting objects to integrate with a different third party tool.

Integration with BI Publisher

Your DBMS, your product, and BI Publisher 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.

Multi-Language and Localization Support

The integration supports a multi-language implementation and supports different localization settings.:

- All labels, headings and messages are defined using field and message meta-data in the application, which support multiple languages.
- The appropriate font, size, and layout are based on the requested report and the user's language.
- Dates and numbers are formatted as per the user's display profile.
- Currency based numbers are formatted as per the currency definition from the product

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.

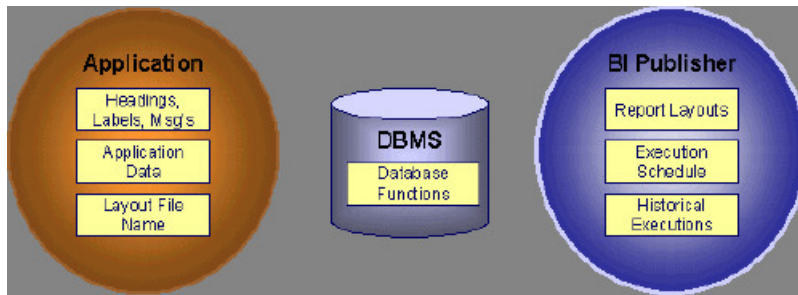


Figure 5: Application and BI Publisher

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 *Installation Guide* or *Optional Products Installation Guide* for information on this configuration.

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](#) 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

If your reporting tools supports it, the [Report History](#) page may be configured to open tool'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.

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

Reporting Folder defines the shared folder where reports are stored.

Reporting Server From Browser 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>`.

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**.

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 CM_TODO is named CM_TODO_Base.

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 your product's *Installation Guide* or the *Optional Products Installation Guide*.

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. The following fields display:

Parameter Code is the identifier of the parameter. This must correspond to the parameter definition in the reporting tool.

Required indicates that a value for the parameter must be defined when submitting the report.

Sort 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 indicates the characteristic type used to define this parameter.

Default Value is option and if populated is displayed to the user when the report is chosen on the [report submission](#) page.

Description is a brief description of the parameter. This description is used when displaying the parameter on the [report submission](#) page.

Long Description is 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

Refer to the *Installation Guide* or *Optional Products Installation Guide* for 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](#).
- Defining a user for integration with your product.
- Publishing the sample reports in BI Publisher.

Contact your system administrator to verify that the above steps have occurred.

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. Refer to the documentation for this products for details about publishing a report. Refer to [Publishing Reports in BI Publisher](#) 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.

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 Type](#) 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	CHAR_TYP	Report
CI_GLTY	GL Account Type	Pre-defined	A- Asset, E- Expense, LM- Liability/	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.

Service Health Check

The system provides a plug-in spot on [Installation](#), called Health Check. The algorithms plugged in here are used to check various system conditions and return details to help assess the health of the system.

What Information is Checked?

Each algorithm is responsible for checking one type of health check condition. The condition should be defined as a valid health component type (as defined in the **HEALTH_COMP_FLG** lookup). The algorithm may check conditions across many records (referred to as Health Components) and return a status and details for each health component. For example, if the algorithm is checking the level of service for batch programs in the system, the batch control is the health component in this case. The algorithm should gather the level of service results for each batch program and return the collection of information to the calling program.

The algorithms for the plug-in spot should return the following information for each health component.

- The **Health Component Detail** should be populated with information specific to the object that was checked. For example, if the algorithm is checking batch control level of service, this will have the Batch Control code. In addition, for enabling display for details of each health component type, the algorithm should populate the maintenance object code and primary key field and value details along with the navigation option.
- The **Health Component Status Flag** and **Health Component Status Description** should be populated with a valid status code and description appropriate for the condition being checked. This information could differ based on the health component type as each health component type may have its own unique status values.
- The **Health Component Status Reason** is meant to provide supporting information about the health component's status value. For example, if the component provides an error type of status, this could have the status reason.

- The **Health Component Response** is used to map the status value of the health component to a standard value defined in the lookup **HEALTH_RESPONSE_FLG**. The values are a subset of HTTP response codes. The supported values are **All Checks Successful** (200); **Non-Critical Function Degraded** (203), **No Content** (204) and **One or More Critical Functions Degraded** (500).
- The **Response Details** list is provided for health components that check several conditions as part of the health check. Each condition could return its own status. In this situation, the algorithm should use the response details list to record the individual responses and populate the overall Health Component Status Flag, Description and Reason with summary information. The suggestion is to do the following:
 - Set the status flag and response based on the details of the individual responses. Refer to the batch level of service information below for an example of this logic.
 - When the response list is populated, the health check user interface will show the overall Status Reason text along with an icon to expand the details. The suggestion is to populate this field with text like "See Results for Details".

The system provides an algorithm that checks the **Batch Level of Service** health component type. This health component type finds all the batch controls that are configured with at least one [level of service](#) algorithm and invokes the algorithms 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 based on whether the batch control has one algorithm or multiple. If there is one algorithm, the Level of Service lookup value returned by the algorithm is returned. If there are multiple, the system determines an overall status based on the detailed status values from each algorithm. If any algorithm returns Error, that value is returned. Otherwise, if any return a Warning, that value is returned. Otherwise Normal is returned.
- The **Status Reason** is populated based on whether the batch control has one algorithm or multiple. If there is one algorithm, the expanded text of the status reason returned by the algorithm is returned. If there are multiple, the text from message category / message number 11002 / 22001 is returned (**See Results for Details**)
- The **Response** is populated based on the value of the overall 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**.

How are the Algorithms Called?

These algorithms are called by a business service provided by the product **F1-HealthCheck**.

That service calculates an overall Health Response value based on all the details returned by all the algorithms. The values described above for the **HEALTH_RESPONSE_FLG** are the ones used.

Also note that the system provides [Inbound Web Services](#) for this business service for both SOAP and REST service calls allowing external systems to use a web service to retrieve this information.

Health Check Configuration

To enable your system to display health check information, navigate to the [Installation Options - Algorithms](#) collection and add an entry for the **Health Check** system event for each health check algorithm provided by the product that your product would like to include in your system health check assessment.

Viewing Health Check Results

The product provides a portal that allows a user to view the detailed results of the [health check](#) business 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.

The grid displays the detail of each individual component checked as part of the system health check. Refer to [Service Health Check](#) for more information about the details returned.

Introducing Health Check Conditions

If your implementation would like to check a condition for system health that is not already provided by your product, here are the steps required:

- Define a new value in the Health Component Type lookup (**HEALTH_COMP_FLG**).
- Provide a new algorithm type, plug-in script and algorithm for the Health Check system event that checks the condition that you wish to check. Refer to the [Service Health Check](#) topic for the information that each algorithm should be supplying for the condition. Note that if you follow the pattern of the system delivered Batch Control - Level of Service algorithm, your algorithm may be designed to check conditions across many records of the same type and gather all the results.
- Plug this new algorithm into the [Installation Options - Algorithms](#) collection using the **Health Check** system event.

Measuring Performance

Many implementations need the ability to track and view the performance of key system processes against a defined target level. The framework provides objects to allow an edge product or an implementation to calculate and display performance measures against a desired target for one or more use cases. The topics in this section provide information about what is supplied in the framework and guidelines for implementing a specific use case for batch processes. Your product may supply out of the box support for additional use cases. Refer to your product documentation for more information.

Understanding Performance Targets

The following are examples of use cases that would be well suited for tracking as performance targets:

- Track and view the duration of key batch processes, either individually or as a group, and how they relate to a defined target.
- Monitor used and free space on a database against critical levels.
- Check the performance of individual user interface zones against a defined performance expectation
- Compare the number of web service requests made to an application against a threshold where performance may be of concern.

Framework provides functionality to define and categorize performance targets and link them to objects such as business services, zones and portals. This supports the calculation and display of the metrics against desired results.

In addition, Framework supplies out of the box support for batch process performance targets. Individual edge applications may supply more specific functionality for other use cases, if applicable.

Ideally, users should have the ability to view these performance targets on a dashboard that groups related measures. Framework provides the necessary components to achieve this for batch process performance targets.

The following sections highlight functionality supported for performance targets in the framework. Refer to the edge application product documentation for more details of other supported use cases.

Performance Target Objects Overview

The setup of a performance target involves a unique combination of configuration data and processing logic that calculates and displays a specific measure.

The framework performance target functionality is supported by a combination of inter-related objects, as shown below. Some of these objects will be generic for use in all performance targets while some are specific to a functional area such as batch processes.

- **Maintenance Objects** for capturing performance target types and performance target instances.
- **Extendable Lookups** to define performance target categories and performance target metrics.
- **Business Services** to calculate known metrics for a group of performance targets, such as for batch processes.
- **UI Maps** to interpret the performance calculation results and display them in charts that show the comparison to the target.
- A **Zone** that serves as a template for system duplication to create specific zones related to each performance target. The zone invokes a business service to perform calculations and display performance measures in the related UI map.
- **Business Objects** to capture configuration data for a specific performance target instance and its related objects.
- Functionality to create **Zone instances** for specific performance targets based on the associated template zone.

Each edge application will deliver the following to compliment the objects delivered by the framework:

- **Portals** to group related performance target zones.
- Specific entries for the performance target category **Extendable Lookup** .

The following sections describe the combined use of these objects for performance targets in more detail.

Calculating and Displaying Performance Targets

Performance targets are intended to be displayed in a portal using an explicit object zone. The zone parameters define both the business service used to calculate the performance metrics and the UI map that displays the results. While an individual performance target needs to reference a zone with a unique configuration that calculates a particular metric, those zones will be based on a template zone which defines the core parameters.

Framework provides a base Batch Performance Target Metric Template zone (**F1-PERFBA**) for batch process performance targets. Refer to this zone and its parameters for more information about the batch performance zone configuration and the related business service and UI map.

Performance Target Metrics and Metric Types

The framework supports two types of metrics for performance monitoring:

- **Value based metrics** are used to record results against a specific numeric target.
- **Time based metrics** are used to track the results against a specific date and time target.

The list of valid metrics for a given performance target category and its associated performance target types is maintained using an extendable lookup. Framework uses the base business object Batch Performance Target Metric (**F1-BatchPerfTargetMetric**) to define batch process metric values. Refer to this lookup for the supported batch process metrics.

NOTE: While users are not prevented from adding new values to the lookup, the list is not intended to be extendable as new values will not be recognized by the business service that performs the base batch performance calculation logic.

Your edge product or implementation may supply other extendable lookup business objects for additional performance measurement use cases, if applicable.

Performance Target Categories and Types

There are key configuration details required by all performance targets. These are defined on two related objects.

Performance Target Categories

Target categories define the template zone and security setting for a group of performance measures. The list of valid categories is maintained using an extendable lookup.

The framework product supplies the business object Performance Target Category (**F1-PerformanceTargetCategory**) for this functionality. Refer to the business object description and configuration for more information.

Performance Target Types

Target types define the related performance target business object and the display portal for a group of performance measures. In addition, a target type references a target category which defines the associated zone details.

The framework product supplies the business object Performance Target Type (**F1-PerformanceTargetType**) for this functionality. Refer to the business object description and configuration for more information.

The framework does not deliver any standard type or category values for batch processing performance targets. Refer to your specific edge application products to verify if any standard values are delivered for the batch processes within your applications. Edge applications may also supply standard categories and types for additional performance target use cases.

Performance Targets Define Specific Metrics

Although the types of measures and the business services and UI maps that govern how they are calculated and displayed are defined using separate objects, a **performance target** record defines the additional configuration needed to measure a specific metric and compare the result to a desired value.

There are key configuration details required by all performance target instances. These include a reference to the metric being measured, the desired target value or time, the desired result for the target and the unique zone by which this performance target will be monitored.

NOTE: The performance target maintenance object has a direct foreign key link to the extendable lookup business object and value that define the performance metric. This is an unusual pattern as extendable lookup values are normally recorded only in an object's schema. The pattern has been adopted to allow the description of the metric to be displayed in the performance target maintenance portal.

A given **performance target type** may require additional details for its calculations. For example, the framework batch process performance target defines additional details to restrict the measurement to specific batch processes that have executed within a given time frame. These details are configured on the performance target business object. Refer to the embedded help text on the business objects supplied by the framework and your edge applications for more information.

The **performance target type** also defines the business object to use when creating the resulting performance target record.

Objects Linked to a Performance Target

Some performance measures such as batch process metrics, derive the data for the calculation from objects within the system. The performance target can be linked to one or more related objects to define the specific data sources included in that target metric. For example, when creating a batch process performance target, the framework supports linking specific batch codes to the performance target to indicate the group of batch processes that are included in the measure.

The performance target business object may be configured to allow only relevant objects types to be linked to a performance target record. Refer to the base Batch Performance Target business object (**F1-BatchPerformanceTarget**) for an example.

Creating Performance Target Zones

Once a performance target and its objects have been defined, a unique zone needs to be created to display and monitor the specific target.

The framework functionality for batch performance targets implements the zone creation via the related business object lifecycle. When a performance target is added, status enter plug-ins are responsible for generating the new zone using the template zone and prefix configured on the target category (or the override values) and adding the zone to the portal configured on the target type. When a performance target is inactivated, an enter plug in is responsible for removing the zone from the portal.

NOTE: While the template zone associated with the performance target category may be overridden, the zone generation algorithm makes certain assumptions about the zone type and parameters. In particular, the logic expects to configure a zone parameter that references the performance target code as input to the business service responsible for calculating the metrics.

Refer to the base Batch Performance Target business object (**F1-BatchPerformanceTarget**) for details of the lifecycle and the enter plug-ins responsible for performance target zone creation.

Setting Up Performance Target Configuration

The following topics highlight the general configuration steps required to use performance target functionality. Your particular product or implementation may supply additional functionality to support specific use cases for performance targets. Refer to your product's documentation and the library of business objects supplied for performance target in your system for more information.

Performance Target Category Lookup

Refer to [About Performance Targets](#) for an overview of performance target functionality.

Each Performance Target Type is associated with a unique Performance Target Category. The categories are defined as an extendable lookup.

Navigate to the [Extendable Lookup](#) portal. Search for and select the **Performance Target Category** business object. Define values that are appropriate to the categories your implementation is assigning for the performance target types in use. In some cases, an edge application may already have delivered the appropriate performance target category for your use

Refer to the embedded help for more information about configuring this object.

Defining Performance Target Types

Refer to [About Performance Targets](#) for an overview of performance target functionality.

To maintain the performance target types applicable to your product or implementation, open **Admin > Reports > Performance Target Type**.

This is a standard [All-in-One portal](#).

The information captured on the performance target type depends on the business objects supported by your product or implementation. Refer to the embedded help text for more information.

Maintaining Performance Targets

This section describes the functionality supported for viewing and maintaining performance targets.

Refer to [About Performance Targets](#) for an overview of performance target functionality.

Navigate using **Main > Reporting > Performance Target**. This is an [All-in-One portal](#) and includes the standard List and display zones for a performance target.

For information about the data defined on the performance target, refer to embedded help.

Capturing Statistics

The product provides objects to allow for an edge product or an implementation to periodically calculate and capture statistics for one or more use cases. The topics in this section provide information about what is supplied in the framework and guidelines for implementing specific use cases.

Understanding Statistics

There are two objects that work together to capture the statistics related to a given business use case:

- The statistics control record which defines configuration related to capturing statistics. It also defines a category that is used to group similar types of statistics together.
- The statistics snapshot is the record where the actual calculated statistics are captured.

Statistics Control

The statistics control record defines whether or not statistics are automatically calculated and how frequently. In addition, you may control the retention policy of snapshot record. You may indicate that only the most recent snapshot is kept or all snapshot records are kept or that they are retained for a defined number of days.

The system provides a business object for statistics control (**F1-Statistics**) that is expected to be used for most use cases. Its lifecycle includes the support for periodic capturing of statistics snapshot records as well as on demand capture.

Statistics Snapshot

The statistics snapshot object is the object that is responsible for calculating and capturing the statistics details. A separate business object must be defined for each use case. The schema of the business object defines the details that are captured. In addition, it must include an algorithm as an enter plug-in on the **Complete** state that is responsible for capturing statistics.

The system provides a 'root' business object for statistics snapshot (**F1-SnapshotRoot**) which includes the lifecycle that statistics snapshot business objects should follow. No explicit statistics snapshot use case is provided by the Framework. Your specific product may supply some out of the box support for certain use cases, in which case specific statistics snapshot business objects are provided. Refer to your product specific documentation for more information.

Viewing / Reporting Statistics

The product provides the mechanism for defining statistics control and statistics snapshot records and viewing the data in basic format. If your product has provided support for specific use cases, there may also be additional portals used to display the statistics in a meaningful format. For example, the statistics snapshot may be capturing data that can be presented in graphical format. Additionally, if multiple historical statistics snapshot records are retained, a zone may be defined to graph changes over time.

Configuring Your System for Statistics

If your product provides statistics use cases that you are planning to implement, all that you need to do is configure the appropriate statistics control record and define the appropriate configuration for your business requirements. Refer to [Defining and Monitoring Statistics](#) for more information.

If your implementation has identified an additional use case where you would like to capture statistics, the following points highlight the steps needed to configure the system to support the use case.

- Determine whether an additional Statistics Category value is needed. This is captured on the statistics category record. Navigate to the [Lookup](#) page. Search for and select the **STAT_CATEGORY_FLG** field. Review the values and determine if additional values are needed.
- Define a new statistics snapshot BO. This should be a child BO of the base delivered BO (**F1-SnapshotRoot**). Its schema should define elements for the specific information that is captured by the statistics calculations. The schema should be designed in conjunction with an appropriate Enter Status algorithm for the **Complete** that calculates the statistics as appropriate for the business requirement.
- Once the snapshot business object is designed and implemented, configure the appropriate statistics control record.

Defining and Monitoring Statistics

Refer to [About Statistics](#) for an overview of the statistics functionality.

To define and maintain statistics control records and view a list of the current snapshot records, open **Admin > Reporting > Statistics Control**.

This is an [All-in-One portal](#) and includes the standard List and display zones for a statistics control.

For information about the data defined on the statistics control, refer to embedded help.

Statistics Snapshot

If there are any Statistics Snapshot records for the statistics control, the **Statistics Snapshot List** zone displays a list of the most recent records. A user may drill into any of the records to view more detail. You are brought to a maintenance portal where you may view details about the calculated statistics. The information captured will be unique to the particular use case. Refer to the embedded help for more information.

Creating Cube Views

Many implementations require the ability to extract and analyze complex sets of data in a way that is simple to visualize and supports business or organizational decision making. A common model for performing this type of analysis is a multi-dimensional array, often referred to as a data cube. Cube views allow data to be represented in a way that provides the user with multiple perspectives of the results.

Although called a "cube", a cube view can have two, three or a higher number of dimensions. Each dimension represents some attribute in the database and the cells in the data cube represent a measure or value. For example, a cell could contain a count of the number of times that attribute combination occurs in the database, or the minimum, maximum, sum or average value of data related to that attribute. Queries are performed on the cube to retrieve information.

The framework provides objects to support the definition of data cubes and a flexible means of viewing them. The topics in this section provide information about what is supplied in the framework to assist in implementing a cube view. Your product may also supply out of the box support for relevant use cases. Refer to your product documentation for more information.

Understanding Cube Viewer

The framework provides a cube viewer portal which is intended to be used for most implementations of cube views. It supports a number of common actions and data representations.

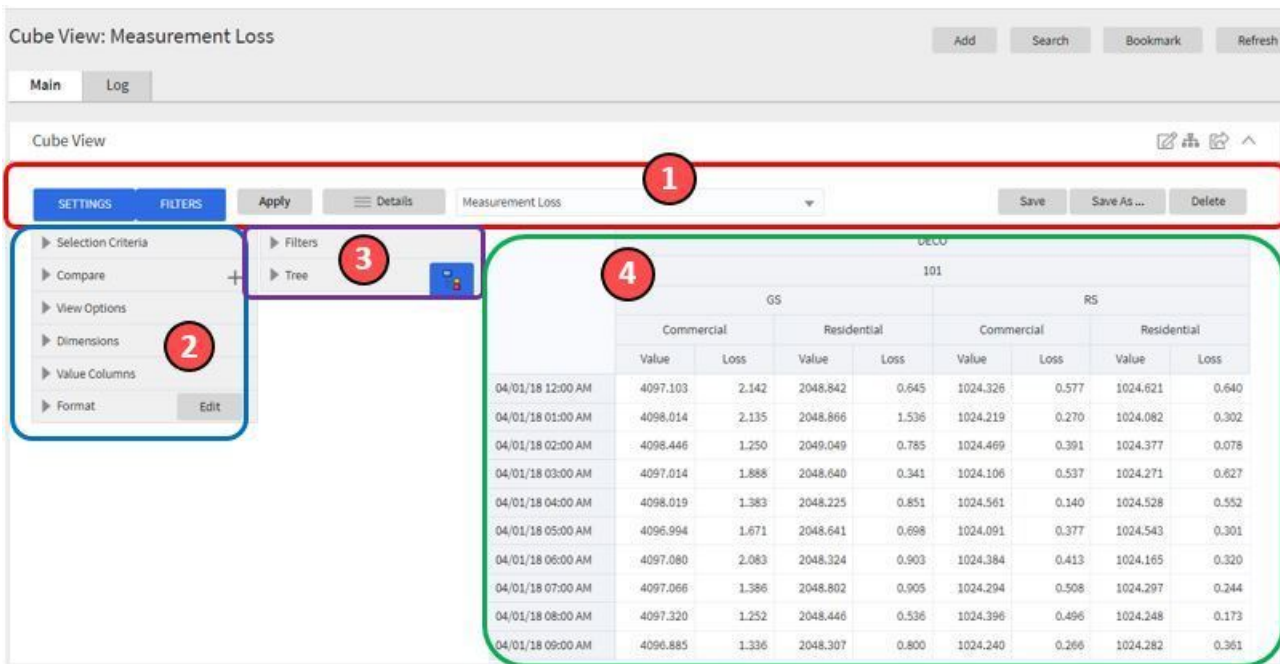
NOTE: The choice of appropriate menu for the Cube Viewer portal will differ from product to product. Refer to your product specific documentation for more information on how to access the viewer.

The following sections describe the basic concepts and features of the cube viewer.

Cube Viewer Components

The base Cube Viewer interface is comprised of the following main areas:

1. The Toolbar
2. The Settings Area
3. The Filters Area
4. The Views Area



The next topics provide more information about each component.

The Toolbar

This following features are available on the Cube Viewer toolbar.

Settings and Filters Buttons

Click the **Settings** and **Filters** toggle buttons to collapse and expand the respective [Settings](#) or [Filters](#) areas.

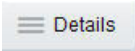
Apply Button

Click the **Apply** button to apply any changes to the settings and display the new results in the views.



Details Button

Click the **Details** button to open a dialog box in which additional attributes of the current cube view can be viewed and updated. Refer to [Cube View Details](#) for more information about the attributes displayed.



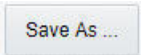
Cube View Dropdown

Click the **Cube View Dropdown** to open a list of cube views, of the same type as the current view, to which you have access. Selecting a cube view from the list will open that view. Use the search box to search for a view within the list using the cube view description.



Save, Save As and Delete Buttons

Click the **Save** button to save the current settings for the cube view. Click the **Save As** button to create a copy of the current cube view. Click the **Delete** button to delete the current cube view.


Cube View Details

Clicking the **Details** button on the cube viewer **Toolbar** allows you to maintain the following information for the cube view:

- The short **Description** for display in the viewer header and a more **Detailed Description** for capturing additional information about the cube view.
- An **Access Type** of shared, private or public.
- The **Access Group** that defines the group of users with access to the cube view. This field is only visible if the access type is shared.
- A specific **User** who has sole access to the cube view. This field is only visible if the access type is private.
- The **Row Functions** text box in which the functions to be applied to the values across the rows in the data grid are defined. Clicking anywhere in the box will display a list of supported functions to choose from.
- The **Column Functions** text box in which the functions to be applied to the values in the columns in the data grid are defined. Clicking anywhere in the box will display a list of supported functions to choose from.
- An **Inactivate** button. Click this button to deactivate this cube view.

The **Fixed Parameters** for the cube view are displayed but cannot be updated here. Refer to [Configuring Cube Types](#) for more information on cube view parameters.

The Settings Area

The Settings Area contains a number of sections whose inputs control various aspects of the current cube view.

Selection Criteria

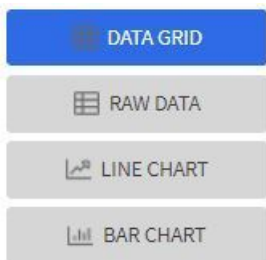
Open the **Selection Criteria** section to enter filter values for the user parameters defined on this cube type. User parameters that have been configured as 'Save with Cube View' on the cube type will be saved when the cube view is saved.

Compare

Open the **Compare** section to enter one or more sets of criteria that are used to retrieve data sets for comparison to the data retrieved by the main Selection Criteria. Use the Add and Delete buttons to add or remove a set of criteria. The comparison values will be displayed side by side with the main cube values in the data views. Note that the pseudo dimension **Dataset** must be included as a dimension column in order to display comparison views.

View Options

Open the **View Options** section to access buttons that toggle on and off the display of the supported data view types. The buttons can be used to change the order in which the views are displayed, using drag and drop. Any changes made to the default button settings will be saved when the cube view is saved. Refer to the [Views Area](#) topic for more information on the available data views.



Dimensions

Open the **Dimensions** section to view the list of dimensions defined on this cube type, as well as the pseudo dimension **Dataset** and the date/time filter, if applicable. Use drag and drop to move a dimension into the list of **Columns** or **Rows** displayed in the data views. Moving a dimension into the **Unassigned** list will remove it from the data views.

Format

Click the Edit button on the **Format** section to open a dialog box in which conditional formatting, such as background or text color, can be defined for the values in the data grid. Refer to [Defining Conditional Formatting](#) for more information about the formatting controls.

NOTE: Changes made to the settings will only be reflected in the views when the **Apply** button is pressed, with the exception of the **View Options**, where changes are reflected immediately upon toggling or rearranging the buttons.

Defining Conditional Formatting

Clicking the **Edit** button in the **Format** section of the **Settings** area allows you to define formatting rules for the display of values in the cube view data grid. The rules are based on mathematical expressions or formulas which use the cube view values as variables.

The formulas must be expressed using the field names of the values for the cube, as used in the cube view query zone. A list displaying the **Column Name** and associated **Label** for each value configured in the cube view is provided, to assist in defining the formulas.


The following attributes need to be defined for each rule:

- The **Sequence** defines the order in which the formatting rules are evaluated.
- The **Active** check box indicates whether the rule should be applied or not..
- The **Formula** defines the mathematical expression that is evaluated to determine whether the formatting should be applied. The expression variables should reference the **Column Name** of the value.
- **Format** defines the formats to be applied to cells whose value meets the condition expressed in the formula. The formats include the **Background** and **Text** colors and whether the text should be in **Bold**.
- The **Stop If True** check box indicates whether to stop evaluating the remaining formulas in sequence if this formula evaluates to true.


The Filters Area

The Filters Area provides the ability to select and apply dimension filters to the current view.

The **Tree** section displays a hierarchical view of the dimension values in the current cube view. Clicking a value will cause it to be displayed in the **Filters** section and immediately applies that filter to the data in the currently selected views. To remove a dimension from the filters, click on the value in the **Filters** section.

The **Hierarchical Filtering** icon, , is used to toggle between hierarchical and non-hierarchical selection mode. In hierarchical mode, clicking a dimension value at any level in the tree will automatically populate the filters with the related higher level dimension values. In non-hierarchical mode, only the selected dimension value is added as a filter.



Use the expand and collapse icons, , to quickly expand the tree, one level at a time, below the currently selected dimension value or collapse the entire level.

The Views Area

This section describes the types of views supported by the Cube Viewer.

Data Grid

The **Data Grid** displays the cube view data set in the form of a pivot table. The fields that appear as rows and columns in the table are defined in the **Dimensions** section of the [Settings](#) area. The default aggregation function used for the values is sum. Additional functions can be defined in the details of the cube view. Refer to the description of the **Details** pop-up window in the [Toolbar](#) area for more information.

Raw Data

The **Raw Data** view displays the cube view data set as a flat table.

Line Chart

The **Line Chart** displays cube view data in a chart with a time sequence as the X-axis and value intervals as the Y-axis. Each line corresponds to a specific combination of the selected dimensions. This view is only visible if the cube view includes at least one date/time filter in the selection criteria. If the criteria include more than one date/time field, the first date/time filter in order is chosen for the time sequence.

Bar Chart

The **Bar Chart** also displays cube view data in a chart with a time sequence and value intervals. Each bar corresponds to a specific combination of the selected dimensions. This view is only visible if the cube view includes at least one date/time filter in the selection criteria. If the criteria include more than one date/time field, the first date/time filter in order is chosen for the time sequence.

Cube Configuration Components

Cube views are supported using a number of standard configuration components.

The system provides a business object for cube views (**F1-CubeView**) which is used to capture access controls and the settings, such as user parameter values, for a cube view instance. The business object also defines the portal navigation and the base display UI Map (**F1-CubeViewDisp**) and accompanying display map service script (**F1_CubeViewD**). The display map and service script together drive the functionality supported in the cube viewer.

NOTE: The system also supplies a UI map fragment (**F1-CubeViewProc**) which is used to display a cube view in a process flow panel. Refer to [Process Flows](#) for more information.

The system provides a portal for cube views (**F1CUVWM**) that can be configured on the appropriate menu item or other navigation path for your implementation. The portal is linked to the base cube view zone (**F1-CUBEVIEW**).

In addition, the system supplies a simple query by cube view information (**F1-CUBVIWQ1**) that is the single entry on a base multi-query zone (**F1-CUBVIWQ**). The multi-query zone is not linked to the base portal as the expectation is that implementations will need to create searches tailored to their specific use cases for cube views. The base multi-query provides a quick means of configuring a cube view search while in development mode.

Configuring Cube Types

Cube Types are used to capture the details that define the sourcing of the data for a cube view and parameters that control certain features of the cube viewer. These details include:

- The business service whose associated zone defines the query used to select and filter the data for the cube view.
- The user parameters that correspond to the filters for this cube type's business service and zone.
- The user parameter settings that determine whether a given filter value should be saved with the cube view and whether the value should be populated from global or portal context, if applicable.
- The fixed parameter values to be applied as hidden filters for cube views of this type.
- The results from the business service and zone to be used as dimensions in the cube view and those that are the cube view values.
- The dimension settings that determine the order in which the dimensions are displayed in the data grid, which dimensions are to be used as filters in the cube view and which dimensions are to be included in the cube view tree.
- The values settings that determine the default sequence in which the values are displayed in the data grid.
- The FK Reference used to navigate from the data grid to the underlying source data.

If your product provides business services that support cube views you plan to use, all you need to do is configure the appropriate cube type records. Refer to [Maintaining Cube Types](#) for more information.

If your implementation has identified additional cube views you need for data analysis, you will need to create a zone and business service that use specific techniques for defining the SQL to derive the source data for the view and for mapping the results. Refer to [Cube Type Advanced Topics](#) for more information

Maintaining Cube Types

Refer to [Creating Cube Views](#) for an overview of the cube viewer functionality.

To define and maintain cube type records, open **Admin > Analytics Configuration > Cube Type**.

This is an [All-in-One portal](#) and includes the standard list and display zones for a cube type.

For information about the settings defined on the cube type, refer to [Configuring Cube Types](#) and the embedded help.

Cube Type Advanced Topics

The following topics describe the technical models that need to be followed when creating a new cube type.

Designing Your Cube Type SQL

In order to support the base cube viewer, the SQL used in your cube type data explorer zone needs to be constructed with two main components. The first component creates a view of the data and its dimensions in a form that supports a tree-like structure and allows for the dynamic use of filters. The second component queries the filtered data set to provide the values to be displayed for each dimension.

The first component should contain the following queries:

- A sub-query constructed using a 'with' clause that builds a tree-like view of the cube's dimensions and optionally other columns used in the main SQL. This portion of the SQL must be enclosed within `/*>tree*/` and `/*<tree*/` hints and use the table alias **C_TREE**. The combination of the dimension columns should form a unique 'key' in the **C_TREE** data. Both the 'with' clause and the 'tree' portion within it may contain other supporting sub-queries.
- A sub-query which is the placeholder for filtered queries on the 'tree' data. It must use the table alias **C_FILTER** and take the form `C_FILTER as (select * from C_TREE where l=1 and ROWNUM <= n)`. The value of 'n' should be set to limit the rows returned to a number that is manageable within the cube viewer.

The second component is the main query that retrieves the measures for the cube view dimensions. This query references **C_FILTER** as the subset of the **C_TREE** and joins with other tables if required to retrieve the numeric values. The first date or date/time result column of the main query (as specified in the zone configuration) designates a time-series column, which forms the time-series 'key' with the dimensions columns.

Defining Your Cube Type Business Service

The business service that identifies the zone for a cube type is also used to provide additional configuration options.

The business service schema should define meaningful element names for the parameters, dimensions and values. It should also define labels for those elements to be used by both the cube type and cube viewer user interfaces. The labels can be derived by mapping the element to a meta-data fields or by using the `'label='` syntax to define a text string directly in the schema definition. The `'required=true'` syntax can be used to require a value to be entered for a user parameter on the cube viewer or for a fixed parameter on the cube type.

The user parameter elements are expected be defined within a group in the schema called 'input' and the fixed parameters within a group called 'hidden'.

Product Use Metrics

The Product Use Metrics portal includes a zone (or zones) that are used to perform a business review of your organization's implementation of the product to confirm licensing / subscription compliance. The information visible on this portal will depend on the product or products installed by your organization.

To view this portal, open **Main > Dashboards > Product Use Metrics**.

For more information about what each zone on the portal is reporting, refer to the embedded help.

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.

Understanding Inbound Web Services

The system supports communicating with the system via RESTful services or via SOAP services. In both cases, the system uses an object called inbound web service (IWS) to store the configuration. A web service class is used to distinguish whether the IWS is used for REST or SOAP. There is also a distinct business object for each web service class value because there is different configuration for each type of IWS.

For both types of inbound web service, the system 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.

REST APIs may be secured by OAuth client credentials grant in a cloud installation. Refer to the Identity and Access Management chapter in the *Oracle Utilities Cloud Services Administration Guide* for more information.

The following topics provide more information about the different configuration of IWS based on whether it is SOAP or REST.

Inbound REST Web Services

The following topics provide more information about the product's support of REST services using IWS. Note that IWS REST records are marked with a REST engine version of 1.0 or 2.0. Version 2.0 is the version used for new services. Version 1.0 is introduced for backward compatibility. Some of the system behavior differs based on the version of the REST service. The sections below highlight cases where this occurs.

HTTP Method and Parameters

When defining operations for inbound REST web services, the product supports the HTTP methods of Get, Patch, Post, Put, and Delete. Note that the product's support of these various HTTP methods are a means of communicating the purpose of the web service to the outside world. However, the actual behavior of the REST web service is driven by the behavior of the underlying schema based object (business object, business service or service script). For example, you may configure the HTTP Method of "Put" for an operation that references a service script that just retrieves a record. The product is not able to detect this type of discrepancy. Configuration users should carefully consider the correct method to use based on the logic of that service.

For operations that reference a business object, the transaction type must be provided. The REST syntax doesn't support defining the transaction type at runtime. There are some basic validation checks in this case related to the transaction type and the HTTP method. For example, a Get method only makes sense with the Read transaction type.

NOTE: Using the transaction type **Change** requires all values to be passed in. Using the transaction type **Update** allows the web service to pass only the primary key and the values to be updated. All other elements will retain their existing values.

You may additionally define parameters. For each parameter, you define an external reference to the element, which is how this parameter is exposed to the external callers and is defined in the API specification. Each of these parameters is mapped to the XPath of the schema element from the underlying business object, business service or service script. For each parameter, you indicate if it is a **Path** parameter or a **Query** parameter.

- Path parameters are parameters that are part of the endpoint and are required. Each path parameter must be included in the operation's URI component surrounded by curly brackets.
- Query parameters are optional. They are not part of the endpoint but rather are included in the endpoint URL after a question mark, followed by name value pairs.

Refer to the URL section below for examples of path and query parameters in the sample URLs.

URL Composition

When building the endpoint URL for a REST service, there are three main parts that make up the full URL.

- The first part is the one that is environment specific. This will differ for an on premise implementation as compared to a cloud implementation. Both will have the host and the port and then additional components that identify the environment.
- The second part is a hard coded path designated by the product, namely "/rest/apis".

These two parts of the URL are defined in the substitution variable **FI_REST_BASE_URL**. This is defined when initializing your environment. Refer to the Server Administration Guide for more information.

The remaining part of the URL is built dynamically based on configuration for each IWS and its operations. The components are "/ownerURIComponent/resourceCategoryURIComponent/iwsURIComponent/operationURIComponent"

- The owner URI component is taken from the owner flag of the inbound REST web service. A special extendable lookup ([Owner Configuration for REST Services](#)) defines this component for each owner flag.
- Each Inbound REST Web Service must reference a [resource category](#). This category is used to associate related web services to a common category of resources. For multiple IWS records linked to the same resource category, external catalogues can use this information to group together related web services. The resource category is an extendable lookup and the URI component is an attribute of this record.
- Each REST IWS record defines a URI component that serves as an external identifier of this IWS record. The value must be unique within a given owner flag.
- Each operation must define the HTTP Method and optionally a URI component. When defining path parameters, the path parameters must be included in the URI component using curly braces.

In all cases, the URI component must begin with a slash (/).

The following are some examples of the dynamic portion of the URL for a REST service. The last example illustrates the use of a query parameter.

IWS Name URI Component	Owner URI Component	Resource Category URI Component	Operation HTTP Method URI Component	Dynamic URL Component	Run Time Example
c1rateCalculation /rateCalculation	C1 /customer	C1-RATES /rates	Post /	../customer/rates/ rateCalculation/	../customer/rates/ rateCalculation/
w1WorkActivity /workActivity	W1 /asset	W1-WORK /work	Get /{activityId}	../asset/work/ workActivity/ {activityId}	../asset/work/ workActivity/5798165498
w1WorkActivity /workActivity	W1 /asset	W1-WORK /work	Patch /scheduleWindow/ {externalSystem}/ {activityId}/ {windowStartDateTime}	../asset/work/ workActivity/ scheduleWindow/ {externalSystem}/ {activityId}/ {windowStartDateTime}	../asset/work/ workActivity/ scheduleWindow/ MY- COMPANY/5798165498/20190101
CM- AccountActivityHistory / accountActivityHistory	CM /cm	CM-ACCT-INFO /accountInformation	Get /{accountId}	../cm/ accountInformation/ accountActivityHistory/ {accountId}	../cm/ accountInformation/ accountActivityHistory/123456789? activityId=5468976

Payload Format

REST services support receiving a request payload in XML or JSON format and returns the payload in either XML or JSON. The default format returned depends on the value of the REST Engine Version.

- Version 2.0 services assume JSON format as the default. The default can be overridden by providing an accept header of **application/XML**.
- Version 1.0 services assume XML format as the default. The default can be overridden by providing an accept header of **application/JSON**.

Root Node for JSON Format

Version 2.0 services with the JSON formatted payload do not accept in the request or return in the response any root node around the payload. The following is an example of the response of a REST call for a version 2.0 service:

```
{
  "batchJobId": "string",
  "requestSuccessful": "string"
}
```

Version 1.0 services with the JSON formatted payload expect a root node in the request and return one in the response. The following is an example of the response of a REST call for a version 1.0 service:

```
{
  "F1CnclBatJob": {
    "batchJobId": "string",
    "requestSuccessful": "string"
  }
}
```

The Open API Specification visible on the inbound web service maintenance page displays the expected format based on the record's REST engine version when viewing the specification.

Data Type Format in JSON

In JSON format, strings are surrounded by quotes and numbers and Boolean data do not have quotes. All services for REST engine version 2.0 follow this standard. Originally services that are REST engine version 1.0 were incorrectly treating numbers and Boolean data as strings and returning the data with quotes. This has been corrected.

To accommodate any integration that may have worked around the behavior for version 1.0 services, the system provides the ability to identify inbound web services that are exceptions to this rule. For any inbound web service identified as an exception, the system continues to treat numbers and Boolean data as strings in JSON responses. To add one or more IWS records to the exception list:

- Go to [Feature Configuration](#).
- Look for an existing feature configuration with the feature type **External Messages**. If one does not exist, create one.
- Add an option for the option type **IWS JSON Data Type Exceptions**.
- In the option value, indicate the IWS record that is an exception. Note that multiple options for the option type may be added. Additionally, the option value supports a comma delimited list.

External Facing Schema

By default the schema of the underlying service (for example, service script, business object or business service) is also the schema for the REST inbound web service operation and acts as both its Request and the Response schema. The product provides the ability to define an explicit schema for a REST inbound web service (IWS) operation, allowing a user to adjust the schema for an external facing consumer. The IWS operation schema also supports some special configuration that allows additional features to be defined only for the IWS operation schema.

The operation schema supports the following capabilities:

- Declaring whether an element belongs only to the request schema, only to the response schema, both (default) or excluded altogether. This is used to establish a clear separation between request and response schema definitions.
- Assign a different element name to an internal element. This allows the internal service element name to more closely align to internal references to the element, if needed. The external facing element name can be different allowing for a more readable schema.
- Introduce specialized elements that support links and other types of structural standards in a way that does not impact internal service processing.

Refer to [Web Service Schema Nodes and Attributes](#) for more information.

The REST servlet uses the request and response schemas to map elements to and from the internal schema.

NOTE: While the caller should only provide elements defined by the request schema, the application does not filter out extraneous elements. Assumption is that the internal service is designed to ignore such elements. However, only elements defined by the response schema are included in the response.

Dynamic Links

There are use cases where the published API includes a "_self" element that contain the endpoint URL of the GET operation related to the data returned in the response. Additionally, response payloads may include foreign keys and for those entities, the response includes a "_link" element that contains the endpoint URL of the GET operation for that entity (if it exists).

Syntax is provided in the REST inbound web service operation schema to support building the runtime endpoint URL for the _self and _link elements. Besides dynamically building the static portion of the URL based on the current environment details, it also builds the dynamic portion of the URL, substituting the URL components for the operation and substituting the path parameters. The syntax allows you to define a specific IWS Operation or allows you to reference a maintenance object and at runtime, the system determines the GET operation for the maintenance object and builds the URL accordingly.

Example of the syntax:

```
<_link getOperation="mo:'TO DO ENTRY';pk1:todoEntryId;" />
```

Example of the runtime endpoint URL:

```
_link: "http://.../common/toDos/todoEntries/28937296450934"
```

A GET operation may be associated with a maintenance object to denote that this is the default GET operation for the entity. A **GET Operation** option at the Maintenance Object and Business Object level may be used to override this default. If the entity is not associated with a business object or the latter is not associated with such option then the operation associated with the maintenance object (if any) is used.

Open API Specification Documentation

The following documentation information may be provided for an Inbound Web Service:

- Short and detailed description for each operation.
- Help text may be provided for individual elements. The same text can be shared across multiple elements across all operations of the Inbound Web Service. If an element is not associated with help text then its internal field's label is used instead.
- Sample request and response documents may be provided for each operation.
- Operations may be associated with a sequence number that controls the order in which they appear in the Open API Specification.

NOTE: Detailed descriptions and element related help text are not translated since the published catalogue for base product APIs is in English language only.

Inbound SOAP Web Services

For inbound SOAP web services, by default the system 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 XSLs to transform into the relevant schema formats. In addition, if desired, the Request and Response Schemas that document the expected message may be provided.

NOTE: The Request and Response Schema fields are not supported in a cloud implementation.

NOTE: The system supports defining the XSLs as a [managed content](#) record. For backward compatibility, the system supports defining the XSL as a file in the file system. This is a system wide setting, defined using a feature configuration option. The feature type is **External Messages** and the option type is **XSL Location**. Set the value to **F1FL** to support XSL in the file system. The file location is configured using a system property. To support the XSL in managed content, no option is needed as this is the default. You may explicitly define the value of **F1MC** if desired.

In addition, note that for business object based operations, when invoking the web service an action is required. This may be passed into the web service as part of the invocation or alternatively, the action may be defined when configuring the operation using the transaction type.

NOTE: Using the transaction type **Change** requires all values to be passed in. Using the transaction type **Update** allows the web service to pass only the primary key and the values to be updated. All other elements will retain their existing values.

Annotations Used for Security

When preparing to deploy inbound SOAP web services, the security aspects of the service must be decided. The product provides a default security policy that is applied when no other policy is defined: **@Policy(uri="policy:Wssp1.2-2007-Https-BasicAuth.xml", attachToWsdL=true)** which requires HTTP Basic over SSL and a WS-Security Timestamp.

If a different security policy is desired, the following options are available:

- Security policies may be attached to the Inbound Web Service via the Java Enterprise Edition (Java EE) Web Application Server. This allows for multiple policies to be attached as supported by the Java EE Web Application Server. In order to enable this capability, explicit system configuration is required so that the product does not assume the default security policy. See the subsequent bullets for more information.
- Define a system wide security policy using a feature configuration option. Find the [Feature Configuration](#) record for the **External Messages** feature type. (It may need to be defined if it does not exist). Choose the option type **Default security policy** and define an appropriate value. If your implementation wishes for the policies to be attached at the Java EE Web Application Server, define this option type with an option value of **<none>**.
- Attach a security policy to the IWS via a Web Service Annotation. The base product provides annotation types that support the standard WS-Policy (**F1POLICY**) and OWSM Security Policy (**F1-OWSM**). No base annotation is supplied by the product for either annotation type.

If your implementation wishes for the policy of a particular IWS to be attached at the Java EE Web Application Server, define a special annotation for the **F1POLICY** annotation type and configure the **uri** parameter value to **<none>**.

NOTE: Refer to WebLogic documentation for more information on supported security policies.

NOTE: In order to use the OWSM Policy, additional system configuration is necessary. Contact your system administrator to confirm if your implementation supports OWSM.

Inbound SOAP Web Service Deployment

A Inbound SOAP Web Service must be deployed to the Java EE Web Application Server in order for it to be available to the Web Service Clients to access the system. Refer to [Deploying SOAP 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 Java EE 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 Java EE 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 SOAP 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.

NOTE: There is no support for XAI inbound services via REST, only via inbound SOAP web services.

Configuring Inbound Web Service Options

This topic describes the configuration needed for using inbound web services.

Configuring SOAP Inbound Web Service Options

This topics in this section describes the configuration needed for using inbound SOAP 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 some base annotation types. Refer to the metadata for more information. 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 Java EE Web Application Server are not sufficient for your implementation's needs.

CAUTION: Important! When adding new records, carefully consider the naming convention of the web service annotation type code. Refer to [System Data Naming Convention](#) for more information.

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.

The system supports the ability for an IWS record to refer to multiple policies. In this situation, the annotation type for the policy should include a reference to a parent annotation type so that the system can properly build the array of annotations.

Maintaining Web Service Annotations

If your implementation wishes to define annotations, use the Web Service Annotation portal. Open this page using **Admin > Integration > Web Service Annotation**.

This is a standard [All-in-One portal](#).

CAUTION: Important! When adding new records, carefully consider the naming convention of the web service annotation code. Refer to [System Data Naming Convention](#) for more information.

Maintaining XAI Inbound Services

XAI Inbound Service is a legacy mechanism for exposing system services to external systems as web services. The current supported mechanism for defining web services is [Inbound Web Services](#), which should be used for any new web service.

Some framework based products and existing implementations may have existing XAI inbound services that may need to be viewed. For these services, the product supports deploying them as SOAP inbound web services for execution. Refer to [Deploying XAI Inbound Service via IWS](#).

The following sections describe basic information about the maintenance pages. Note that some of the information is related to legacy functionality that is no longer supported.

XAI Inbound Service - Main

To view an inbound service, open **Admin > XAI > XAI Inbound Service**.

Description of Page

XAI In Service Name 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 **XAI Service ID** serves as the primary key.

Owner indicates if this XAI inbound service is owned by the base package or by your implementation (**Customer Modification**). This information is display-only.

The **Adapter** defines the interface with the target application server.

If adapter for this service invokes a system service, then the appropriate **Service Name** is visible.

If adapter is the **Business Adapter** then **Schema Type** and **Schema Name** reference the object to invoke.

Web Service Category is visible if the XAI inbound service is linked to one or more [web service categories](#).

Use the **Description** and **Long Description** to describe the service.

Check the **Active** switch if this service is enabled and available for execution.

Post Error is not applicable.

Trace is not applicable.

Debug is not applicable.

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 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.

XAI Inbound Service - Staging

The configuration on the staging tab is no longer supported.

XAI Inbound Service - Parameters

The configuration on the parameters tab is no longer supported.

Configuring REST Inbound Web Service Options

This topics in this section describes the configuration needed for using inbound REST web services.

Owner Configuration Lookup

Refer to [Inbound REST Web Service](#) for an overview of REST IWS functionality.

When generating the URL for a REST web service, part of the URL is taken from the owner of the Inbound REST Web Service record. The component to add to the URL is defined in this lookup using the URI Component field. The values are provided by the product.

To view the values, navigate to the [Extendable Lookup](#) portal. Search for and select the **Owner Configuration for REST Services** business object.

Resource Category Lookup

Refer to [Inbound REST Web Service](#) for an overview of REST IWS functionality.

Each Inbound REST Web Service is associated with a Resource Category. The categories are defined as an extendable lookup. The REST URL is built using a component from the IWS record's resource category.

Navigate to the [Extendable Lookup](#) portal. Search for and select the **Resource Category** business object. Define values as needed. Typically the category is related to a logical grouping of entities, or maintenance objects, referred to as a 'resource' in REST nomenclature. In some cases, an edge application may have already delivered appropriate resource categories for your use.

Populate the appropriate URI component to include when building the URL for IWS records linked to this resource category.

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 support for REST and SOAP services and provides several inbound web services out of the box.

For inbound SOAP web services, 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 > Integration > 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.

CAUTION: Important! When adding new records, carefully consider the naming convention of the inbound web service code. Refer to [System Data Naming Convention](#) for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Inbound Web Service.** This zone displays the configuration information for the record and its operations.
- **Operations.** This zone lists the operations defined for the selected inbound web service. The zone is only visible for a REST inbound web service. You may navigate to the Inbound Web Service Operation portal to further describe the operation's web service schema.
- **Help Text Fields.** This zone maintains help text information for fields across all operations of the selected inbound web service. The zone is only visible for a REST inbound web service.

Refer to [Inbound SOAP Web Services](#) for specific information about Operations for SOAP records.

Refer to [Inbound REST Web Services](#) for specific information about Operations for REST records.

Note that in addition to standard actions available on this portal, there is also a special **Add to Category** button in the page action area. Click this button to link an inbound web service to one or more [web service categories](#).

Maintaining Inbound Web Service Operations

This portal is used to maintain a REST inbound web service operation. You can access the portal from the **Operations** zone on the Inbound Web Service portal for a REST service.

Refer to [Inbound REST Web Services](#) for specific information about Operations for REST records.

The following zones may appear as part of the portal's **Main** tab page

- **IWS Operation.** This zone provides information about the selected operation.
- **Schema.** This zone describes the request and response schema as defined by the operation's web service schema. The zone is only visible when an operation schema is explicitly defined.
- **Schema Extensions.** This zone manages web service schema extensions that correspond to extensions made to the operation's internal schema. This zone is only visible if the underlying schema was extended using dynamic data area extension functionality. Each owner that extended the internal schema may provide web service schema information for their extension elements.
- **Options.** This zone lists various options for the operation such as request and response samples. The zone is visible when such options exist for the operation. Note that options are maintained as part of operation record.

The following zones may appear as part of the portal's **References** tab page

- **IWS Operation References.** This zone lists references to the current web service operation. The list includes other inbound web service operation records that reference the current operation via their web service schema. For a GET operation, the list also includes the maintenance object or business object that explicitly references the operation using a **GET Operation** option.

Web Service Schema Nodes and Attributes

A REST inbound web service (IWS) operation invokes a single schema based object, either a Business Object, a Business Service or a Service Script. The object has one schema which by default acts as both the Request and the Response schema. The product provides the ability to define an explicit schema for a web service operation that represents a view of the underlying internal schema for an external facing consumer. Note that the operation schema is a strict view of the internal schema and as such does not introduce structural changes.

While optional, defining an operation schema has the following benefits:

- Declaring the **usage** of an element as to whether it is part of the Request schema, Response schema, both or neither.
- Assigning an **external element name** to an internal element. Useful to avoid cloning of internal services for this purpose only.
- Better management and visibility of **GET operation** references.
- Support HATEOAS style structures outside of the internal schema. This allows for seamless internal processing.
- Providing help text for individual elements.

The following documentation is a complete list of the attributes available to you when constructing an IWS operation schema.

Contents

- [Operation Schema Definition](#)
- [Specialized Elements](#)
- [Open API Specification](#)

Operation Schema Definition

The following attributes defines the external facing operation schema that establishes the request and response schema for the operation.

Mnemonic	Valid Values	Description	Examples
usage=	"REQ"	Indicates that the element should be included in the request schema only. When specified for a group or list element the value applies to the entire container. A specific element in the container may override the container default.	<code><customerName usage="REQ" /></code>
	"RESP"	Indicates that the element should be included in the response schema only. When specified for a group or list element the value applies to the entire container. A specific	<code><customerId usage="RESP" /></code>

Mnemonic	Valid Values	Description	Examples
	"BOTH"	<p>element in the container may override the container default.</p> <p>Indicates that the element should be included in both request and response schema definitions. This is the default value</p> <p>When specified for a group or list element the value applies to the entire container. A specific element in the container may override the container default.</p>	<pre><customerName usage="BOTH" /></pre>
	"EXCL"	<p>Indicates that the element should not be included in either the request or the response schema definitions.</p> <p>When specified for a group or list element the value applies to the entire container. A specific element in the container may override the container default.</p>	<pre><httpMethod usage="EXCL" /></pre>
mapTo=	"internal element name"	<p>Maps a web service element to an internal element. By default the internal element name is assumed to be the same as the web service element name. Use this attribute to assign an external facing element name in the operation schema and map it to a corresponding internal element.</p> <p>Note that this is not an XPath expression but rather a reference to a valid element name that corresponds to the structure of the expanded internal schema.</p> <p>The owner of the element must be the same as the owner of the web service schema. If an operation's underlying schema was extended (using dynamic data area extension) then the extending owner may only reference their elements on their web service schema extension for the operation.</p>	<p>Internal schema element:</p> <pre><todoPriority .../></pre> <p>Operation schema element:</p> <pre><priority mapTo="todoPriority"/></pre>
role=	"FKGP"	<p>Establishes a group structure that represents a foreign key to an entity.</p> <p>The element itself does not exist in the internal schema. It</p>	<p>Internal schema element:</p> <pre><user .../></pre> <p>Operation schema group:</p> <pre><user role="FKGP"> <user></pre>

Mnemonic	Valid Values	Description	Examples
		<p>groups together the foreign key value element(s) along with a corresponding _link element that at run time provides the endpoint URL of that entity's GET operation. Consumers of the web service may use the link to get the data for the entity.</p>	<pre><_link getOperation="mo:'USER';pk1:us > </user></pre>
	<p>"COLL"</p>	<p>Establishes a group structure that represents a collection of entities that corresponds to an internal service list.</p> <p>The element itself does not exist in the internal schema. It groups together the internal list mapped to a _data element along with a corresponding _link element that at run time provides the endpoint URL for the collection's GET operation. Consumers of the web service may use the link to get a more detailed set of information for each entity in the list.</p> <p>By default all the elements in the referenced list are included in the operation schema. You may explicitly reference and define attributes for specific elements as needed.</p>	<p>Internal schema element:</p> <pre><alerts type="list"> <alert/> <startDate /> <endDate /> <version /> </_data> </alerts></pre> <p>Operation schema group:</p> <pre><alerts role="COLL"> <_link getOperation="... " /> <_data mapTo="alerts"> <version usage="EXCL" /> </_data> </alerts></pre>
<p>getOperation=</p>	<p>"expression"</p> <p>Expression components:</p> <ul style="list-style-type: none"> • mo:'<maintenance object>'; • pk1:<relative xpath to an element in the internal schema>; • pk2:...; • pk3:...; • pk4:...; • pk5:...; <p>Parameters pk2–5 are required to match the maintenance object's prime key definition.</p>	<p>Only valid on _self and _link elements. The attribute is used to reference the default GET operation for an entity as defined by its maintenance object or business object.</p> <p>The application determines the GET operation to use as follows:</p> <ul style="list-style-type: none"> • If the entity's business object references an operation using the GET Operation option then it is used. • If the entity's maintenance object references an operation using the GET Operation option then it is used. • If a GET operation is found that references the maintenance object then it is used. 	<pre><_link getOperation="mo:'TO DO ENTRY';pk ></pre>

Mnemonic	Valid Values	Description	Examples
		If a GET operation cannot be determined for the entity a standard text indicating that a link is not available is returned instead.	
	<p>"expression"</p> <p>Expression components:</p> <ul style="list-style-type: none"> iws: '<inbound web service name>'; operation: '<operation name>'; parms: [<external reference as defined on the GET operation record>:<relative xpath of an element name in the internal schema that holds the value>; <another external reference name>:<another relative xpath>;...]; 	<p>Only valid on <code>_self</code> and <code>_link</code> elements. The attribute is used to reference a specific GET operation along with the information needed to compose its path parameters.</p>	<pre><_self getOperation="iws:'Fl- ToDoEntry'; operation:'getToDoEntry'; pa [ToDoEntryId:ToDoEntryId;]"/ ></pre>

Specialized Elements

The operation schema definition makes use of specialized element names in the style of the Hypermedia as the Engine of Application State (HATEOAS) standard. These elements should only be defined by the operation schema as they are not relevant to nor should be managed by the internal operation service.

The following list describes each element and how it is used.

Element Name	Description	Examples
<code><_self .../></code>	<p>Response payloads may include a "<code>_self</code>" element that contains the endpoint URL of the GET operation related to the entity returned in the response.</p> <p>This is an optional element and only one may exist in an operation schema.</p> <p>Refer to getOperation= schema definition attributes for more information.</p>	<pre><_self getOperation="mo:'TO DO ENTRY';pk1:ToDoEntry "></pre>
<code><_link .../></code>	<p>Response payloads may include foreign keys and for those entities, the response includes a "<code>_link</code>" element that contains the endpoint URL of the GET operation for that entity (if it exists).</p> <p>Refer to role= and getOperation= schema definition attribute for more information.</p>	<pre><_link getOperation="mo:'TO DO ENTRY';pk1:ToDoEntry "></pre>
<code><_data .../></code>	<p>Provides a standard reference to a list of entities in the internal schema.</p>	<pre><alerts role="COLL"> <_link getOperation="... "/> <_data mapTo="alerts"> </_data> </alerts></pre>

Element Name	Description	Examples
	Refer to role="COLL" schema definition attribute for more information.	

Documentation Attributes

The following attributes provide additional documentation for individual elements that is included as part of the web service's Open API specification.

Mnemonic	Valid Values	Description	Examples
helpText=	"help text field name"	<p>Associates an element with a specific help text record defined for the Inbound Web Service. If the element is included in both request and response schema the same help text is used in both.</p> <p>If this attribute is not specified, and a single Help Text Field record exists for the Field associated with the internal element, that help text record is used. If more than one such Help Text Field record exists the one where the help text name is the same as the Field name is used.</p> <p>Note that an internal element is associated with a Field name via the mdField= attributes, if any, else via the mapField= attribute.</p> <p>No valid for <code>_self</code> and <code>_link</code> elements.</p>	<pre><ToDoEntryId usage="BOTH" helpText="TD_ENTRY"></pre> <p>In this example the inbound web service defines a help text field name "TD_ENTRY" that provides help text for this element. The same help text is included for the element in both request and response schema.</p>
helpTextResponse=	"help text field name"	<p>Associates an element that is included in both request and response schema with a specific help text record to be used in the context of the response schema.</p> <p>Follows same rules as the helpText= attribute.</p>	<pre><ToDoEntryId usage="BOTH" helpText="TD_ENTRY"></pre> <p>In this example the inbound web service defines the following help text records:</p> <ul style="list-style-type: none"> • Help text field name "TD_ENTRY" describes the element sent in the request schema. • Help text field name "TD_ENTRY_RESP" describes the element populated in the response schema.

Deploying Inbound SOAP Web Services

Once an Inbound SOAP 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 SOAP Web Services to the Java EE 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 Java EE 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 Java EE Web Application Server. This can also be done manually for clustered deployments, if desired.

There are two methods available for deploying inbound SOAP web services:

- Deployment at the command line using the **iwdeploy[.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 only supported in development (non-production) environments.

Inbound Web Service Deployment Portal

To use the online Inbound Web Service Deployment portal, navigate using **Admin > Integration > Inbound Web Service Deployment**.

The following sections describe the base zones that are provided on the portal.

Deploy Inbound SOAP Web Services

The Deploy Inbound SOAP Web Services zone provides information about the last deployment. If the region is a development (non-production) region you may 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 SOAP 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 Deployment**.

- 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 SOAP 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 Deployment.** 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 Undeployment.** Indicates that the IWS is marked as inactive but the IWS is found to be deployed at this time.

If the IWS has been deployed, the View column will include a **WSDL** link allowing you to launch a separate window to view the WSDL definition.

NOTE: For cloud installations, the system supports defining the WSDL URI via a substitution value defined for the token URI: @F1_BASE_IWS_URI@. The system will use this value per the functionality described in [URI Substitution](#). When no value is defined, the system uses a default URL, which is the appropriate URL for on-premise installations.

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 Deployment.** 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 Undeployment.** 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 Deployment" in cases where a redeployment may not be necessary. The recommendation when this occurs is to simply Deploy again to be safe.

If the IWS has been deployed, the View column will include a **WSDL** link allowing you to launch a separate window to view the WSDL definition.

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.

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 collecting related messages into a batch 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 do the following:

- 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.

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: OUAF Outbound Message that may be used by an implementation to define messages to process and how to process them. This transport processes outbound messages in order of the priority defined on the outbound message type.

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 technical reference paper *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 system supports special functionality for sending an Email message real-time. Refer to [Sending Email](#) for more information.

For other types of real-time messages, the system also uses outbound message type and external system configuration to format and route the message. When defining the configuration for real time messages, an additional step is required to define the mechanism for routing the message using a message sender. The system supports routing messages via HTTP and via JMS. Note that for HTTP routing, the system also supports sending the message using a JSON format.

Just like near real-time messages, initiating a real-time outbound message may also be done 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. Note that the outbound message BO should be configured to capture a response XML in its schema.

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 two business services: Outbound Message Dispatcher (**F1-OutmsgDispatcher**) and Outbound Message Mediator (**F1-OutmsgMediator**) that further facilitate making web service calls. Both business services are similar, allowing the calling script to configure the following behavior (with differences noted):

- 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.
- Whether or not the sent message is persisted as an actual outbound message record.
 - If a persisted message is desired, the recommendation is to use the Outbound Message Dispatcher. This business service creates the message using standard BO processing, relying on the outbound message logic to route the message and store the record. The message is routed after the BO pre-processing algorithm and after the record is persisted but before the BO post-processing and audit plug-ins are executed. If you need to send the outbound message ID as part of the message, refer to [Capturing the Outbound Message ID in the Message](#) for more information.
 - If the message should not be persisted, then the recommendation is to use the Outbound Message Mediator. As mentioned, the Outbound Message Dispatcher creates the outbound message record, relying on the outbound message logic to route the message. If it should not be persisted, it is subsequently deleted. In contrast the Outbound Message Mediator executes the BO pre-processing algorithms and then routes the message explicitly without creating a message record. It is more efficient for scenarios that don't require persistence. Note that the Outbound Message Mediator also supports persistence, but it does so by creating the records without using BO processing. This is not recommended. The Dispatcher is the better option if persistence is desired.

Refer to the descriptions of the two business services for more information.

Designing 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 Business Object and Type

The product supplies many outbound message Business Objects along with Outbound Message Types for out of the box functionality.

In addition, implementations may need to define configuration for custom outbound messages. 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 elements that make up the XML Source field (XML_SOURCE). These are the elements that are the basis for the XML message. XSL transformations may be applied to this XML source to produce the XML message.

If your integration is real-time and a response is expected, the Outbound Message business object should also map to the XML Response field (XML_RESPONSE).

- You may decide to capture the response as is and define the element as “raw”. For example

```
<responseDetail mapXML="XML_RESPONSE" type="raw"/>
```

In this scenario, a Response XSL may or may not be needed.

- Alternatively, if the details of the response are needed, you may define specific elements for the response. For this option, depending on how the integration is designed, a Response XSL may be needed to transform the response into the expected XML format.

Once you have your business object and schema, define an [outbound message type](#) for each unique outbound message.

Referencing an XSL

The system supports defining the XSLs as a [managed content](#) record. For backward compatibility, the system supports defining the XSL as a file in the file system. This is a system wide setting, defined using a feature configuration option. The feature type is **External Messages** and the option type is **XSL Location**. Set the value to **F1FL** to support XSL in the file system. The file location is configured using a system property. To support the XSL in managed content, no option is needed as this is the default. You may explicitly define the value of **F1MC** if desired.

Capturing the Outbound Message ID in the Message

If your integration would like to use the system generated Outbound Message ID as a unique identifier with the external system, the following configuration is needed:

- Define an element within the XML Source that should be populated with the system generated outbound message.
- Configure a BO Option on the outbound message's business object using the Option Type **Outbound Message ID XPath** and set the Option Value to the XPath of the element defined to capture the ID.

NOTE: This functionality is only applicable if the outbound message is persisted.

Support for Dynamic URLs

The product supports the ability to build a dynamic URLs. These are cases where the URL requires information determined at runtime. This is supported with a combination of BO schema definition, URL configuration and appropriate code when creating the outbound message. The following points highlight the steps needed to support this functionality.

- When defining the [Message Sender's](#) URL, use the syntax **\${pathParms}** in the location of the URL where runtime information must be inserted. For example: **http://[SERVER_NAME]:1000/rest/services/\${pathParms}**
- Include the data area **F1-OM-DynamicConfig** (Outbound message dynamic configuration) in the schema of the business object for the outbound message. This data area includes elements for **pathParms** and **queryParms**.

- In the code that creates the outbound message, populate the **pathParms** and if applicable **queryParms** elements with the appropriate information. The system will build the URL by plugging in the **pathParms** element value followed by a question mark, followed by the **queryParms** element value into the **{pathParms}** location in the URL.

The following is an example of run time values.

```
<dynamicConfiguration>
  <pathParms>job/1234</pathParms>
  <queryParms>firstName=John&lastName=Doe</queryParms>
</dynamicConfiguration>
```

Add SOAP Header Parameters at Runtime

The product supports the ability to add SOAP header parameters to an external message at runtime. The following points highlight the steps needed to support this functionality.

- Include the data area **F1-OM-DynamicConfig** (Outbound message dynamic configuration) in the schema of the business object for the outbound message. This data area includes the element **soapHeaders** .
- In the code that creates the outbound message, populate the **soapHeaders** element with the self contained XML to add to the SOAP header section of the outgoing SOAP request.

Real-Time Message Configuration

Messages are routed to an external system real-time using the outbound message dispatcher or using the real-time send email business service. The system supports routing messages using HTTP and routing messages using JMS. In addition, there is a special type of message sender used for sending emails. The following sections highlights the supported real-time communication and the configuration needed for each.

Email Messages

For sending emails, the following configuration is needed:

- Define a [message sender](#) configured for email. Senders of this type should be configured with the **RTEMAILSNDR** class. The sender context is used to configure the connection information for the connecting to the SMTP server.
- This sender may be defined as the default email sender 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.

Outbound Messages

For other outbound messages that are routed using the real-time outbound message dispatcher, a [message sender](#) must be configured to define how to route the message. The following points highlight more detail related to this configuration.

Determine the communication mechanism prior to configuring the sender.

- When routing the message using JMS, the following configuration must be defined
 - Define an appropriate [JNDI Server](#) that indicates where the JMS resources are located.
 - Define a [JMS Connection](#) to define additional configuration needed for the connection.
 - Define the [JMS Queue](#) or [JMS Topic](#) to define the queue or topic to use.
- When communicating using a JSON format, determine the method to use to convert the XML to JSON. The desired method is driven by how the request should be sent.
 - If choosing the **Base JSON Conversion** method, if XSL transformation needs to be applied prior to the conversion to JSON, then the target XML Request Schema must be defined (using a data area) so that the conversion logic knows the format of the XML it is converting. The XSL is applied to the outbound message's XML Source, resulting in the defined XML Request Schema, which is then converted to JSON. If XSL transformation is not required, then the outbound message's XML Source is converted to JSON.

- If choosing the **Rootless JSON Conversion** method, the group element mapped to the XML source field is removed by the conversion, resulting in a rootless JSON request document.
- If the XML source on the outbound message can be converted to JSON using an XSL, then the **XSL Transformation** method may be chosen.
- You may also choose to convert the XML Source to JSON via the **Standard API Conversion** method (using a Jettison library). With this method, an XSL may optionally be provided. The conversion will be performed on the transformed XML.
- For the response, if the outbound message BO defines detailed elements for the XML Response field, then the JSON should be converted to this format.
 - If your conversion method is **Base JSON Conversion**, then if the JSON response cannot be converted directly to the XML Response elements on the outbound message BO, then define a Response Schema (data area) that represents the results for the base JSON conversion. In addition, define an XSL that can transform the response from the converted XML to the XML format expected on the BO.
 - If your conversion method is **Rootless JSON Conversion**, the response document is assumed to be rootless. The group element mapped to the XML response field is added by the conversion process, resulting in a well formed XML response document.
 - If the conversion method is **Standard API Conversion** or **XSL Transformation**, the standard API is used to convert JSON to XML. An XSL may be defined to convert the response to the XML Response if needed.
- If the outbound message BO defines a "raw" element to capture the response, then a response schema and XSL are not necessary. In this case, the system will perform a JSON to XML conversion using the **Standard API Conversion** method (regardless of the conversion method defined) and the result is captured in the XML response.
- For HTTP senders including JSON senders, the system provides the following support for sending messages secured by OAuth authentication:
 - Using Oracle Web Services Manager (OWSM). The system provides a pre-configured set of policies for OAuth (**F1-OAUTH**) using a special extendable lookup. Note that the values of this policy set defines a specific CSF key repository that the implementation should use for capturing its CSF keys. In addition, there is a substitution value defined for the token URI: @F1_OAUTH2_URI@. Configure the appropriate URI for this implementation as described in [URI Substitution](#). By default the system does not support additional policy sets to be defined. If your implementation requires a different policy set, contact support.
 - You may provide OAuth related settings as part of the [message sender](#) configuration for a REST API.

Define a [message sender](#) configured for each appropriate routing method. The invocation type should be configured as **Real-time**. For routing via HTTP, use the **RTHTTPSNDR** - HTTP sender class. For routing via HTTP with SOAP format automatically applied, use the **SOAPSNDR** - HTTP SOAP sender class. For routing via HTTP using JSON format, use the **RTJSONSNDR** - JSON sender class. For routing via JMS, use the **RTJMSQSNDR** - JMS queue sender class or **RTJMSTSNDR** - JMS topic sender class and configure the JMS Connection and JMS Queue or JMS Topic. Use the sender context to configure the required values for connecting to the appropriate destination.

NOTE: Refer to [Support for Dynamic URLs](#) for configuration needed to support dynamic URLs when sending an outbound message. There is specific configuration expected when defining the URL on the Sender in order to support dynamic URLs.

Configure the [external system](#) / outbound message type collection. The processing method defined for the external system / outbound message type must be **Real-time**.

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.

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. Note that the validation is applied just before communication with the sender and therefore after any Request XSL has been applied.

- 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 use for validation.

You may turn on or off this validation checking using the Message option **Schema Validation Flag**.

This feature is not supported for cloud implementations.

Configuring the System for Outbound Messages

The following sections describe the setup required when using [Outbound Messages](#) to communicate with an external system.

JNDI Server

If using JMS to communicate outbound messages, define a new JNDI Server. Open **Admin > Integration > JNDI Server**.

Description of Page

Enter a unique **JNDI Server** and **Description**.

Indicate the Provider URL to indicate the location of the JNDI server. A variable may be used in place of all or part of the URL. The variable must be predefined in the substitution variable property file. The value here should enclose the variable name with @. Refer to [Referencing URIs](#) for more information.

Indicate the **Initial Context Factory**, which is a Java class name used by the JNDI server provider to create JNDI context objects.

JMS Connection

To define a JMS Connection, open **Admin > Integration > JMS Connection**.

Description of Page

Enter a unique **JMS Connection** and **Description**.

Indicate the **JNDI Server** to be used. Refer to [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

JMS Queue

To define your JMS Queue values, open **Admin > Integration > JMS Queue**.

Description of Page

Enter a unique **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 **JNDI Server** where the queue is defined. Refer to [JNDI Server](#) for more information.

JMS Topic

To define your JMS Topic values, open **Admin > Integration > JMS Topic**.

Description of Page

Enter a unique **JMS Topic** and **Description**.

Select the **JNDI Server** where the topic is defined. Refer to [JNDI Server](#) for more information.

Enter the **Topic Name** as defined in the JNDI server. This is the JNDI lookup name identifying the topic.

Message Sender

The topics in this section describe the maintenance of a message sender

Message Sender - Main

To define a new sender, open **Admin > Integration > Message Sender**.

Description of Page

Enter a unique **Message Sender** and **Description**.

CAUTION: Important! When adding new records, carefully consider the naming convention of the message sender code. Refer to [System Data Naming Convention](#) for more information.

Set **Invocation Type** to **Real-time**.

NOTE: The invocation type **MPL** remains in the product for upgrade purposes but is not recommended.

Indicate the **Message Class** for this sender, which indicates the method used to route the message. The real-time sender classes are as follows:

Message Class	Description
RTEMAILSNDR	Email sender.
RTHTTPSNDR	HTTP sender.
RTJMSQSNDR	JMS queue sender.
RTJMSTSNDR	JMS topic sender.
RTJSONSNDR	HTTP JSON sender.
SOAPSNDR	HTTP SOAP sender.

The following sender classes are related to MPL processing and remain in the product for upgrade purposes:

Message Class	Description
DWNSTGSNDR	Download Staging sender.
EMAILSENDER	Email sender.
FLATFILESNDR	Flat file sender.
HTTPSNDR	HTTP sender.
JMSSENDER	JMS Queue sender.
OUTMSGSNDR	Outbound Message sender.
STGSENDER	Staging Upload sender.

Message Class	Description
TPCSNDR	JMS Topic sender.
UPLDERRHNDLR	Upload Error Handler.

Indicate whether or not this sender is currently **Active**.

For JMS related senders, indicate whether the **MSG Encoding** is **ANSI message encoding** or **UTF-8 message encoding**. (Note that for all other types of senders, the Character Encoding context type is used to configure encoding.)

If the Message Class is one that connects to a JMS Queue or JMS Topic, indicate the appropriate **JMS Connection**

FASTPATH: Refer to [JMS Connection](#) for more information.

If the Message Class is one that connects to a JMS queue, indicate the name of the **JMS Queue** to define where the message is to be sent.

FASTPATH: Refer to [JMS Queue](#) for more information.

If the Message Class is one that connects to a JMS topic, indicate the name of the **JMS Topic** to define where the message is to be sent.

FASTPATH: Refer to [JMS Topic](#) for more information.

The **XAI JDBC Connection** remains for legacy purposes but is not applicable for supported functionality.

Message Sender - Context

The sender may require context information to define additional information needed by the system to successfully send outgoing messages. Open **Admin > Integration > 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 the supported context values for different types of senders.

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 **RTHTTPSNDR**. 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 **RTHTTPSNDR**, **RTJSONSNDR** or **SOAPSNDR**.

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. Note that the substitution string @XMLMSG@ may be used for GET calls if an XSL has been applied to convert the message into HTTP GET parameters. It is useful if the HTTP Form is not applicable to the type of message.</p> <p>Refer to Support for Dynamic URLs for configuration needed to support dynamic URLs when sending an outbound message.</p>	
HTTP Method	<p>The HTTP method used to send the message.</p> <p>NOTE: The SOAP sender message class SOAPSNDR only supports the POST method.</p>	POST or GET
HTTP Transport Method	<p>Specifies the type of the message. You can either send the message or send and wait for a response.</p>	Send or sendReceive
HTTP Form Data	<p>Used when the message is in the format of an HTML Form (Content-Type: application/x-www-form-urlencoded).</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 x=y where x is the form parameter name and y is its value.</p> <p>If y contains the string @XMLMSG@ (case sensitive) then this string is replaced by the content of the service response XML message. The @XMLMSG@ string can be</p>	

Context Type	Description	Values
	<p>used in the HTTP Form Data or in the HTTP URL, but not in both.</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 @XMLMSG@ is not specified in one of the context records.</p> <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.	UTF-8 or UTF-16
Response Time Out	The amount of time the system should wait for the remote system to send a response.	
Sender Security Type	Indicate the desired security type to apply. (Note that additional security types are applicable only to the SOAP Sender. See below for more information.)	BASIC (HTTP Basic), TEXT (Username Token plain text), DIGEST - Username Token Digest, OWSM - OAuth Security via OWSM, OAUTH - OAuth Security.

OWSM Policy Set	Applicable only if the Sender Security Type is OWSM . Defines the policy set to apply.	Enter a valid value for the extendable lookup Set of Policies (F1-SetOfPolicies). The product provides the value F1-OAUTH that may be used here.
OAuth Grant Type	Applicable only if the Sender Security Type is OAUTH . Defines the OAuth specification method for acquiring an access token.	client_credentials
OAuth Access Token URL	Used to construct the URL of the target authorization server. You may use substitution variables to construct the URL.	
OAuth Client ID	The generated identifier for the current application in the external system's authorization server.	
OAuth Client Secret	The secret paired with the OAuth Client ID as provided by the external system's authorization server.	
OAuth REST API Scope	The authorization level granted by the end point.	

Real-time HTTP Sender

The following context type is only applicable to senders with the **RTHTTPSNDR** message class.

Context Type	Description
Content Type	Populate a value here to override the Content-Type attribute in the HTTP header, which defaults to text/xml .

SOAP Sender

A SOAP sender is an HTTP sender that automatically adds support for the SOAP format. For this type of sender (message class of **SOAPSNDR**), besides the context values listed above, the following context entries may also be supplied.

Context Type	Description	Values
Message Namespace URI	Used to indicate a specific namespace to be included in messages for this sender. Note that this value is used only when the External Message link to this sender is configured with a Namespace Option of Configured on Sender .	
SOAP Insert Timestamp	Indicate whether a timestamp should be added. Default value is 'N'.	Y or N
SOAP Expiration Delay (in seconds)	Indicate an expiration delay to add to the timestamp. The default value is 60.	
Sender Security Type	Additional security types are supported for SOAP senders for specific third party integrations.	CIM - Uses an appropriate namespace and adds security information in the SOAP header specific to Landis+Gyr, MS_V30 - Uses an appropriate namespace and adds security information in the SOAP header specific to MultiSpeak v3.0, MS_V41R - Uses an

Context Type	Description	Values
		appropriate namespace and adds security information in the SOAP header specific to MultiSpeak v4.1. ITRON - Uses appropriate SOAP header information security information in the SOAP header specific to Itron.

NOTE: Refer to [Add SOAP Header Parameters at Runtime](#) for information about dynamically including SOAP Header parameters when sending a message.

JMS Sender

A JMS sender is one that sends messages to a JMS queue or JMS topic. JMS senders should reference a Message Class of **RTJMSQSNDR** or **RTJMSTSNDR**, respectively.

The following parameters are used to connect to the JMS resource.

Context Type	Description	Values
JMS Message Type (Bytes(Y)/Text(N))	Indicates whether the data is sent as a bytes message or as a text message.	Y or N
JMS User Name	Enter the user name to connect to the JMS resource.	
JMS User Password	Enter the password to use to connect to the JMS resource.	
JMS Header	If JMS header values are required for the message, use this context type. For each JMS header that has to be specified, add a context record with a value having the following format x:y where x is the header name and y is the value.	

Defining Outbound Message Types

Refer to [Outbound Messages](#) for an overview of this functionality.

Use this page to define basic information about an outbound message type. Open this page using **Admin > Integration > Outbound Message Type**. You are brought to a query portal with options for searching for outbound message types.

NOTE: This page is not available if the **External Message** module is [turned off](#).

Once an outbound message type has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Outbound Message Type** zone provides basic information about an outbound message type. Besides the code and description, the outbound message type captures the outbound message business object and the relative priority of the outbound message. The priority is only applicable if the [outbound message is processed using OSB](#). In addition, if the outbound message type is linked to any web service categories, they are displayed.

CAUTION: Important! When adding new records, carefully consider the naming convention of the outbound message type code. Refer to [System Data Naming Convention](#) for more information.

Note that in addition to standard actions available on this portal, there is also a special **Add to Category** button in the page action area. Click this button to link an outbound message type to one or more [web service categories](#).

The **External System Links** zone is visible if the outbound message type is linked to any external systems.

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 > Integration > 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**.

CAUTION: Important! When adding new records, carefully consider the naming convention of the external system code. Refer to [System Data Naming Convention](#) for more information.

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. Valid values are **Batch**, **Real-time**, **SOA** and **XAI**.

Define an appropriate **Message Sender** if the processing method is **XAI** or **Real-time**.

Namespace Option is used when your message should include a namespace in the resulting XML. The valid options are **Standard Namespace** and **Configured on Sender**. If the value is Standard Namespace, the system will generate a namespace for the resulting WSDL with the following value: http://ouaf.oracle.com/outbound/AAA_BBB, where AAA is the external system code and BBB is the outbound message type code. If the value is Configured on Sender, then the value of the namespace is taken from the sender context with context type **Message Namespace URI**.

Define an appropriate **Batch Control** if the processing method is **Batch**.

If the message sender is one with a message class of **RTJSONSNDR**, indicate the **JSON Conversion Method**. The valid values are **Base JSON Conversion**, **Standard API Conversion** and **XSL Transformation**. Refer to [Real Time Message Configuration](#) for more information about these methods and the additional configuration that may be applicable.

If the **JSON Conversion Method** is **Base JSON Conversion**, the **Request Schema** is enabled. Populate a data area that defines the schema for the XML format to convert the outbound message's BO schema to prior to performing the JSON conversion. Refer to [Real Time Message Configuration](#) for more information.

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**. Refer to [Referencing an XSL](#) for information about configuring where the XSL is defined.

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**. This feature is not available for cloud implementations.

If the **JSON Conversion Method** is **Base JSON Conversion**, the **Response Schema** is enabled. Populate a data area that defines the schema for the XML format that the JSON message is initially converted to. The XML is then converted to BO XML. Refer to [Real Time Message Configuration](#) for more information.

Response XSL will only be displayed when the processing method is **Real-time**. Refer to [Real Time Messages](#) for more information on how it is used. Refer to [Referencing an XSL](#) for information about configuring where the XSL is defined.

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 > Integration > 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.

Message Option

The Message Option page defines various system settings used by the system when processing external messages.

To define options for your environment, open **Admin > Integration > Message Option**.

Description of Page

The following options are supported.

Option	Description	Option Name
Default Email Sender	This is the default Message Sender used for sending e-mails when no explicit Message Sender is specified.	defaultEmailSender
Default User	The default user is used by the system to access your product when no other user is explicitly specified.	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

Option	Description	Option Name
Messages Language	The default language to use for the messages.	language
Outbound Message Schema Location	Enter the full path of the virtual directory where valid W3C schemas are stored if your implementation wants to validate outbound message schemas . For example: http://localhost/cisxai/schemas.	xaiOuboundSchemaLoc
Schema Validation Flag	Enter Y to turn on schema validation for outbound messages . Enter N to turn this off.	xaiSchemaValidationCheck
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 F1-INJMS that may be used here.	toDoTypeforInboundJMSMessageErrors
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 F1-OUTMS that may be used here.	outboundErrorTodo

Note that the following options are no longer applicable.

Option

Automatically Attempt Resend to Unavailable Sender (Y/N)

Default Response Character Encoding

JDBC Connection Pool Max size

Maximum Errors for a Sender

Messages JDBC Connection

MPL Administrator Port

MPL HTTP Server Authentication Method

MPL HTTP Server Password

MPL HTTP Server User

MPL Log File

MPL Trace File

MPL Trace Type

Privileged Users

Records MPL Receiver Will Process At a Time

Schema Directory

Send SOAP Fault as HTTP 500

Sender Retry Seconds

System Error JDBC Connection

System Error Max Retry

System Error Retry Interval

Thread Pool Initial Size

Thread Pool Max Size

Thread Pool Non Activity Time

WSDL Service Address Location

XAI Authentication Password

XAI Authentication User

XAI Trace File

Option

XAI Trace Type

XSD Compliance

XSL Directory

Managing Outbound Messages

Use this page to view information about outbound messages.

Outbound Message - Main

Open this page using **Menu > Integration > 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 or **Real-time**. (The value of **XAI** is no longer supported). 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 no longer applicable..

The **Creation Date** indicates the date that this record was created.

The status is no longer applicable.

Outbound Message - Message

Use this page to view the XML source used to build an outbound message. Open this page using **Menu > Integration > 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 > Integration > 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.

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 > Integration > 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" using a specific business service. 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 real time. 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. In addition, the business service supports using a default message sender defined as a [message option](#). Review the business service schema for information about the input elements.

NOTE: Retry setting. An option in the system properties file allows your implementation to configure the number of times to retry (if any) if the SMTP server is unavailable. Refer to the server administration guide for more information.

- The business service supports sending attachments. The may either reference attachments from the file system (file name and content ID) or reference attachment records captured in the system's attachment table. Note that attachments in the file system are not supported in a cloud implementation.

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.

- Using an outbound message to send an email. This option allows for different variations as described in [Outbound Messages](#).
 - Some emails may be created en masse (for example a large group of emails routed to users for a given set of To Do entries). In this case, the records can be created in the staging table for processing using OSB.
 - Messages may still be sent real time using one of two business services described in [Real Time Messages](#). This option is an alternative to the dedicated email service described above when aspects of the outbound message functionality are needed, such as the ability to instantiate a record as an audit or to include additional logic via BO plug-ins as part of sending the email.

Web Service Category

The product provides the ability to categorize web services that are defined in the system. The term web services refers to Outbound Message Types, Inbound Web Services, and for those implementations where they are applicable, XAI Inbound Services.

A given web service may be linked to more than one category. The product supplies web service categories and most base delivered web services are configured with appropriate categories. Implementations may define new categories and may link custom web services to base delivered categories or to custom categories. In addition, implementations may link additional categories to base delivered web services. However, implementations may not remove base delivered web services from base delivered categories.

If your implementation uses the [Integration Cloud Service Catalog](#), the integration provides the categories for each web service that is sent to the catalog.

The web service category portal supports adding and removing web services from the category. In addition, the [Outbound Message Type](#) portal and [Inbound Web Service](#) portal each provide a page action button to **Add to Category**.

Defining Web Service Categories

Refer to [Web Service Category](#) for an overview of this functionality.

Use this page to define basic information about a web service category. Open this page using **Admin > Integration > Web Service Category**. You are brought to a query portal with options for searching for web service categories.

NOTE: This page is not available if the **External Message** module is [turned off](#).

Once a web service category has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Web Service Category** zone provides basic information about a web service category.

CAUTION: Important! When adding new records, carefully consider the naming convention of the web service category code. Refer to [System Data Naming Convention](#) for more information.

Once a web service category exists, an **Add Web Services** zone is provided to find and select web services to link to the web service category.

When viewing a category that already has web services linked to it, they are visible in the zone **Included Web Services**. This zone allows a user to remove a customer owned web service from the list.

JMS Message Browser

The JMS Message Browser portal allows you to select a JMS queue and view messages currently in the queue.

In order for a JMS queue to be available on the portal, a [message sender](#) must be defined that is configured for the appropriate JMS queue with the credentials to connect to the queue.

- If your organization sends real-time outgoing messages to a JMS queue, this configuration would exist as per the details in [Real-Time Message Configuration](#).
- If inbound web service messages are routed to the system via a JMS queue, no configuration is needed in the system. However, if you would like to view the messages in the queue in the JMS message browser portal, configuration for the JMS queues as described for outgoing messages is required.

Navigate to the portal using **Main > Integration > JMS Message Browser**.

The JMS Senders zone provides a list of configured message senders eligible for selection.

The JMS Message List zone is visible for the JMS Sender broadcast from the first zone. This zone supports selecting one or more records to delete from the queue. Use the message selector to limit the results to messages that satisfy the message selector. This uses standard JMS API message selector functionality. Refer to the zone embedded help for information about the supported syntax.

The JMS Message Details zone displays a message broadcast from the list zone.

Oracle Integration Cloud Catalog

Oracle Integration Cloud (OIC) is an offering that serves as integration infrastructure for Oracle cloud solutions. The product provides an adapter for OIC to streamline integration between your edge application and OIC.

The product provides a mechanism (referred to as an adapter) for OIC to retrieve the REST catalog and another to retrieve the SOAP catalog from a given product. The expectation is that a given implementation would use either REST or SOAP and not both, but for those implementations that do support both classes of web services, OIC can retrieve both catalogs.

For the SOAP catalog, the adapter retrieves the name, the source system, the WSDL location and namespace for each web service.

For the REST catalog, the adapter retrieves the name, the source system and the OpenAPI specification for each web service.

It is possible that not every web service supported by an edge product is managed by OIC. In order to only include the appropriate web services in the adapter, configuration is needed to identify which web services to include.

The web service catalog is used to identify the records that should be retrieved by the adapter. Separate catalogs exist for REST and for SOAP

- For SOAP inbound messages, the system supports both the use of [inbound web services](#) and XAI inbound services that are [deployed via IWS](#). Each IWS or XAI inbound service that should be included in the catalog must be flagged in the SOAP catalog. Note that only [deployed](#) services are returned to the catalogue.
- For REST inbound messages, each IWS that should be included in the catalog must be flagged in the REST catalog.
- For outbound message, the system requires the creation of an External System that includes each outbound message type that the external system receives. For outbound messages that are integrated through OIC, the external system itself will represent OIC. Rather than identifying each outbound message type to include in the catalog, only the external system needs to be flagged. The adapter will return web service information for all the outbound message types configured for the external system. If an implementation supports both REST and SOAP, separate external systems should be created to separately group the REST and SOAP outbound messages.

NOTE: The subordinate catalog functionality is no longer applicable and will be removed in a future release.

FASTPATH: Refer to [Maintaining the Web Service Catalog](#) for more information.

Web Service Catalog Configuration

The topics in this section describe the configuration needed in your edge application to integrate with OIC.

Web Service Catalog Master Configuration

The **Service Catalog Configuration** ([master configuration](#)) record defines several system wide settings related to integrating with the service catalog.

It includes configuration for the WSDL base URL for the SOAP catalog. Note that this element supports the functionality described in [Referencing URIs](#). Note that for the REST catalog, the system will use the URL defined in the substitution variable **F1_OPEN_API_BASE_URL**

The subordinate server section is no longer applicable and will be removed in a future release.

For more information about specific fields in the master configuration, refer to the embedded help.

Maintaining the Web Service Catalog

Refer to [Oracle Integration Cloud Catalog](#) for an overview of web service catalog functionality.

To add or remove services that are reported to the OIC catalog, navigate to the portal using **Admin > Integration > Web Service Catalog**.

The Catalog by Class zone provides the list of classes (REST and SOAP). Broadcast the web service class that you want to work with.

The Catalog zone provides a list of services that are currently in the catalog based on the web service class chosen. Users may use this zone to remove services from the catalog.

The Candidate Services zone provides a list of external systems and inbound web services that are not currently linked to the catalog. Users may use this zone to add services to the catalog. The inbound web services will be limited to those with the same web service class as the one broadcast. All external systems are shown regardless of the web service class because there is no web service class configuration on external systems.. The expectation is that implementers will create the external system to include either only REST or only SOAP services and will know which external system to include in which catalog. For the SOAP web service class, if XAI inbound services are applicable, they are also visible.

NOTE: For inbound SOAP web services and XAI inbound services, they may be selected for the SOAP catalog at any time. However, only [deployed](#) services are returned to the catalog by the OIC adapter.

XAI Documentation Note

The XAI functionality is legacy functionality and not recommended for new implementations. The help topics are no longer provided.

Integrations

This chapter provides high level information about product integrations supported for all products that use Oracle Utilities Application Framework.

Creating Application Links in External Applications

When integrating Oracle Utilities applications with other external applications, you can create links from external applications to specific portals and pages within the Oracle Utilities applications (also known as "deep linking"). These links typically leverage Business Process Assistant (BPA) scripts to help navigate the user to the appropriate target portal or page and to make provided parameter values available for use on the portal or page.

NOTE: This technique may also be used within the product to launch a new browser tab from within the application when it's important to maintain the current screen. For example, launching the GIS Viewer in Oracle Utilities Work and Asset Management by creating a new browser tab.

To do this, parameters are added to the application URL by appending a question mark (?) to the end and using the syntax "key=value". If you require more than one parameter, use an ampersand (&) to separate each key=value pair. There are several predefined parameter key values recognized by the application. In addition, specific parameters understood by the script may be provided. The following list highlights the predefined parameters.

- **location**=<navigation key> - If the exact page to launch is known, the navigation key value of that page can be part of the URL using 'location='. For example, if an approval process has sent an email to a user to approve an adjustment, the link configured in the email may include the adjustment's navigation key.
- **script**=<BPA script name> - In almost every use case, a script is needed to help navigate to the correct page and display the correct information. Even if a location is known and provided in the URL, if there is context data provided as a parameter, it is the responsibility of a script to move that parameter data from the URL into the page data model, so that the page can access the information and display the correct record. In cases where the target page is dynamic based on information provided by parameters, the script may be used to determine what page to launch.
- **portal=true** - When using the location parameter where the target page is a portal page and passing context data, use this parameter. The allows the system to know that the parameter data must be pushed into portal context.
- **initNav=false** - By default, if a script is provided but the location attribute is not provided, the system navigates to the user's home page prior to executing the script. If the script itself includes a step to navigate to a target page as one of its initial steps, the navigation to the home page is unnecessary and may degrade performance. This optional attribute may be included in the URL to bypass the home page.

NOTE: The system still requires a page to be launched for technical reasons. A blank portal with no zones is used for this purpose. Users may see this portal (called **Launching Application**) briefly before the navigation initiated by the script. In addition, this is the portal that the user will remain on if there are any errors in the script or if the script does not navigate anywhere.

- **minDashboard=true** - This option minimizes the dashboard area as part of launching the application. (The user will be able to maximize the dashboard themselves if needed). You would use this option to potentially improve the speed at which the application is launched. Whether this is useful will depend on the use case. If the reason for launching the application is to bring the user to a specific record to view details about that record, the dashboard may be providing valuable information about the record. In this case, it may be correct to display the dashboard by default (and not include this parameter in the URL). On the other hand if the use case is that a user is already in the application and clicks a link that opens a new browser tab to show the user specific detail about a chosen record, while keeping the original page open on the first tab, the dashboard may contain the same information already visible on the original tab. In this case, minimizing the dashboard may be beneficial.

For example, the following URL illustrates launching the script **CM-StopAutoPay** for a given account, assuming the application URL for launching the system is `http://system-server:1234/cis.jsp`:

- **`http://system-server:1234/cis.jsp?script=CM-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=CM-StopAutoPay&location=accountMaint`**
- **`http://system-server:1234/cis.jsp?ACCT_ID=1234512345&location=accountMaint&script=CM-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 would also be a temporary storage variable called **ACCT_ID** that holds the passed value).

Determining the Application URL

When building the link from within application logic, the service script **F1-EnvURL** may be used to retrieve and build the application URL information.

Navigate to a Given Record's Maintenance Portal

The product provides a script **F1-GotoPrtl** (Navigate to portal for an MO and key values) that may be used for any use case that simply navigates to the maintenance page for a given record and displays that record. This script is only applicable to records that are governed by a business object (and define a navigation option as a BO option). It requires the parameters **mo=** with the maintenance object code of the record and the parameters **pkValue1=** through **pkValue5=** for providing the key values. It looks up the appropriate portal navigation option from the record's BO, populates keys into the 'page data model' and navigates to the portal.

It is also recommended to include the **initNav=false** attribute when building a URL with this script. The example below navigates to the appropriate Migration Data Set portal for the given ID's business object.

- **`http://system-server:1234/cis.jsp?script=F1-GotoPrtl&pkValue1=1234512345&mo=F1-MIGRDS&initNav=false`**

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* technical reference 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-LDAP batch control** can be updated with system wide configuration.

- JNDI 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.

- If the LDAP service has any limitation to the number of objects that may be imported, configure the **LDAP Query Page Size** parameter to enable querying.

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: - Group - User
	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: - Group - User
	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 F1-LDAP 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

Element	Attribute	Description
		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. 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 one of the following occurs: - The LDAP attribute contains a null or empty value - The LDAP attribute does not exist or is not specified. Default values are applied only when creating a new entity and are not applied to updated entities.
	autoGenerate	Set this to true in order to turn on auto generation of the user ID. If this is true, the system will define the 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: uppercase truncate . Note that this attribute should not be used in conjunction with the autoGenerate attribute.
LDAPEntryLink	linkedToLDAPEntity	The name of the linked entity (User or Group). 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: - onelevel (the value normally used) - subtree

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>
</LDAPEntry>
</LDAPEntries>
```

```

<LDAPEntryLinks>
  <LDAPEntryLink linkedToLDAPEntity="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 linkedToLDAPEntity="User" linkingSearchFilter="
(&(objectClass=inetOrgPerson)(departmentNumber=%distinguishedName%))"
linkingSearchScope="onelevel" />
  </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* technical reference paper.

In order to use this functionality, [feature configuration](#) options for the **External Messages** feature type must be configured.

- Set option type **Support SPML Deployment in IWS** to **true**.
- Set option type **Default SPML service security policy** to an appropriate value per your implementation rules.

Template User Functionality

The user object in this product captures configuration used to control access but also preferences. The identify 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 one 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* technical reference paper for more information.

User Redirection

The product allows an implementation to support the ability for a customer to authorize a customer service representative (CSR) to view the customer's account information in a customer facing portal (Digital Self Service), to see exactly what the end customer is seeing/experiencing.

When the request for the CSR to view the customer information is received, the following steps occur:

- The security server for the customer facing portal is contacted to obtain a temporary grant code.
- The redirection response returns the grant code and the URL of the target customer facing portal.
- The browser redirects to the user to this URL with the grant code, which is further authenticated and automatically exchanged for an authorization token, allowing the CSR to log in.

Standard security measures with private/public key encryption using JSON Web Tokens (JWT) ensures that only the implementation's enterprise application is allowed to communicate with the implementation's customer facing portal server. The system also provides a plug-in spot to allow for additional information, such as the customer's name and email address, to be included in the call to the security server.

This feature works in conjunction with functionality provided by the enterprise customer application. Refer to the documentation for that product for information about how this feature is supported.

NOTE: The product has been designed to potentially support other types of redirection besides the mentioned use case by adding another entry in the Redirection Target lookup (F1_REDIRECTION_TARGET_FLG). However, the internally provided services work specifically for the Digital Self Service use case. There is no guarantee that they would work for a different redirection use case. Please contact product support if your implementation would like to implement a different redirection scenario.

Redirection Security Context Plug-in

The product supports a plug-in spot to allow for the system to populate additional keys and values to add to the JWT (JSON Web Token) when calling the external security service.

In addition, the product provides an Extendable Lookup BO **Redirection Security Context (F1-RedirectionSecurityContext)** to define the key mnemonic values. For an enterprise product or implementation to populate additional keys, use the extendable lookup to define the key mnemonics. Then provide an algorithm that uses the available information about the customer and populates the security context keys and values by determining the related data in the system.

Master Configuration

The master configuration **Redirection Targets Configuration (F1-RedirectionTargetsConfig)** defines the information needed to contact the security server to obtain a grant code. It also defines the redirection URL.

Build an Anchor Hyperlink

The service script **Return a Hyperlink to the Redirect Servlet for a given Target (F1TargetLink)** is provided to allow for a link to be placed on a UI map or in a zone to initiate the redirection request. Refer to the service script detailed description in the metadata for more information.

Batch Scheduler Integration

The Oracle Database includes an enterprise wide scheduler to simplify the scheduling of background processes. The scheduler is implemented by the *DBMS_SCHEDULER* package. The product provides an integration with the Oracle Scheduler to facilitate scheduling background processes shipped with the product.

At a high level, the integration with the Oracle DBMS Scheduler supports the following entities:

- **DBMS Program.** A program should be defined for each Batch Control that needs to be scheduled by the DBMS scheduler. A program would typically invoke a batch job, but it could be configured to set certain options instead.
- **DBMS Chain.** A Chain defines a series of steps with dependency rules between them. A step references a program, with the program performing the actual work for that step. A rule is attached to each step to identify its dependent steps and the condition for when that step should be executed. For example, in a chain consisting of STEP_A and STEP_B, where STEP_B can only start if STEP_A was successful, the rule for STEP_B to start would specify a condition of "STEP_A SUCCEEDED".
- **DBMS Schedule.** A predefined frequency for jobs that need to be run periodically, for example, nightly jobs.
- **DBMS Job.** Defines a plan to perform a specific program or a chain periodically on a specific schedule or ad-hoc.

The product provides a set of business services to maintain these entities as well as submit jobs, manage submissions and report on past submissions. Refer to business services that start with the prefix "F1-DBMS" for more information.

NOTE: For details on the integration, refer to the *Server Administration Guide* which contains the API. In addition, refer to the technical reference paper *Oracle Scheduler Integration* that provides guidelines for using this integration.

Data Synchronization

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.

Understanding Data Synchronization

The product supports various options for managing an ongoing synchronization of an entity to an external system. While each integration involves unique business rules, in many cases the process is quite similar and involves the following overall steps:

- When an entity is added, changed or deleted logic determines whether the change is eligible for synchronization and if so a sync request record is initiated to manage the synchronization process. The base product uses the **Audit** plug-in spot on the maintenance object to allow for such logic to be performed when a change is detected to a record for that MO. The framework calls the algorithm defined on this plug-in spot in the event a change to the MO has been detected. Refer to the description of the plug-in spot on [Maintenance Object - Algorithms](#) for more information about when this plug-in is called.
- The processing of the sync request is deferred to batch so that the communication with the external system does not impact the updating transaction. It also ensures that the data being shared with the external system is fully committed to the database.
- When processed, sync request logic manages the following aspects of the process:
 - Determines whether the synchronization is still needed and if not discards the request.
 - Prepares the outbound message with the entity's synchronization details and sends it to the external system.

- Manages errors and retry attempts as needed.

The following sections describe the various data synchronization options supported by the product.

Accumulative Data Synchronization

The product supports a data synchronization method by which several changes made to an entity may be accumulated and sent together as a single message to the external system.

The following points describe the functionality at a high level:

- When an entity is added, changed or deleted, a snapshot of the entity's data from before the change is captured on one or more sync requests, one for each external system the data needs to be interfaced to. The before-update snapshot is used as a base line for detecting all changes made since then at the time of sending the message to the external system.
- Sync requests are processed by the next monitor batch process.
- The processing of the sync request typically involves business rules that compare the before-update snapshot with the current content of the entity to detect all change made. If changes were made, logic may decide to proceed with the sending the synchronization message otherwise the request is discarded. These rules are also responsible for handling any errors encountered while communicating with the external system and manage retry attempts as needed.

This functionality is supported by logic supplied in the base business object Sync Request (**F1-SyncRequest**). Note that each edge application delivers an appropriate child business object for this BO for each specific synch scenario supported in that product. Some of the functionality below is accomplished using configuration on the parent BO delivered by the framework while other functionality may be delivered by the child BO. In addition, there may be more complex use cases supported by your specific product integration. Refer to your specific application's library of Sync Request business object along with the documentation related to your specific product integration for more information.

The following sections describe key aspects of this data synchronization approach.

Capturing the Change

The base product provides a change data capture (MO Audit) algorithm **F1-GCHG-CDCP** that may be used by maintenance objects that need to be synchronized by this method. This algorithm creates a Sync Request record for the changed record, capturing the MO code and the primary key, if it doesn't find an existing sync request for the same record (and the same business object) in the initial state. The sync request business object used is the one defined in the **Sync Request BO** option on the MO for the record that was changed. If more than one such option exists then multiple sync requests are created.

Your specific product may also introduce additional Audit algorithms to cater for more sophisticated examples.

When creating the sync request record, typically the Sync Request BO will have a pre-processing plug-in that captures a snapshot of the record's data prior to its change. This will be used in subsequent steps to verify that the external system needs to be notified of the change.

Confirming that a Sync is Needed

Once a sync request is captured, there are several steps performed prior to any information being sent to the external system.

NOTE: This section only highlights key steps. Please refer to the business object configuration, its lifecycle and algorithms for a thorough picture of the full functionality..

- When a Sync Request record is created, its initial state (**Pending**) is configured to be processed by a batch monitor. That way, records are added to the sync request table throughout the day but all are processed together. The MO audit algorithm ensures that a new synch request is not created if a Pending record already exists for a given MO / PK combination (for the same business object). However, it is possible that a record for that MO / PK exists in a subsequent "non-final" (such as **Awaiting Acknowledgement**). This state includes a monitor algorithm to check for that condition

and to skip transitioning if another record exists. This is done to ensure that the existing record is fully processed before this new record is processed.

- The next state of the lifecycle is **Determine if Sync Needed**. This step uses an algorithm to take a snapshot (called the ‘final snapshot’) of the data and compare it against the initial snapshot taken when the record was created. Based on the logic of the algorithm, it may decide to proceed (transition to **Send Request** or to discontinue (transition to **Discarded**).

Communicating to the External System

Once it is confirmed that the sync should occur, a message must be sent to the external system. The following points highlight the basic functionality.

- An algorithm linked to the **Send Request** state. The expectation is that this algorithm creates an outbound message that routes the information to the external system appropriately. The algorithm must determine the external system and outbound message type to use. Business objects for Sync Request support BO options to define the external system and outbound message type to use for this algorithm.
- Once the outbound message is triggered, the record transitions to the **Awaiting Acknowledgement** status. This state is used to hold the sync request from further state transitions until an acknowledgement is received from the external system. Note that this step relies on implementation of a response mechanism from the external system. It is recommended to implement a response as this helps control the chronological flow of information. The product supplies 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. In addition, this state may be configured with a monitor algorithm that detects that a timeout limit has been reached.
- For records that enter the **Error** state, it is recommended to configure an algorithm that creates a To Do entry to alert someone of the problem. Refer to the integration documentation for more information. The state is already preconfigured with an algorithm to complete To Dos when exiting the state.
- The final state **Synchronized** is used to mark the successful synchronizations. However, for more complicated use cases, this state may be used to trigger some additional action. Refer to the documentation for your specific product integration for more information.

Generic Data Synchronization

The product supports a generic data synchronization method by which each change to an eligible entity triggers a synchronization flow that communicates details about the entity to one or more external systems.

The following points describe the functionality at a high level:

- The integration of a specific entity (maintenance object) and an external system is enabled and configured using a **Data Synchronization Control** (F1-DataSyncControl) extendable lookup record.
- Rules supporting the integration reside in a **synchronization script** referenced on the control record.
- When an entity is added, changed or deleted, a sync request may be initiated for each integration according to business rules defined by the corresponding synchronization script.
- Sync requests are processed by the next monitor batch process.
- When the sync request is processed, data is synchronized to the external system according to business rules defined by the corresponding synchronization script.

This functionality is supported by logic supplied in the base business object Generic Data Sync Request (**F1-GenericDataSync**). This business object does not need to be extended to support specific integration rules as these rules reside in the synchronization script associated with each integration’s data synchronization control record.

The following sections describe key aspects of this data synchronization approach.

Data Synchronization Control

The **Data Synchronization Control** (F1-DataSyncControl) extendable lookup defines the configuration needed to support an ongoing data synchronization of an entity to an external system. The configuration defines the entity's maintenance object, the external system it is synchronized with and references a synchronization script that manages the specific synchronization rules for this integration. An integration is considered enabled when its corresponding control record is in active status.

The data synchronization script is called in the following modes:

- When the entity is added, changed or deleted, the script is called in **Check** mode to determine whether a sync request should be created for the changed entity and a specific external system.
- When the sync request is processed, the script is called in **Process** mode to prepare and send the outbound message to the external system. Alternatively the script may request to discard the request or indicate that it should be processed later. The latter may be used when there is a need to wait until certain conditions are met before sending the message to the external system.

Note that the configuration for a maintenance object and a specific external system may be overridden by another configuration defined with a higher sequence number.

Confirming that a Sync is Needed

The base product provides a change data capture (MO Audit) algorithm **F1ONGDATASYN** that may be used by maintenance objects that need to be synchronized by this method. The algorithm consults the synchronization script associated with each active integration control record configured for the entity as to whether the external system needs to be notified for this change. Once confirmed, the algorithm creates a Sync Request record for the changed record, capturing its MO code and the primary key, along with the corresponding external system.

Note that the decision as to whether to create a sync request or not is controlled by the integration specific synchronization script while the creation of the sync request is the responsibility of the audit algorithm. As such, the same sync request BO is used for all integrations, with no provision for capturing integration specific details.

It is recommended to design the synchronization script to only approve the creation of a synchronization request when it is needed. This prevents unnecessary creation of sync requests that would otherwise be discarded when eventually processed.

Managing the Synchronization Process

Once a sync request is captured, there are several steps performed prior to any information being sent to the external system.

NOTE: This section only highlights key steps. Please refer to the business object configuration, its lifecycle and algorithms for a thorough picture of the full functionality..

- When a Sync Request record is created, its initial state (**Pending**) is configured to be processed by a batch monitor. That way, the changes that triggered the request are fully committed before being communicated to an external system.
- When the sync request record is processed, a monitor algorithm calls the synchronization script associated with the corresponding integration's control record to manage the synchronization process. The synchronization script may indicate either of the following:
 - The request is not yet ready to be processed, i.e. it should remain in its current status and get processed again by the next batch run.
 - The request should be canceled. This indication results in the transition of the record to the **Canceled** status.
 - The request was successfully processed. The synchronization script has prepared the outbound message and routed the information to the external system appropriately. This indication results in the transition of the record to the **Sent** status.

- If an error occurs while processing the request, a To Do Entry is raised to capture the error details and the record is set to the **Error** status. Retry processing of the sync request continues with each batch process until the request is either sent or canceled.

Communicating to the External System

Typically the communication with an external system is finalized when the message is successfully sent and successfully processed by the external system. Depending on the specific integration, the message may not be processed right away by the external system but rather queued and processed later. In this situation, the sync request may need to be further acknowledged as successfully processed (set to the **Acknowledged** status) upon an explicit response mechanism from the external system. The product supplies the web service **F1-UpdateAndTransitionSyncRequest** that may be used to process such acknowledgements.

Maintaining Sync Requests

The system provides a Sync Request portal that is used to view the in progress or completed sync request records.

The menu location of the portal depends on your specific edge product. It may be in a Data Synchronization menu or perhaps in the Batch menu. You are brought to a query portal with options for searching. The options may differ based on your specific product.

Once a sync request has been selected, you are brought to the maintenance portal to view and maintain the selected record.

An **Actions** zone may appear to display specific actions. Alternatively, the actions may be displayed directly in the display area of the **Sync Request** zone.

The **Sync Request** zone provides basic information about the sync request record.

Depending on your specific product additional zones may appear.

Operational Analytics

The framework provides several building blocks and tools that the edge applications may use to implement analytic reports directly from the operational application using Oracle Utilities Analytics Visualization product (referred to in this section as the analytics visualization product). The following sections provide more information about this functionality.

Understanding Operational Analytics

The following sections describe the type of configuration supported in your product to integrate with the analytics visualization product. Refer to the Oracle Utilities Analytics Visualization documentation for more information.

Direct Data Access

The analytics visualization product's canvases report directly off of the operational system thus eliminating the need of an ETL process. Selected tables and views have been designated as dimensions and facts for the purpose of generating the Start Schemas used by these canvases.

Calendar and Time Dimensions

Analytic reports rely on calendar and time dimensions to support various hierarchical grouping by date and time.

The Calendar dimension provides a level-based definition of the standard calendar and fiscal calendar in a flattened representation that is commonly used in data warehouses. This is necessary to enabling customers to group by calendar week, months, quarter and fiscal calendar period, quarter and fiscal year etc. Note that fiscal information about a specific date is optional and sourced from your edge application specific accounting calendar

In the same way, the Time dimension provides a level-based definition of each minute in a day, supporting reports that group by hours, AM/PM etc.

NOTE: Calendar dimension records are generated by a batch process; one record for each date in a specified period of time. You need to run the **Generate Calendar Dimension** batch process on an ongoing basis to cover the standard and accounting calendar days needed to support your business. Refer to **F1-BICDD** batch control for more information.

NOTE: Time dimension records are also generated by a batch process; one record for each minute in a day. You need to run the **Generate Time Dimension** batch process once to generate all the records. Refer to **F1-BITMD** batch control for more information.

Bucket Configuration

The analytics visualization product provides support for defining a set of ranges, each representing a bucket for which extracted measures can be grouped and classified under the relevant bucket. The framework product provides support for viewing and defining the buckets. Refer to [Bucket Configuration](#) for more information.

Characteristic Mapping

The product supports mapping of characteristics to user defined fields associated with dimensions in the analytics visualization product. Each characteristic table that is associated with a dimension table is provided with pre-generated user defined fields identified by unique column sequence numbers in the analytics visualization product. The mapping of characteristics to dimension user defined fields is maintained directly on the dimension portal. You may also review these mappings using the [Characteristic Mapping](#) portal.

Logical Dimension

The Element Name of a foreign key field to a dimension represents a distinct logical version of the dimension. This allows for different logical dimensions to be based on the same dimension table. For example, a “Location” dimension may be referenced as a “Main Location” on one fact and also as an “Alternate Location” on the same or other fact. While both elements refer to the same dimension as foreign keys, each represent a distinct logical dimension via their element names. In the same way, while each date field is implicitly considered as a foreign key to the Calendar dimension, each distinct element name establishes a unique logical dimension. For example, a “Start Date” element name and an “End Date” element name are distinct logical dimensions of the underlying Calendar dimension.

Some logical dimensions may represent an aggregated or summarized form of another logical dimension. They are also known as “shrunk” dimensions. For example, the “Calendar Month Dimension” view is an aggregated dimension of the Calendar dimension. Establishing a link between a summarized dimension and its detailed dimension allows the analytics visualization tool to drill from one fact to another for performance reasons when they are indirectly linked via the same logical dimension but at different aggregation levels. For example, “Fact A” represents daily records and “Fact B” aggregates the same measures at a monthly level. Associating the “month” field in “Fact B”, which is a foreign key to the “Calendar Month Dimension” view, with the logical dimension name of the “date” field in “fact A” allows analytic queries to use “Fact B” for queries at the monthly level or above which perform better than performing them on the more granular “Fact A”.

Analytics Configuration

Depending on which product or products you have installed, there may be some configuration needed for the analytics visualization product. Refer to [Defining Analytics Options](#) for more information.

Calendar and Time Dimensions

This portal is used to view generated calendar and time dimension records.

Refer to [Understanding Operational Analytics](#) for an overview of operational analytic functionality.

Navigate using **Admin > Analytics > Calendar and Time Dimensions**.

The following zones may appear as part of the portal's **Main** tab page:

- **Calendar Dimension.** This zone lists most recent calendar dimension records. You may filter the list to review records generated up to a specific date.
- **Calendar Date.** This zone provides information about a selected calendar date.
- **Time Dimension.** This zone lists all time dimension records.

Maintaining Bucket Configurations

Several key performance indicators in the analytics product look at measurement values (for example: the age of an asset or the age of debt) classified into a number of pre-defined groups also known as buckets. The overall metric can then be reported by the different buckets and allow various analyses.

For example, the age of an asset can be classified into the following buckets:

- Less than 6 Months
- 6-12 Months
- One Year and Older

The age of debt, also known as arrears can be classified onto the following buckets:

- 0-30 Days
- 30-60 Days
- 60-90 Days
- 90+ Days

The definition of the buckets is extracted to the Business Intelligence data warehouse, to be used as dimensions.

Bucket Definition Considerations

Each type of bucket is defined using a bucket configuration Business Object. The bucket definition considerations and/or rules will vary based on the bucket configuration business object used. The business objects available are driven by your specific product. For a list of available bucket configurations business objects, navigate to the business object page and view the business objects for the Bucket Configuration maintenance object.

Setting Up Bucket Configurations

To maintain the bucket ranges for the bucket configuration(s) applicable to your product, open **Admin > Analytics Configuration > Bucket Configuration**.

You are taken to the query portal where you can search for an existing bucket configuration. Once a record is selected, you are brought to the maintenance portal to view and maintain the selected record.

NOTE: Your specific product may also include an Analytics Configuration portal that displays the list of existing and potential bucket configuration records, allowing you to drill into this page to view the record in detail.

The **Bucket Configuration** zone provides basic information about the bucket configuration.

For more information about the elements supported refer to the zone's help or to the relevant analytics integration documentation for your product.

Defining Analytics Options

Refer to [Understanding Operational Analytics](#) for an overview of the operational analytics functionality.

Your specific product may require some system wide configuration to be defined when using operational analytics.

Navigate to the [Extendable Lookup](#) portal. Search for and select the **Analytics Options** business object.

Use the embedded help to review information about the options to define.

Defining Analytics Tables

This portal is used to search for Analytics Table definitions. An Analytics Table record exists for each table or view in the operational system that represent dimension and fact tables in the Operational Analytics product.

Refer to [Understanding Operational Analytics](#) for more information.

You can access the portal from the **Admin Menu > Analytics Configuration > Analytics Table**. You are brought to a query portal with options for searching for a specific record. Once a record has been selected you are brought to its maintenance portal to view and maintain the selected record.

Defining Analytics Dimensions

This portal is used to maintain metadata for Analytics Dimension tables. A dimension record exists for each table in the operational system that represents a dimension table in the Operational Analytics product.

Refer to [Understanding Operational Analytics](#) for more information.

You can access the portal from the **Admin Menu > Analytics Configuration > Analytics Table**. You are brought to a query portal with options for searching for a specific record. Once a dimension record has been selected you are brought to this portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page

- **Analytics Table.** This zone provides information about the selected dimension record.
- **Fields.** This zone lists all the fields that define the dimension table. Fields may be selected from the dimension's underlying operation table using the **Excluded Fields** zone or added as computed fields.
- **Dimensions.** This zone lists dimension tables that this dimension references via foreign key constraints.
- **Referenced By.** This zone lists dimension and fact tables that reference this dimension via foreign key constraints.
- **Characteristic Mapping.** This zone manages [characteristic mapping](#) extensions for the dimension.
- **Excluded Fields.** This zone lists fields from the dimension's operational table that are not included in the dimension table.

NOTE: The ability to introduce custom analytics dimensions is currently not supported.

Defining Analytics Facts

This portal is used to maintain metadata for Analytics Fact tables. A fact record exists for each table or view in the operational system that represents a dimension table in the Operational Analytics product.

Refer to [Understanding Operational Analytics](#) for more information.

You can access the portal from the **Admin Menu > Analytics Configuration > Analytics Table**. You are brought to a query portal with options for searching for a specific record. Once a fact record has been selected you are brought to this portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page

- **Analytics Table.** This zone provides information about the selected fact record.
- **Fields.** This zone lists all the fields that define the fact table. Fields may be selected from the fact's underlying operation table or view using the **Excluded Fields** zone or added as computed fields.
- **Dimensions.** This zone lists dimension tables that this fact references via foreign key constraints.
- **Excluded Fields.** This zone lists fields from the fact's operational table or view that are not included in the fact table.

NOTE: The ability to introduce custom analytics facts is currently not supported.

Defining Characteristic Mapping

This portal maps characteristic extensions to user defined fields provided for each characteristic table that is associated with an analytics dimension table.

These mappings may be directly maintained on the operational [Analytics Dimension](#) portal. Refer to [Understanding Operational Analytics](#) for more information.

If your system is integrated with the Oracle Utilities Analytics product refer to [Understanding Analytics Integration](#) for more information about ETL based characteristic mapping.

You can access the portal from the **Admin Menu > Analytics Configuration > Characteristic Mapping**. You are brought to a query portal with options for searching for a specific mapping record. Once a record has been selected you are brought to the this portal to review the information for the selected record.

The **Characteristic Mapping** zone on the portal's **Main** tab page provides information about the mapping record.

NOTE: Only **Adhoc** and **Pre-defined** characteristic types are supported.

Analytics Integration

The framework provides several building blocks and tools that the edge applications may use to implement integration with the Oracle Utilities Analytics product (referred to in this section as the analytics product). The following sections provide more information about this functionality.

Understanding Analytics Integration

The following sections describe the type of configuration supported in your product to integrate with the analytics product. Refer to the Oracle Utilities Analytics documentation for more information.

Master Configuration

Edge applications that include an integration to the analytics product typically include a master configuration record that captures information needed for the extract, load and transformation step, such as extract parameters. These records are provided by the specific edge products and may be viewed and maintained on the [master configuration](#) portal.

Note that your specific edge application may deliver an **Analytics Configuration** portal that displays the information from the master configuration record along with other analytics related configuration.

Bucket Configuration

The analytics product provides support for defining a set of ranges, each representing a bucket for which extracted measures can be grouped and classified under the relevant bucket. The framework product provides support for viewing and defining the buckets. Refer to [Bucket Configuration](#) for more information.

ETL Based Characteristic Mapping

The product provides objects to allow mapping configuration of characteristics in the product to user defined fields on dimensions in the analytics product. The mapping is used by the extract process to populate target user defined columns with corresponding characteristic data.

This type of mapping relies on configuration in the Allowed Target Dimensions [extendable lookup](#). The extendable lookup is used to define each Target Table in the analytics product that has one or more user defined fields that may be populated with characteristic values. It also defines the valid characteristic entities that may act as the source for the characteristic data.

NOTE: The framework does not provide any values in this lookup, but edge products that support mapping provide values in this lookup. Please refer to your specific product's integration chapter and refer to the Oracle Utilities Analytics documentation for more information.

Refer to [Defining Characteristic Mapping](#) for more information.

Change Data Capture Using Sync Request

Depending on the specific edge application and version you are using, there may be components of the integration that use Sync Request for the change data capture step. If that functionality applies to your implementation, the following points highlight how to get more information:

- Refer to the administration guide for Oracle Utilities Analytics to confirm if your product integration is using Sync Request for any change data capture functionality.
- Review the Sync Request Business Objects provided by your product for analytics integration.
- Refer to [Data Synchronization](#) for a high level understanding of the process.

Business Flags

It is possible that information detected in one product may be useful or even critical to share with another product. The framework provides functionality for receiving information from an external system that acts as a type of flag or alert that may need investigation. This allows any system to store detected business flags in a common way and share that information with one or more other systems.

Understanding Business Flags

The following is an example of a use case for business flags. Imagine that DataRaker highlights potential theft of service at a certain location. That product may initiate a business flag alert to various products owned by the implementation with a recognized "standard name" for the business flag, such as "TAMPER".

- If Oracle Utilities Meter Data Management receives this business flag, it may initiate a service investigation monitor.
- If Oracle Utilities Mobile Workflow Management receives this business flag, it may initiate a service investigation activity.
- If Oracle Utilities Customer Care and Billing receives this business flag, it may initiate a hold on billing for that location.

Note that the framework product supplies basic functionality to support logic that is common to all edge applications that implement business flag functionality. However it is the individual edge applications that supply more specific functionality (business objects and algorithms) for specific use cases, if applicable.

The following sections highlight functionality supported for business flags in the framework. Refer to the edge application product documentation for more details for supported use cases.

Standard Name

To ensure that Business Flags are universally understood across all edge applications and to simplify integration each Business Flag will have a Standard Name. This is a name that is used by all the products when sending information to each other. That way, if DataRaker product sends Oracle Utilities Meter Data Management a "TAMPER" business flag, it should result in the same functionality as when Utilities Customer Care and Billing sends a "TAMPER" business flag.

Business Flag Standard Names are defined using an extendable lookup. In addition to standard extendable lookup fields, the standard name also references a category. In addition, the lookup supports defining one or more external names, for cases where information is communicated from an external system that does not send the expected Standard Name.

The framework does not deliver any standard names or category values. Refer to your specific edge application products to verify if any standard names or categories are delivered. Your implementation may configure appropriate standard names and categories based on your business rules.

Business Flag Type Defines Behavior for a Standard Name

Although the definition of the business flag standard names should be universally understood by the various integrated products that support them, each individual product defines what should occur when a business flag with a given standard name is created. This is configured using a business flag type. Only one active business flag type may exist for a given standard name. Business flags that are received from an external system will define the standard name, but will not have knowledge of the specific business flag types defined. The business flag type is determined based on the standard name.

The business flag type defines the business object to use when creating the resulting business flag record. The business object defines the lifecycle of a resulting business flag record.

Business Flag Type Algorithms

The business flag type includes support for algorithms. This allows for an implementation to define a common business object that may be used for different business flag types (if a common lifecycle is followed) but allow for different functionality to kick in depending on the business flag type.

The product supplies a plug-in spot for **Additional Processing** that may be invoked by a business flag that enters the Additional Processing state. Refer to your product's library of business objects to determine if there is an Additional Processing state that supports calling algorithms on the business flag type.

Objects Linked to a Business Flag

There are two types of links between an object in the system and a given business flag.

Affected Entity

Each business flag is associated with a single record in the system that is considered the "affected entity" or the entity that the business flag is associated with. The affected entity is defined by the specific business objects designed for the use cases supported by your edge product. For example, many utility base products may configure service point as the affected entity for its business flag use cases. Each business flag created is linked to a specific service point. Linking a specific entity to each business flag allows for algorithms to trigger functionality for that entity such as an investigation order. In

addition, algorithms may be implemented in other business process areas that look for the existence of a business flag and act accordingly.

Related Objects

The business flag supports linking one or more related objects to a business flag to make it easier to trigger functionality or for impacted business processes to look for business flags. For example, when creating a business flag for a given service point, it may be useful to link all the accounts that are currently linked to the service point. Then, if an account oriented process should check for a business flag, it can look directly for a business flag linked to the account in its related object.

Impacted Business Process

The product supplies support for associating one or more impacted business processes to a business flag type. This configuration is used when functionality for that business process is impacted in some way based on the existence of a business flag of a given type. For example, maybe some process is put on hold when a certain type of business flag exists.

Note that configuring a business process on business flag type is not enough to trigger any impact on that business process when a business flag exists. There must also be some logic implemented in the business process functionality itself that knows to look for a business flag for a given record that is configured to impact the business process.

The definition of the business process is at the discretion of the edge application that supplies functionality to support this. For example, the business process could be defined as something broad such as “billing” or could be something more granular such as “billing estimation”. The system supplies an extendable lookup to use for configuring the supported business processes. Refer to the values of the business process extendable lookup in your edge application or to the edge application specific Business Flag documentation for more information about supported business processes.

Dates

A business flag supports two dates: a Business Flag Date/Time and a Business Flag **End** Date/Time

- The business flag date / time is required for all business flags. For some types of business flags only one date is needed.
- For business flags that have a start and end period, the business flag date/time acts as the start date and the other field is the end date.

For a business flag that has a date range, it may be important for functionality implemented for [impacted business processes](#). How the process treats the date will depend on its functionality.

- For some processes, the business flag is essentially expired after the end date has passed. This applies to impacted processes that are only looking at the current status of data in the system. For example, collection processing could be held if there is a business flag currently in effect (where the current date is within the date range). It would never look at historical business flags.
- For some processes where historical data may be relevant, a business flag effective during that same historical period may impact the process. For example if a business flag denotes an outage event for a given time period, perhaps estimated consumption should never be calculated for that time period.

Note that because business flags have a status, the design for the lifecycle of the business objects for the above effective dated use cases must carefully consider the states. For business flags that are considered expired after the end date passes, the BO lifecycle may be designed to transition to a final state after that date such that the record is no longer included in active processing. For business flags that continue to impact processing for a historic period, the BO lifecycle may be designed to remain in a non-final state such that it is clear that the record is still applicable.

Creating Business Flags

Business flags may be created in a system for one of the following reasons:

- A message is received from an external system that initiates the creation of a business flag. In this case, logic in the external system has detected some situation that this product is being alerted about.
- Business logic in this product detects a situation that should be investigated or should act as a flag. In this scenario, there may not be any integration needed depending on the business rules.
- Business logic in this product detects a situation that another integrated product should be alerted about. In this scenario, the business flag record is used to send out information to the integrated product.
- A user manually creates a business flag based on knowledge of the affected entity. For example, a customer service representative may create a business flag as a result of contact with the customer.

Creating a Business Flag from a Web Service

The system supplies both a REST and SOAP inbound web service that may be used for an external system to initiate (or update) a business flag. The REST service is **F1-BusinessInsights**. The SOAP service is **F1-BusinessFlagSync**. In both cases, the web service invokes a "driver" service script whose ultimate responsibility is to determine the appropriate Business Flag Type based on the standard name or external standard name, and therefore the appropriate business object for the new business flag. Because different products may have different logic related to creating a business flag, the service script calls another service script linked to the maintenance object using the Business Flag Sync MO option.

The "driver" service script includes a "contextData" group, which in turn references the data area **F1-BusinessFlagContext**. This data area is to allow for an external system to provide additional information for a given business flag use case. For any integration that requires additional information to be passed in from the external system, a separate data area should be used to define the additional elements; and that data area should refer to **F1-BusinessFlagContext** as its extended data area.

The framework does not supply a Business Flag Sync service script, however individual edge applications supply a service script based on the use cases it supports out of the box.

NOTE: For products that are continuing to use XAI for external messages, the product also includes an XAI inbound service for the same Business Flag Sync Driver service script. Note that the product recommendation is to discontinue use of XAI and use [inbound web services](#) instead.

Error Handling

If there is a problem in trying to create a business flag based on incoming information, the Business Flag Sync Driver service script creates a special business flag record using the Business Flag Error Business Object. This is also configured on the maintenance object as an option. The framework product supplies the business object Business Flag Error (**F1-BusinessFlagError**) for this functionality. Refer to the business object description and configuration for more information.

Confidence

There may be use cases where a condition is suspected, but not confirmed. The originating system should be able to assign a "confidence" level to the business flag.

For example, Oracle Utilities Analytics Insights or other machine learning platforms will aggregate and analyze large amounts of data to identify potential insights that need investigation. These platforms can initiate a business flag with a **Suspected** confidence.

An application receiving this business flag may adjust the confidence to either **Confirmed** or **Rejected**.

- Setting the confidence to **Confirmed** is the equivalent of a true positive.
- Setting the confidence to **Rejected** is the equivalent of a false positive.

Note that the application that receives a business flag is responsible for acting on the value based on business rules. Any updates to the confidence can be communicated to the other products that received the same business flag, if desired.

Additionally, creating a business flag in the application with a confidence of **Missed** provides a way to denote a missed insight. This is the equivalent of a false negative.

Because a utility implementation may have multiple applications installed that support business flags, the following guidelines are suggested for designing where the confidence flag should be updated.

- If Oracle Utilities Service Order Management is implemented, it has the responsibility of updating the confidence flag and communicating the update to other products.
- Otherwise, the assumption is that Oracle Utilities Customer Care and Billing owns field work orchestration and that it will have the responsibility for updating the confidence flag and communicating the update to other products.

No product logic is provided to enforce the above suggestions, however, the business objects supplied by the different edge applications will support the recommended implementation.

Business Flag Updates from External System

When the product that is responsible for updating the Confidence flag makes a change, it should initiate an outbound message to alert other products. On the receiving side, the same inbound web service and Business Flag Sync service script is responsible for the update. Refer to [Creating Business Flags](#) for more information.

Setting Up Business Flag Configuration

The following topics highlight the general configuration steps required to use business flag functionality. Your specific product may supply additional functionality to support specific use cases for business flags. Refer to your specific product's documentation and the library of business objects supplied for Business Flag in your product for more information.

Standard Name Category Characteristic Type

Define one or more categories for grouping your standard names into logical business groupings.

Navigate to the [Characteristic Type](#) page. Search for and select the Business Flag Category characteristic type (**F1–BUSFC**).

Define desired category values and descriptions to be used for the standard names.

Business Flag Standard Name Lookup

Navigate to the [Extendable Lookup](#) portal. Search for and select the **Business Flag Standard Name** business object.

Define values that are recognized in the external systems that your implementation is receiving business flag details from.

Define a **Category** for the standard name that is appropriate for your product. Note that the category does not have to be in sync with standard name definitions in external products.

Refer to the embedded help for more information about configuring this object.

Business Process Lookup

If your specific product supports configuring business processes that may be impacted by the existence of a business flag, they are defined as an extendable lookup.

Navigate to the [Extendable Lookup](#) portal. Search for and select the **Business Process** business object.

Integration Configuration

The following points highlight configuration required to support receiving business flag information from an external source:

- Define a record for each [External System](#) that the product may be receiving business flag records from. This should be a value known by the external system and provided when new business flags are sent to this product.

When this product should initiate business flag information to be sent to an external system, configure one or more [Outbound Message Type](#) records. For each one, update the External System to configure how each outbound message type is communicated to the external system.

Defining Business Flag Types

Refer to [About Business Flags](#) for an overview of business flag functionality.

To maintain the business flag types applicable to your product, open **Admin > Integration > Business Flag Type**.

This is a standard [All-in-One portal](#).

The information captured on the business flag type depends on the business objects supported by your product. Refer to the embedded help text for more information.

Maintaining Business Flags

This section describes the functionality supported for viewing and maintaining business flags.

Refer to [About Business Flags](#) for an overview of business flag functionality.

Navigate using **Main > Integration > Business Flags**. You are brought to a query portal with options for searching for business flags.

Once a business flag record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Business Flag** zone provides basic information about a business flag. Refer to the embedded help for more information.

Market Transaction Management

This section describes various entities and aspects involved in the configuration and maintenance of market transactions. The framework product supplies basic functionality to support the objects related to market transaction management. However more specific functionality (business objects and algorithms) for specific use cases are supplied by the edge applications.

Understanding Market Transaction Management

Several objects are provided to support market transaction management.

- **Market Message Type.** This object is used to define the different types of market messages, both inbound and outbound and to define the appropriate market message Business Object to use when creating objects of this type.
- **Inbound Market Message.** This object captures the instance of a market message received from an external source.
- **Outbound Market Message.** This object captures the instance of a market message that will initiate a message to an external source.
- **Market Process Type.** This object is used to define configuration related to processes that may be initiated to support steps that should be execute to support an inbound or an outbound market message. It defines the market process Business Object to use when creating market processes of this type. It may also define one or more market process event business objects to initiate events that are generated to support steps within the market process.

- **Market Process.** This object captures the instance of a market process that may be created to carry out business logic related to a specific inbound or an outbound market message.
- **Market Process Event.** This object is used to orchestrate individual events that may occur to support steps in a given market process.
- **Market Configuration.** This object may be used to capture configuration needed to support market transactions. Each type of configuration should use a Business Object to define what information needs to be captured. Each type of configuration may define up to five key fields that may be used to uniquely identify an individual configuration instance.

Configuring Market Transaction Management

The following sections describe market transaction management configuration options.

Defining Market Configurations

Each market configuration is used to define one or more options for a given type of configuration (based on a delivered business object).

The product defines a generic set of portals, described here, that may be used to define any type of market configuration. However, it may be that your implementation includes more specific configuration portals for certain types of market configurations.

Open this page using **Admin > Market Transaction Management > Market Configuration**.

You are brought to the **Market Configuration Query** where you need to search for the market configuration (i.e., its business object).

Once you have found the appropriate market configuration, select the value and you are brought to a standard [All-in-One](#) portal that lists the existing values for the market configuration. The standard actions for an All-in-One portal are available here.

Defining Market Message Types

Refer to [Understanding Market Transaction Management](#) for an overview of market messaging functionality.

Open this page using **Admin > Market Transaction Management > Market Message Type**.

This is a standard [All-in-One](#) portal.

The information captured on the market message type depends on the business objects supported by your product. Refer to the embedded help text for more information.

Defining Market Process Types

This page is used to define market process types which control behavior of the resulting market processes and the valid market process events that could be triggered as a market process progresses through its lifecycle.

Refer to [Understanding Market Transaction Management](#) for an overview of market messaging functionality.

Open this page using **Admin > Market Transaction Management > Market Process Type**.

This is a standard [All-in-One](#) portal.

The information captured on the market process type depends on the business objects supported by your product. Refer to the embedded help text for more information.

Maintaining Market Transactions

The following sections describe various pages available to view and work on market transactions.

Maintaining Market Processes

Refer to [Understanding Market Transaction Management](#) for an overview of market messaging functionality.

Open this page using **Main > Market Transaction Management > Market Process**. You are brought to a query portal with options for searching for market processes.

Once a market process record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Market Process** zone provides basic information about a market process. Refer to the embedded help for more information.

Related Objects

The related object tab includes several zones that display various objects that may be related to the market process.

Related Market Messages and Processes

This zone displays any entries in the related object collection that are outbound market messages, inbound market messages or market processes.

Related Processes

This zone displays market processes that are linked to this market process using the explicit related process table. The market processes linked here are marked either as a parent, a child or a sibling market process.

Other Related Objects

This zone displays market process events linked to this market process. In addition, the zone displays any miscellaneous objects that are linked to the market process in the related objects collection, besides outbound market messages, inbound market messages and market processes.

Maintaining Market Process Events

Refer to [Understanding Market Transaction Management](#) for an overview of market messaging functionality.

You can view this page when drilling into a market process event from the [Market Process](#) related objects tab.

Once a market process record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Market Process Event** zone provides basic information about a market process. Refer to the embedded help for more information.

Related Objects

The related object tab includes a zone that displays all entries in the market process event's related object collection.

Maintaining Market Messages

Refer to [Understanding Market Transaction Management](#) for an overview of market messaging functionality.

Open this page using **Main > Market Transaction Management > Market Message**. You are brought to a query portal with options for searching for market messages. Both inbound market messages and outbound market messages may be queried here.

Once a market message record has been selected, you are brought to the appropriate maintenance portal (inbound market message or outbound market message) to view and maintain the selected record. The inbound market message portal and the outbound market message portal have similar portal tabs and zones.

The **Market Message** zone provides basic information about the market message. Refer to the embedded help for more information.

Related Objects

The related object tab includes several zones that display various objects that may be related to the market process.

Related Market Messages and Processes

This zone displays any entries in the related object collection that are outbound market messages, inbound market messages or market processes.

Other Related Objects

This zone displays any miscellaneous objects that are linked to the inbound / outbound market message in the related objects collection, besides outbound market messages, inbound market messages and market processes.

Content Migration Assistant (CMA)

This chapter describes the Content Migration Assistant (CMA), a facility to enable the export of data from one environment to another.

CAUTION: This chapter is intended for users responsible for creating environments for testing.

Understanding CMA

The Content Migration Assistant (CMA) provides implementers with a flexible, extensible facility for migrating 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 following points highlight some common use cases for using this tool:

- Seeding a newly created environment with configuration data. For example, if you are creating test environments, you can import configuration data from a development environment. In this case, you will probably set up a "copy all configuration" type of migration. Another use case is if you want to troubleshoot an issue in production, you may want to copy all production configuration to a test environment.
- Updating an existing test environment with new configuration or updates to configuration from a development environment. In this case your set of objects will be more targeted and you are doing more of an incremental migration.
- Promoting tested configuration to your production environment.
- Migrating selected master and transaction entities from one environment to another for testing purposes.

Here are key aspects of the import process that apply to all cases:

- When records exist in both the target and the source, the CMA tool will fully **replace** the object with the version from the source. Special consideration should therefore be given to records whose [primary key is system generated](#) to ensure the correct entity is being replaced.
- While typically the source version of an object is applied directly onto the target environment, CMA does provide a way to [Adjust Imported Data](#).
- The import process **does not handle deletion** of records in the target environment. When importing records from a source environment to a target environment, 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 orchestrate 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.

Configuration steps are mainly used to define the data to migrate at the source system but also involve tasks to be performed at the target environment. 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. At a high level, configuration tasks involve the following entities:

- System wide settings are defined in the **Migration Assistant Configuration** master configuration record.
- 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.
- A **Migration Request** is used to define the data to include in a given migration export.

Refer to [CMA Configuration](#) for more information.

Once you have a migration request defined to include desired data to export, the next steps are to export from the current environment and import to the target.

- The **Export** process includes all the steps needed to select records to be exported from the source environment and create the export file. 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 [Additional Migration Considerations](#) 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. For more information, refer to [Importing and Applying a Migration](#).

The topics in this section provide a general understanding of the Content Migration Assistant (CMA) tool.

Types of Migrations

The Content Migration Assistant is used for two general types of migrations: wholesale and targeted. The following topics 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. 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.

With the ability to group migration requests, the expectation is that implementations follow these guidelines:

- Multiple migration requests using the Criteria-based or Entity List migration request classes are used to group information logically together to allow for more reuse.
- A Group migration request is used for the export. This allows for one data set to export and one data set on the import side, simplifying the process. Note that depending on the amount of data, this may be a large import set to process. An implementation may find it easier to create multiple migration requests that break down the process into several steps.

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 can build a Group migration request that includes the base migration request and your custom migration requests to have a consolidated export.

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.

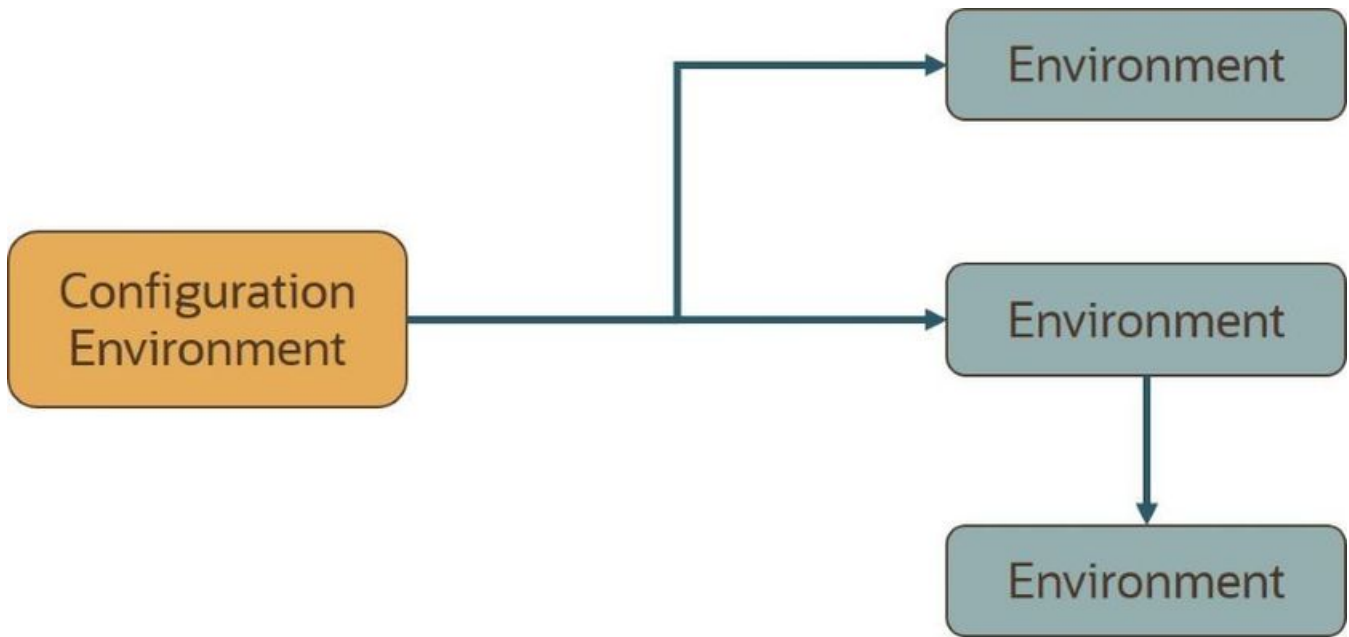
Targeted Migrations

A targeted migration refers to migrating a specific subset of data from one environment to another. Migration plans used in targeted migrations are designed to be self-sufficient and include all necessary subordinate instructions needed to ensure that the exported entity can be imported without encountering referential integrity errors. With this type of migrations, any relationship explicitly excluded from the migration plan assumes the related entity already exists in the target environment. Examples of targeted migrations include:

- Migration of a new portal, its zones, and its application service.
- Migration of all outbound message types.
- Migration of selected accounts and their related master and transactional entities to a lower environment for testing purposes.

Migrating Configuration Data

A main use of CMA is to migrate configuration data, where the source environment is considered a kind of system of record and the target environment is getting updates based on changes to the source environment. Alternatively you may be seeding a new environment. The expectation is that the target record is updated to look like the source record.

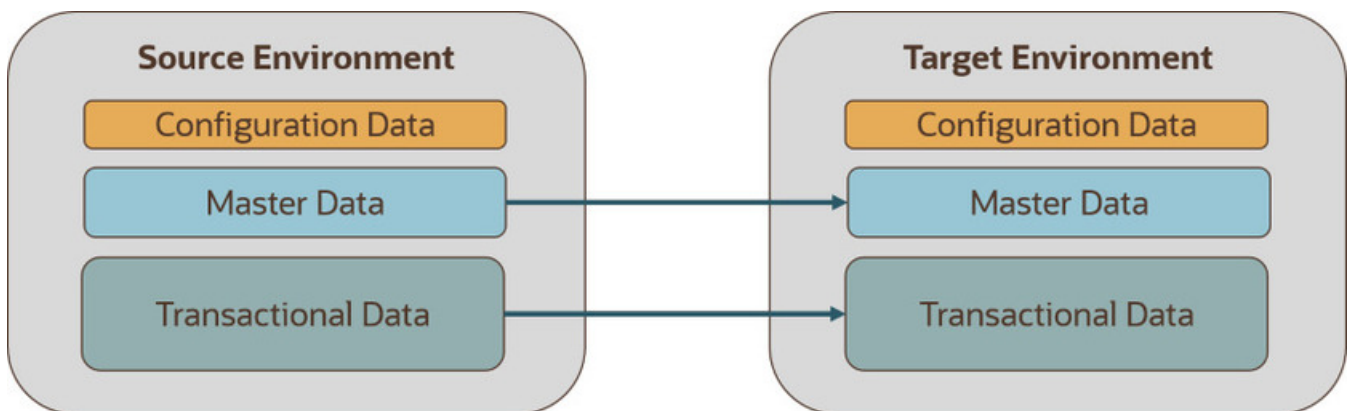


Configuration data may include a mixture of records that have an owner flag and ones that do not. The comparison step of the import process will generate appropriate insert or update SQL statements for the following data found in the export:

- 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.

Migrating Business Data

CMA may be used to perform a targeted migration of selected master entities and their related transactional data from one environment to another. For example, migrating a subset of accounts and their related data for testing purposes.



The following points highlight key differences between business data and configuration data that call for special considerations when designing your business data migrations:

- Migration of business data typically involves a much higher number of records in a data set compared with a configuration-only data set. While there is no explicit limit to the size of a data set, some reasonable limit has to be assumed.
- Having large sets of entities depend on relatively few entities in the same data set significantly slows the import process. This situation occurs when mixing configuration data, master data and high volume transaction data in a single migration. We therefore recommend designing separate migrations for these data classes. Refer to the **Reduce Dependency Between Migration Objects** section below for more information.
- All business entities have a system generated key. Refer to [Data with System Generated Primary Keys](#) for more information.

The following sections expand on topics and concepts related to business data migration.

Reduce Dependency Between Migration Objects

CMA leverages the application's referential integrity and business rules to ensure entities remain compliant and valid when imported into another environment. These rules are there to prevent an entity from being imported when any of the entities it references do not exist in the target environment. When both the referencing entity and the entity it references are part of the same data set they establish a **dependency** that dictates the sequence in which these entities should be processed in order to be successfully imported. Note that the tool supports even the more complex and rare situations wherein the dependency between entities is cyclical, i.e. entity A (directly or indirectly) refers to entity B while entity B (directly or indirectly) refers back to entity A, for which sequential processing is not enough to guarantee a successful import.

The tool uses foreign key relationships between entities in the same data set to identify sets of dependent entities and group them into separate migration transaction records. Note that these foreign keys are defined in the application's [data model](#) repository and not at the database level for greater flexibility. The larger the migration transaction the more complex the process is to successfully import all its objects. So while the tool is designed to support all types of dependencies (sequential and cyclical), handling very large sets of dependent objects may incur a significant performance cost.

While configuration entities tend to be highly inter-dependent, the total number of entities that comprise the entire set of configuration objects is relatively small. Therefore, the impact on performance such dependencies have when importing a configuration-only data set is negligible. On the other hand, configuration entities tend to be referenced on many master and transaction entities as they control many aspects of the latter's business rules. As such, mixing configuration and business data in the same data set is likely to form extremely large sets of business entities that dependent on a small number of configuration entities. It is therefore strongly recommended to import or set up configuration data in the target environment prior to importing business data into it.

In a similar way, some transactional data tends to be of a much higher volume compared to the master data entities they reference. Here too, it is strongly recommend to migrate master data entities prior to their high volume related transactional data. For example, many interval measurement records (transaction data) reference the same measuring component (master data). Migrating a large volume of measurement data along with their measurement component would result in a very large migration transaction that may take long to import. Importing the measurement components prior to migrating their interval measurements will eliminate unnecessary dependencies within the measurements only data set and significantly improve the performance of the overall process.

Reasonable Data Volume

The overall volume of all business entities to migrate in a single data set should be reasonably sized. For example, importing several hundreds of accounts and their related master and transactional data is considered a reasonable size. Migrating too much data may reach physical and performance limitations of the tool.

You may use any supported migration request methods for describing the entities to export. If you opt for an Entity List migration request, consider using the [Collect Entity](#) dashboard zone to build the list of entities as you browse them on their respective portals.

Single Source of Data

CMA uses an entity's prime key to determine if it is new to the target environment and therefore should be added or refers to an existing record and therefore needs to be replaced with the new version. All business entities have system generated keys, which are environment specific. It is therefore possible to have different entities in separate environments having the same prime key. When migrating entities whose keys are system generated, it is strongly recommended to migrate data to a target environment from a single source environment and avoid creating such entities in the target environment using the application. This practice ensures that prime keys for an imported entity are always in sync with their source environment. Refer to [Data with System Generated Primary Keys](#) for more information and considerations.

No Deletion

CMA does not handle deletion of entities of any kind, whether they are configuration or business entities. If a test needs to be repeated using an initial snapshot of the data, you should restore your target environment to a backup taken before test data was imported and import the latest version of the test data. If there is no need to delete previously imported entities, you may keep reloading test data from a single source environment as needed.

Data with System Generated Primary Keys

For records with system generated keys, there is an issue if the target environment already has a record with the same key but doesn't represent the same record. The tool shouldn't be just updating that target record with the view of the source in case this will break foreign key relationships that exist in the target.

The tool provides support for administrative data with system-generated primary keys. The logic relies on the maintenance object to use a method that looks at other attributes of the record (considered a "logical key") to detect whether the record being migrated already exists in the target region or not. The examples in this section are based on the Attachment maintenance object. Common attachments are considered administrative data. The attachment MO uses the file name and the creation date as the "logical key".

Imagine a common attachment for the "standard rate codes" file exists in a source region with the key 123456789. The table below highlights possible situations at the target region and actions supported in CMA.

Scenario	Target Situation	Action	Comments
1	No matching record	Record can be added with key 123456789.	
2	Record exists with key 123456789 and logic confirms that it is also the "standard rate codes" attachment.	Record can be updated.	
3	Record exists with key 123456789, but logic detects that it is not the "standard rate codes" attachment.	Record is not updated. An error is issued.	The system cannot update this record because it's not the right attachment record.
4	The system detects that another attachment record exists for the "standard rate codes" attachment with a different ID.	Record is not updated. An error is issued.	Assumption is that the record was created directly in the target or was copied from a different source.

The use cases described in scenarios 3 and 4 above would require key mapping to keep track of the id from the source to the id in the target so that any other records from the source that reference this key as a foreign key would be updated as part of the migration. This functionality is not supported.

Scenarios 1 and 2 above are supported for maintenance objects that use the method to detect the logical key.

NOTE: If a maintenance object with a system generated key does not supply a method to detect the logical key, CMA will update an existing record with the same ID. For maintenance objects in the framework that provide this method, refer to [Framework Provided Migration Configuration](#). For your specific edge application, refer to the CMA addendum for information about support for data with system generated keys.

The product recommends that an implementation establishes a migration strategy such that records with system generated keys are always created in the same region and always follow a standard migration path for promoting the data from this

source region to other regions. Following this strategy, you would minimize or eliminate the possibility that a record for the same logical key is created in multiple places such that different IDs would be generated as described by scenario 4 above.

MOs with a Mixture of Administration and Non-Administration Data

There are some MOs that contain a mixture of master or transaction data and administrative data. The Attachment is an example of this. The product supports common attachments and owned attachments. Owned attachments are records that are specific to its owner. The owner could be master or transaction data and its attachments are therefore considered master or transaction data. Owned attachments are not candidates for migration using CMA. Common attachments on the other hand are considered administrative data and may be candidates for migration using CMA. For these use cases, an implementation may follow the suggested strategy of only creating the administrative data in one region so that IDs for common attachments are not reused. However, it is reasonable and expected that owned attachments are being created in the target region and may receive a system generated key that matches the key of a common attachment from the source region.

To try to minimize this issue, the system includes special logic to be used by any MO that may contain administrative data mixed in with master or transaction data. This special logic generates the key of an administrative record with a zero (0) in the middle of the key and ensures that the keys for master and transaction data do not include a zero in this spot. For maintenance objects in the framework that use this method, refer to [Framework Provided Migration Configuration](#). For your specific edge application, refer to the CMA addendum for information about additional maintenance objects that may be in this category.

Importing Data that References a User

Although the tool supports copying users from one environment to another, the assumption is that it will be more common for implementations to define their valid users for each environment separately. In fact, the default CMA configuration for copying security configuration assumes that users are not being copied.

There may be times were a record is being copied from one environment to another where a user is referenced on the record but that user is not valid in the target environment. For example, if a batch control is configured as Timed with a user referenced, if this batch control is copied to another environment, what should happen if that user is not found? You may not want that user defined in the target environment. To handle this, the product provides support for a Default User that you can define in the [migration assistant configuration](#). If this default user is configured then any time the CMA import process recognizes that a record references a User and the user does not exist in the target environment, it updates the record to instead refer to the default user.

NOTE: Please note that if you are trying to copy entities that includes valid users, you should ensure that all the users are copied to the target environment first. Otherwise, the entity will be updated to refer to the default user, which may not be your intention.

NOTE: If you copy records that reference a User within an XML field or CLOB, the CMA input tool is not able to detect that. A pre-compare algorithm is recommended if this use case exists in your implementation.

Additional Considerations

The following sections describe some additional considerations for CMA.

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.

Multi-Language Environment Considerations

If your implementation uses a language other than English, it means that migrated administration objects may have multiple language rows (because English is always enabled). There are some important points with respect to multiple languages and CMA:

- As described in [User Language](#), there are steps to follow when supporting an additional language. The steps outlined in that topic highlight that for system data, translation of the strings may be provided via a language pack provided by the product or may be the responsibility of your implementation. In either case, this effort is non-trivial and will have its own established plan. The expectation is that the translation of the system data is applied for each region at your implementation site. CMA should not be used to create a new language in a target region.
- For administrative / control data that your implementation develops as part of your project, the expectation is that descriptions for your supported language are entered in the region that is considered the source region used to promote changes to regions in the "chain". For example, control data is entered in a development region and promoted to a test with the supported language enabled in both regions.
- What if you export data from a region with more languages enabled than your target? This scenario is perhaps a case where the source region is a type of test or playpen region where the additional language is enabled for other purposes. In this case, if the language code doesn't exist at all in the target region, the import will produce an error given that the code is invalid. If the language code exists but is not enabled, this will cause the extra language rows to be inserted in the target region, but will not cause any issues. They are simply ignored.
- What if you export data from a region with fewer languages enabled than your target? In this situation, the import process will only create language rows for the languages that were copied from the source. It will not automatically create language rows in the target as part of the import. For this situation, the recommendation is to run the **New Language** batch program (**F1-LANG**) that creates any missing language entries.

CMA Configuration

The following sections describe tasks required for CMA configuration.

Master Configuration - Migration Assistant

The **Migration Assistant Configuration** [master configuration](#) record defines system wide configuration used in both the source environment and the target environment.

As described in [Importing Data that References a User](#), if you have records being imported that may reference a user does not exist (and shouldn't exist) in the target environment, define a Default User here.

In both the source environment and the target environment, the system needs to know the location of the export directory and the import directory along with the expected file suffix. For these fields, implementations may also rely on the system defaults. If no Migration Assistant Configuration record is found, the system assumes that there is an entry defined in the system's substitution variable list for **F1_CMA_FILES**. Further it defaults the values as follows:

- Export directory is the value for this variable plus "**\export**".
- Import directory is the value of this variable plus "**\import**".
- File suffix is set to **cma**

Refer to [Referencing URIs](#) for more information about the substitution variable list.

For additional information about specific fields in the master configuration, refer to the embedded 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 migration plan is always related to a given maintenance object, defined using its primary instruction. Multiple migration plans may exist for a given maintenance object if there are different requirements for migrating records in that MO under different circumstances. For each migration plan, you decide if additional data should be included as part of an object using subordinate instructions. For example, when copying a single algorithm type, should the system only copy that algorithm type record, or should it also copy all its algorithms and its related script (if applicable)? Refer to [Types of Migrations](#) for more information.

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 > Implementation Tools > 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 an XML column to be referenced.

NOTE: The #parent expressions may access elements that are stored in an XML column 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
Pre-Compare	Optional	Algorithms of this type may be used to adjust the data after it is moved to the target system. These may only be defined on the primary instruction. Refer to Adjusting Imported Data for more information.
Import	Optional	Algorithms of this type are no longer supported.

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 XML-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. Defining instructions ensures that the related objects are included in the migration, if appropriate. If you know that the related objects are included because of other criteria, 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 purpose.

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 used to define the data to be included in a migration. To view or define a migration request, navigate using **Admin > Implementation Tools > 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.

There are three types or classes of migration request. The system provides a base business object for each along with a migration request class, which matches the business object. The subsequent sections provide more information about each class of migration request.

Note that all migration requests support defining a Category, which allows implementers to categorize the migration request.

In addition, all classes of migration request include the following zones:

- **Migration Request** This zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.
- **Referencing Migration Requests** This zone is only visible if the displayed migration request is included in a Group migration request. It lists each group migration request that includes it.

Other zones may appear for specific classes of migration requests. See the following sections for more information.

Criteria-based

This type of migration request defines a set of migration plans to be logically included together as part of a migration. For each migration plan, selection criteria is defined to indicate which records for each MO should be included. Selection may be defined using SQL, an algorithm or explicit primary key values.

- For selection using SQL Statement, refer to the embedded help for examples.
- For selection using Algorithm, the algorithms that are eligible to be plugged in here are those valid for the **Migration Request - Select** system event.
- For selection using Specific Keys, the primary key (1 through 5) must be explicitly specified. Multiple rows are allowed.

Entity List

This type of migration request allows the user to choose explicit MO / prime keys. The MOs that are eligible are those that are configured with a **Default Migration Plan** option. Although the user is managing MO / PKs, the migration instructions are still defined with a migration plan. The system maps the migration instructions in a similar way to a **Criteria-based** migration requests that use a Specific Key selection type. Note however that it will create a separate migration instruction for each MO / PK combination. It does not try to group all PKs for the same MO / migration under one migration instruction.

For this type of migration request, a user adds a migration request record with its description and other main information. Then, a special zone **Add Entities** is provided to find and select records based on a selected maintenance object and add to the migration request. The user is prompted to provide a reference and comments, if desired. If the category selected is one that requires a reference, then this will be validated. Alternatively, you may also use the [Collect Entity](#) dashboard zone to add specific entities as you review them on their respective maintenance portals.

When maintaining a migration request with existing entities, they are visible in the zone **Migration Request Entities** . This zone allows a user to remove the entity from the list.

Group

This type of migration request points to other migration requests. This allows you to define separate migration requests that represent logical groupings of migration plan instructions for ease of maintenance, but to combine all the separate migration requests into a single "grouped" migration request for streamlined export / import purposes.

The CMA export process will build an extract that includes the union of all the objects that qualify for the export and group them together based on their relationships.

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 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, a To Do Type includes a collection of valid To Do Roles and in turn the To Do Role refers to its To Do Types. If an implementation wants to copy a single To Do and include all its related information, including its To Do Roles, it does not want the migration of each To Do Role to in turn copy all its To Do 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 Content 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. Also be sure that this FK reference defines an Information program.
- 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 to [Characteristic Type](#) 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.

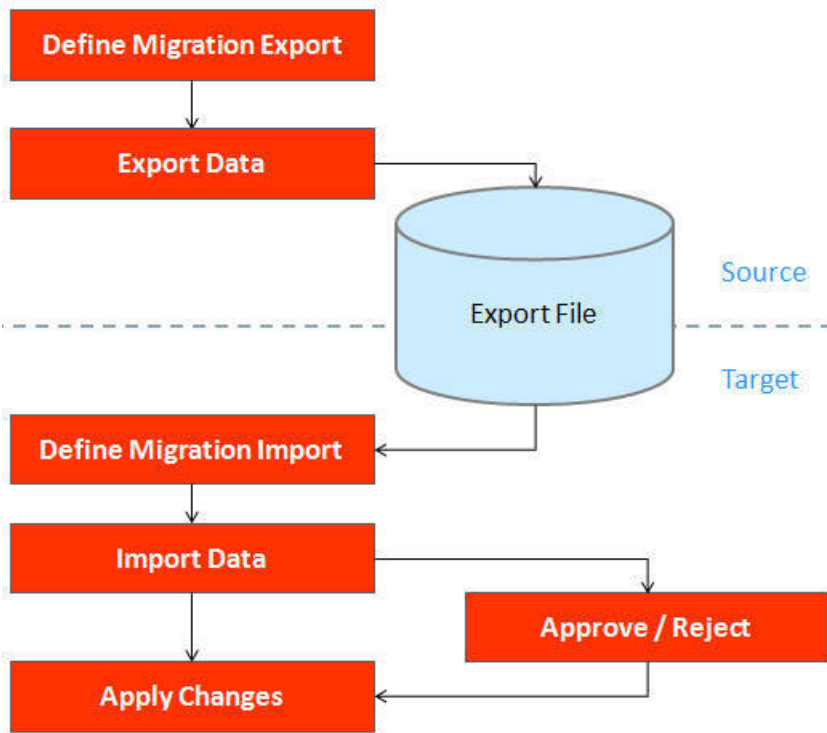
In order to support **Entity List** migration request, a default migration plan must be defined as an option on the maintenance object. This should be a single migration plan that supports all types of business objects for the MO.

If your implementation has a template migration request to use for 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.

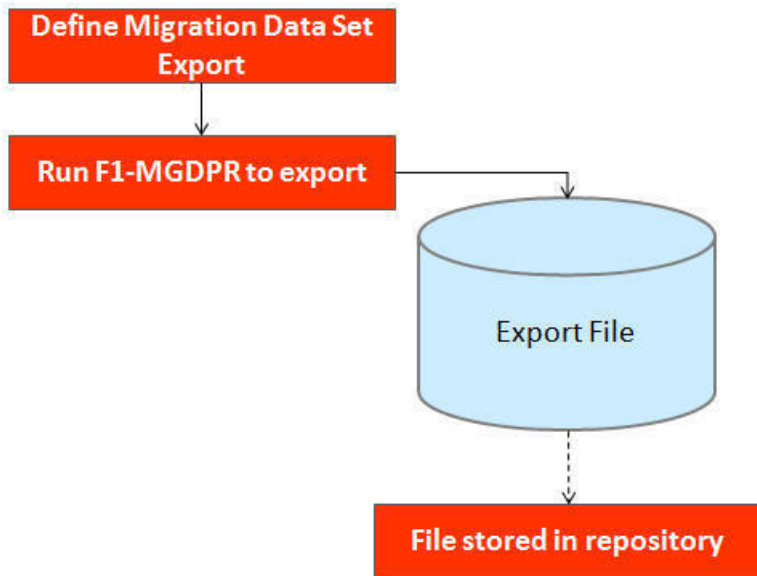
The CMA Execution Process

The following diagram illustrates a high-level view of the Content 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 Export Monitor** batch job can be submitted generate the export file. 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.



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.

To view an existing migration data set export, navigate using **Admin > Implementation Tools > 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**.

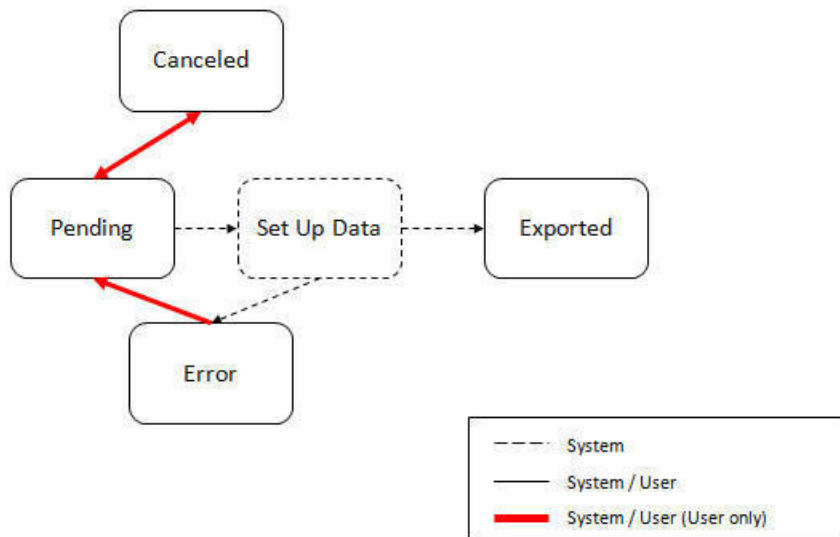
Enter a unique **File Name** for the export. Do not use spaces in the name, and do not enter the file extension or a path. The output location and file extension of the intended export file, which should appear in the **Export Directory** and **File Suffix** labels, are defined as described in the topic [Migration Assistant Configuration](#).

Enter an **Export Description** to provide information about the purpose of the export. Note that this field is not language enabled.

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.

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 content 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. Users may choose to suppress individual SQL statements for a given object that is approved. 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. Optionally, a migration import may be configured to automatically apply.
- **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 as described in the following topic [Master Configuration - Migration Assistant](#) 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.

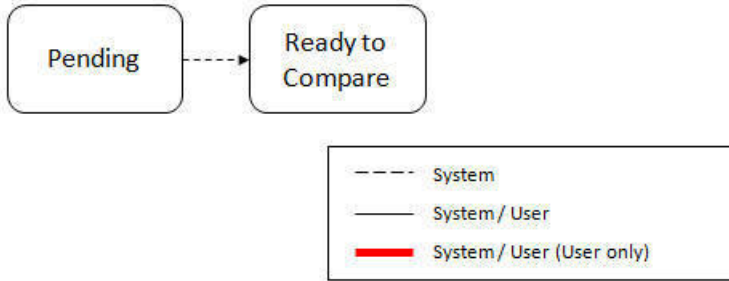
- The **Default Status for Add** sets the default status for objects that are determined to be *new* during the import comparison process. The default is to automatically set new objects to **Approved** status. Other options are to set any new objects to **Rejected** or **Needs Review** status.

- 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 user may also configure the **Automatically Apply** flag to **Yes**. This allows for use cases where the migration is repetitive and has been tested and the user feels that there is no need for manual approval. Note that when configuring this setting, neither of the Default Status values may be set to **Needs Review** and at least one must be set to **Approved**.

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 is created, the next step is for the system 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.



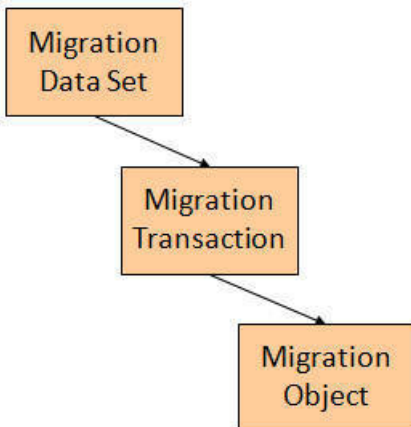
The following points describe the lifecycle.

- 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](#) and [Running Batch Jobs](#) 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

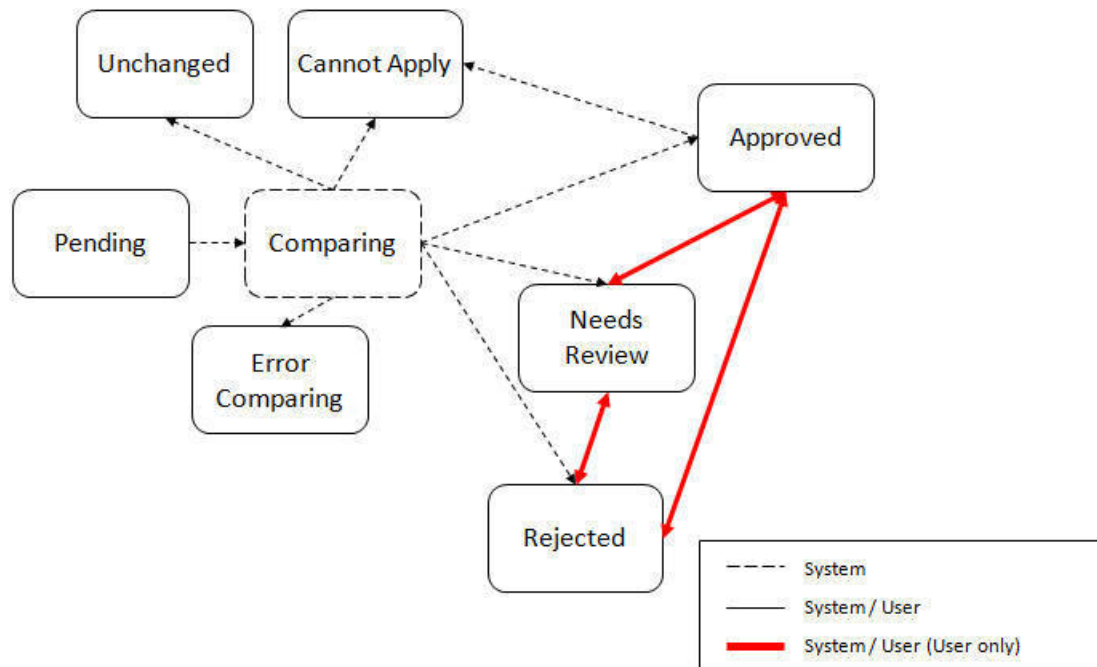
The import step results in the creation of one or more migration objects, one for each record selected in the export based on the export's migration request and its configuration. Related objects are grouped together in migration transactions. The next step in the import process is the Comparison step. 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.

NOTE: Refer to [Running Batch Jobs](#) for more information about streamlining the various steps in the process.

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.



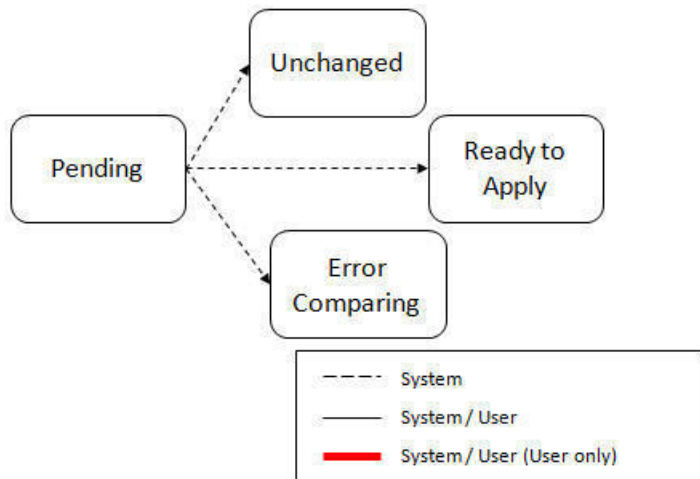
The following points describe the lifecycle.

- When **Pending** records are selected by the monitor batch job, it transitions to **Comparing**. If the migration object refers to one or more pre-compare algorithms, they are executed to [adjust the data prior to comparison](#). Then algorithm will determine the appropriate next state by comparing the source data to the target data.
- 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** (with the object action value also set to **Unchanged**).
- If the record in the migration object is found in the target environment and the data is different, the algorithm sets the object action value to **Change** and generates the appropriate SQL to be used later in the Apply step to update the record. It 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 sets the object action value to **Add** and generates the appropriate SQL to be used later in the Apply step to insert the record. It 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.

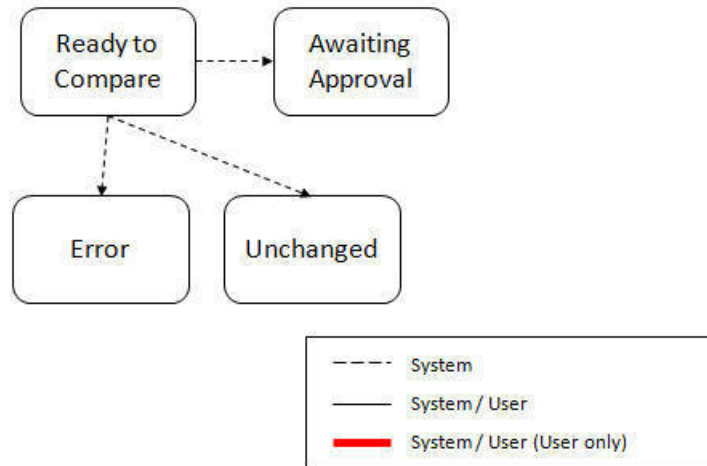
- If any related migration object is in the Error Comparing state, the transaction transitions to **Error Comparing**.
- If all related migration objects are in the Unchanged state, the transaction transitions to **Unchanged**.
- Otherwise, the transaction transitions to **Ready to Apply**. This means 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.

A user has the option to mark one or more SQLs within a migration object to be suppressed. In this case, if the migration object is approved, then only the SQLs that are not marked as Suppressed will be applied. This option is useful when importing a record that has many child records. It may be that the user wants to only include a subset of the child records. This may be done by suppressing the child records that should not be included.

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.

Alternatively, if the Automatically Apply flag was set to **Yes** when creating the import record, the import data set will progress from **Awaiting Approval** to **Apply Objects**. Refer to [Import Process Summary](#) for more information.

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.

NOTE: Refer to [Running Batch Jobs](#) for more information about streamlining the various steps in the process.

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 referenced foreign key is in a XML 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. The tool handles these dependencies as follows:

- Dependency between master and transaction entities is typically hierarchical and in most cases straight-forward. The tool leverages that knowledge to orchestrate the processing of objects in an optimal way that follows their dependency order as much as possible. Note that relationship between entities could be complex and this approach does not eliminate all processing order related errors but rather significantly reduce them.

- Dependency between configuration entities is more complex and inter-twined and therefore migration objects are not ordered, i.e. the multi-threaded batch process may not process records in the desired order.
- To overcome the potential issue of processing order related errors, 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.

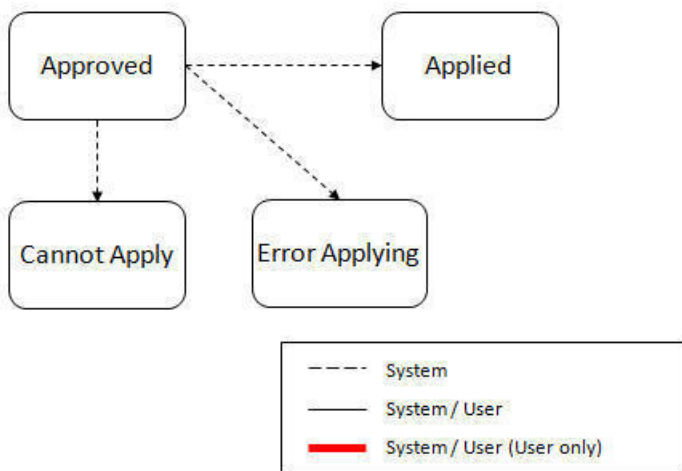
Apply Objects

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.

The following diagram is the portion of the migration object lifecycle that pertains to the Apply step.



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.

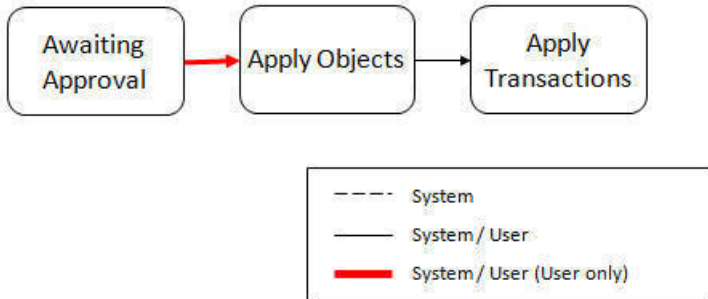
- They are in category 5 described above where the records have a circular reference with another record. For this scenario, the Apply Transactions step described below should successfully apply the records.
- There is some other error that is unrelated to the records in the current migration. In this case, manual intervention may be 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** or the count of records in **Error Applying** does not exceed a configured limit. If so, it automatically transitions the record to the **Apply Transactions** state.

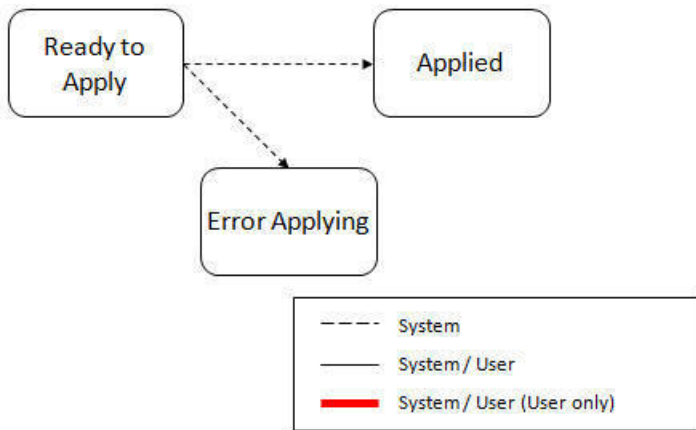
If the number of objects in **Error Applying** exceeds a configured limit, the monitor algorithm does not automatically transition the record. In that case, a user must determine if the large number of errors can be resolved or manually transition to **Apply Transactions** (despite the large number of errors). 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.



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. If the number of records in error is below a certain limit (and the data set monitor batch job is submitted to execute the monitor algorithms) the system will automatically transition the data set to the **Apply Transactions**. If the monitor batch job is not run or if the number of objects in error exceeds a certain limit, a user must make the decision after viewing the errors in the Objects in Error zone on the [Migration Data Set Import](#) portal.

- 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. Errors are visible in the **Transactions in Error** zone on the [Migration Data Set Import](#) portal.

- 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**. The following points describe the Retry logic for migration objects.

- If a user decides to **Retry Objects** (using an action button on the Migration Data Set Import page), the data set transitions to the **Retry Objects** state. At this point the Migration Object monitor must be run.
- The monitor on the **Error Applying** state for the objects detects that the data set is in the state of **Retry Objects** and that triggers the transition back to **Approved**.
- The next step is to transition the data set from **Retry Objects** to **Apply Objects**. This may be done manually or by running the Migration Data Set Import monitor process.
- Now the objects are eligible to be picked up by the object level Apply process.

Analogous logic exists for the migration transactions.

- If a user decides to **Retry Transactions** (using an action button on the Migration Data Set Import page), the data set transitions to the **Retry Transactions** state. At this point the Migration Transaction monitor must be run.
- The monitor on the **Error Applying** state for the transactions detects that the data set is in the state of **Retry Transactions** and that triggers the transition back to **Ready to Apply**.
- The next step is to transition the data set from **Retry Transactions** to **Apply Transactions**. This may be done manually or by running the Migration Data Set Import monitor process.
- Now the transactions are eligible to be picked up by the transaction level Apply process.

The retry logic may also occur when transitioning between the Apply Objects and Apply Transactions depending on whether or not there are errors. The following scenario highlights this point.

- After the **Apply Objects** step there are objects in **Error Applying**. The data set transitions to **Apply Transactions** and the Apply step is done at the transaction level.
- After the **Apply Transactions** step there are transactions in **Error Applying**.
- User chooses to try to apply objects again (by clicking **Retry Objects**). The steps outlined above for retrying objects are followed at this point.
- After the apply objects, user may choose to retry objects again (after fixing errors if applicable).
- At some point the user will transition to **Apply Transactions** again. If there are transactions in **Error Applying**, the system will automatically transition the data set to **Retry Transactions** and the steps outlined above for retry transactions are followed.

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 **Completed** 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 Prior to Comparing

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](#) primary instruction may be used to adjust the data when importing. This algorithm is executed by the comparison algorithm before any comparison is performed. Algorithms of this system event receive the view of the source record (being imported) and the view of the existing record in the target region, if it exists. The data is provided using the physical BO of the migration plan's maintenance object. The algorithm may make changes and pass a new view of the record that should be used for the comparison. This system event supports multiple algorithms that are executed in sequence. Each algorithm receives the original record's data, the target record's data (if applicable) and the 'new' view of the data (as populated by previous algorithms, if any). The final 'new' view of the data is used for the object comparison.

FASTPATH: Refer to [Base Business Objects](#) for more information about physical BOs.

Some examples of records that may require import algorithms.

- Batch Control references its next batch sequence number along with snapshot information like the last run date / time. This information 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.

Note that it is possible to use the algorithm to "reset" the source data as a way of indicating that the record should not be imported. For these situations, the migration object comparison step will transition the record to **Unchanged** and will use an object action value of **Canceled**. (Note that object action is a simple lookup value. The record is not transitioned to the **Canceled** BO state as to reserve that status for user initiated cancellations of the object or one of its parent records). This technique not expected to be used often because ideally using appropriate selection criteria at export time should ensure that the only records exported are those that should be imported.

NOTE: Legacy 'Import' system event. The system originally provided an Import system event / plug-in spot. The purpose of algorithms for this plug-in spot were similar in that they were meant to adjust imported data prior to adding or updating. The algorithms were executed in the Apply step. The logic does not allow for easily interacting with the record using a BO. This makes it difficult to use a plug-in script as the plug-in type. In addition, it is difficult to update elements in an XML column. The support for the plug-in spot will be removed in a future release. Algorithms to adjust the data should be using the pre-compare system event.

Import Process Summary

The following table summarizes the steps required to complete the import process from start to finish. Note that this section **only a summary** and assumes that you are familiar with the details described in the previous sections. It highlights what steps are manual and what steps are performed by a batch monitor process. For each step, the table highlights the Next Action sequence that would occur. For the Apply steps, there are two parts where multiple next actions are possible based on whether there are errors and the user's decision on how to resolve the error. Refer to [Resolving Errors](#) for more information. The possible next actions have the same sequence with a letter following the sequence highlighting the action to take based on the results of the previous step.

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	10	Create Import record	Manual	Migration Data Set Import - Add		Migration Data Set Import - Pending	11
	11	Import file	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Ready to Compare Migration Transaction - Pending Migration Object - Pending	20
Compare	20	Migration Object Compare	Batch		F1-MGOPR - Migration Object Monitor	Migration Object - Approved, Needs Review, Rejected, Unchanged	21

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
						or Error Comparing	
	21	Migration Transaction status update	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply, Unchanged or Error Comparing	22
	22	Migration Data Set Import status update	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Awaiting Approval, Apply Objects (if configured for Automatic Apply), Unchanged or Error	30
Approval	30	Review comparison results, approve / reject as needed (This step is skipped if the data set is configured for Automatic Apply)	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	Apply Objects	Batch		F1-MGOAP - Migration Object Monitor - Apply	Migration Object - Applied or Error Applying	41 Appropriate next action is based on error review, if applicable.
	41a	Migration Data Set Import status update - auto transition to Apply Transactions. Only applicable if the number of migration objects in Error Applying is below a threshold	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Transactions	42
	41b	Migration Data Set Import status update - manual transition to Apply Transactions. 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			

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
		System detects that all the transactions are in the Ready to Apply state and proceeds to the Apply Transactions state.				Migration Data Set Import - Apply Transactions	42
		System detects that there are transactions in the Error Applying state and transitions instead to the Retry Transactions state.				Migration Data Set Import - Retry Transactions	45
	41c	Migration Data Set Import status update - manual transition to Retry Objects. Occurs if user reviews errors and 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 - Retry Objects	44
	42	Apply Transactions	Batch		F1-MGTAP - Migration Transaction Monitor - Apply	Migration Transaction - Applied or Error Applying Migration Object - Applied or Error Applying	43 Appropriate next action is based on error review, if applicable.
	43a	Migration Data Set Import status update - auto transition to Completed Applicable if all transactions are Applied	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Completed.	N/A
	43b	Migration Data Set Import status update - manual transition to Retry Objects. Occurs if user reviews errors and decides to fix an external error and wants	Manual	Migration Data Set Import - click Retry Objects		Migration Data Set Import - Retry Objects	44

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
		to try the batch level Apply again at the Object level.					
	43c	Migration Data Set Import status update - manual transition to Retry Transactions. Occurs if user reviews errors and 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 - Retry Transactions	45
	44	Migration Objects status update from Error Applying back to Approved. Occurs if user chose to Retry Objects.	Batch		F1-MGOPR - Migration Object Monitor	Migration Object - Approved	
		Migration Data Set Import status update from Retry Objects to Apply Objects	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Objects	40
	45	Migration Transactions status update from Error Applying back to Ready to Apply. Occurs if user chose to Retry Transactions or if the user transitions to Apply Transactions and the system detects that there are Transactions in the Error Applying state.	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply	42
		Migration Data Set Import status update from Retry Objects to Apply Objects	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Transactions	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: <ul style="list-style-type: none"> • Pending • Ready to Compare • Apply Objects • Retry Objects • Apply Transactions • Retry Transactions
F1-MGTPR	Migration Transaction Monitor (Deferred)	Processes transaction records in the following states: <ul style="list-style-type: none"> • Pending • Error Applying
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>NOTE: 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"> • Pending • Needs Review (check for Data Set cancellation) • Rejected (check for Data Set cancellation) • Error Applying
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>NOTE: 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, including ways to automate the above steps.

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.).
- If multiple migration requests are exported at the same time, on the import side, 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 the object to transition to "Cannot Apply". If instead you wait until the first file is completed 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. This may also be avoided by using a Group migration request to include all objects in one file rather than multiple files.
- The system provides an algorithm to purge "unchanged" migration objects for a given migration data set. This may be plugged in as a BO exit algorithm on the **Ready to Compare** state for the Migration Data Set Import business object (**F1-MigrDataSetImport**).

Caching Considerations

There are two important caching considerations when using CMA to migrate administrative data.

- The "apply" batch processes update data. As described in [Batch Cache](#), if any batch job is updating data marked as "cached for batch" it must be run in a special thread pool. Contact your system administrator and ask for the name of a thread pool that has turned off caching. This thread pool name must be entered in the batch controls used for CMA processing in the **Thread Pool Name** parameter.
- After a successful migration, the target region now has new administrative data which needs to be part of various caches. It is recommended to flush the server cache (which will trigger a 'global' flush of the cache). If the thread pool workers in the target region are configured to refresh their caches when a global flush is requested, then this is the only step required. If not, then the **F1-FLUSH** batch job should also be submitted to refresh the caches used in batch processing.

FASTPATH: Refer to [Caching Overview](#) for more information.

FASTPATH: Refer to [Running Batch Jobs](#) for more information about running CMA batch jobs.

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. Refer to [Importing and Applying a Migration](#) for an overview of the import process.

Navigate using **Admin > Implementation Tools > Migration Data Set Import**. You are brought to a query portal with options for searching for import data sets. In addition, the query provides an option to specifically search for data sets that have either objects in error or transactions in error. Once a data set has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones are visible on the **Main** tab:

- **Migration Data Set Import.** This zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.
- **Entity Summary.** This zone provides a summary count of all migration objects included in the current data set across the various statuses.

The following zones are visible on the **Transactions** tab:

- **Migration Data Set Transactions in Error.** This zone is only visible if there are transactions for this data set in a non-final state that have errors. It indicates the error for each transaction. 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 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.

The following zones are visible on the **Objects** tab:

- **Migration Data Set Objects in Error.** This zone is only visible if there are objects for this data set in a non-final status that have errors. 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 drill into the record to transition it to **Error Applying** or choose to manually fix the cause of the errors and click **Retry Objects**. The user may also choose to select one or more records to **Reject**.

NOTE: Refer to [Apply Step](#) for more information about resolving errors.

- **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. A user may choose to update the status of one or more records by checking the records and clicking **Approve**, **Reject** or **Needs Review** accordingly.

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.

Refer to [Importing and Applying a Migration](#) for an overview of the import process.

The following zones are visible on the main tab:

- **Migration Transaction.** This 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. A user may choose to update the status of one or more records by checking the records and clicking **Approve**, **Reject** or **Needs Review** accordingly.

- **Entity Summary.** This zone provides a summary count of all migration objects included in the current transaction across the various statuses.

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.

Refer to [Importing and Applying a Migration](#) for an overview of the import process.

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.

The **Migration Object - List of SQL Statements** zone lists the SQL statements that have been generated as a result of the comparison step. A check mark in the Suppressed checkbox indicates that the SQL for that row will not be applied during the apply step.

The **Migration Object - Algorithms** zone displays any pre-compare algorithms that are associated with the migration plan for this migration object.

Running Batch Jobs

There are several batch jobs that are part of the CMA process, especially the import step (which are highlighted in [Import Process Summary](#)). 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 the [Batch Control](#) page 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.
- **Event Driven.** The system provides BO enter plug-in algorithms and batch control post processing plug-in algorithms that automatically submit the appropriate next batch job for that step in the process. This allows for as much automation as possible for the steps that don’t require user input. Note that configuration is required because the BOs / batch controls are not configured for this scenario by default. The following table highlights the BO and status where an algorithm may be plugged in and the name of the algorithm to use.

Business Object	Status	Algorithm
Migration Data Set Export	Pending	F1-MGDPR-SJ (Submit Migration Data Set Export Monitor).
Migration Data Set Import	Pending	F1-MGDIM-SJ (Submit Migration Data Set Import Monitor).
	Ready To Compare, Retry Objects	F1-MGOPR-SJ (Submit Migration Object Monitor).
	Apply Objects	F1-MGOAP-SJ (Submit Migration Object Apply Monitor).
	Apply Transactions	F1-MGTAP-SJ (Submit Migration Transaction Apply Monitor).
	Retry Transactions	F1-MGTPR-SJ (Submit Migration Transaction Monitor).

The following table highlights the batch controls where an algorithm may be plugged in and the name of the algorithm to use.

Batch Control	Algorithm
F1-MGTPR (Migration Transaction Monitor)	F1-MGDIM-NJ (Submit Migration Data Set Import Monitor).
F1-MGTAP (Migration Transaction Monitor - Apply)	F1-MGDIM-NJ (Submit Migration Data Set Import Monitor).
F1-MGOAP (Migration Object Monitor - Apply)	F1-MGDIM-NJ (Submit Migration Data Set Import Monitor).
F1-MGOPR (Migration Object Monitor)	F1-MGTPR-NJ (Submit Migration Transaction Monitor).

- **Manual submission.** The user managing the CMA import process submits the appropriate batch jobs on demand when a particular step is ready. Navigate to [Batch Job Submission](#), select the appropriate batch control and fill in the parameters as needed.

Note that for the CMA batch processes used to migrate administrative data, there are multiple important caching considerations. Refer to [Caching Considerations](#) for more information.

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 wait until the supported number of threads are finished, possibly resulting in a large number of errors in the Apply steps.

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.

The following points highlight some information about Framework-provided migration requests. Navigate to the migration request page in the application to view the details of all provided objects.

- Several base migration requests are supplied to logically group system and administrative tables. For example, there is a migration request for Framework System Configuration **F1-SystemConfig** where most system configuration objects are included. There is another one provided for CMA related configuration objects.
- The system supplies a group migration request **F1-FrameworkConfig** (Framework Configuration), which includes several other migration requests. The expectation is that this migration request includes all the typical objects that are included in a wholesale migration. Your specific product may include this migration request into its own group migration request to support a wholesale migration of all the framework and product administrative tables. An implementation may choose to build a custom group migration request. In this case, review the various migration requests provided by base to see if any may be included as components for the custom migration request. Then any new migration plans added to the base migration request in future releases are automatically included in future migrations.
- There are several different security related migration requests that include different combinations of migration plans to support multiple possible business requirements related to security migration. Note that the security migration request included in the above mentioned group migration request is the one that does not include users. If your implementation wants to copy users to a target environment, refer to [Importing Data that References a User](#) for some considerations.

NOTE: 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 following points highlight some information about the Framework-provided migration plans. Navigate to the migration plan page in the application to view the details of all provided objects.

- Fields and characteristic types are not migrated with an object (like a business object or a data area) unless specifically indicated.
- The **Application Service** used by an object is migrated only if it is CM-owned.
- The **Batch Control** object optionally references a User (for 'timed' batches). Refer to [Importing Data that References a User](#) for some considerations about copying the user. Also note that when running a batch job, snapshot information is captured on the batch control. Updates like this increment the version number. If a batch control record is part of the migration and the comparison step has detected a change to the batch control, the Apply step will error out for this batch control if a batch job is submitted between the compare and apply step.

NOTE: CMA batch controls that are part of the import step are executing and as such, the system does not include these records in a migration. If your implementation changes default parameters for any of the batch controls, the recommendation is to manually make those changes to the target region.

- 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 targeted 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 targeted 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.
- For scripts, schema-based objects and zones, the migration plans provided by the product migrate, through constraints, some of the typical associated data 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. The iterative processing functionality of the import step should resolve any timing issues that may result in validation errors for these types of objects.
- 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). 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. It may be included in any appropriate custom targeted 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 (**Administrator** access mode for the Feature Configuration application service (**CILTWSDP**)).
- The common attachments in the Attachment maintenance object may be considered administrative data to include in a migration. Because this MO has a system generated key, as described in [Administrative Data with System Generated Primary Keys](#), it uses a logical key of the file name and the creation date to determine if the record exists in the target environment. In addition, this MO contains administrative data (common attachments) and non-administrative data (owned attachments). To try to minimize the possibility of key "collision", new common attachments receive a generated key that includes a zero in the middle whereas owned attachments receive a generated key that does not have a zero in the middle.
- The Menu maintenance object has a user defined key, however, its menu lines and menu items have system generated keys. To avoid the possibility of overriding a menu line or menu item incorrectly, the menu MO will check the menu line's menu name in the source and target to be sure they match and will check the menu item's menu line in the source and target to be sure they match otherwise an error will be issued in the comparison step.
- For the system messages, the product provides three different migration plans.

- Message Category and its Messages (F1-MessageCategory). This migration plan is included in the **F1-SystemConfig** migration request.
- Message Category (F1-MessageCategoryOnly). This migration plan is provided to support a targeted migration where an implementation has created a custom message category and wants to move it but doesn't want to move all its messages.
- Message (F1-Message). This migration plan is provided to support a targeted migration where only specific messages within a message category should be migrated.
- For lookup values, the product provides two different migration plans.
 - Lookup Field and its Values (F1-Lookup). This migration plan is included in the **F1-SystemConfig** migration request.
 - Lookup Value (F1-LookupValue). This migration plan is provided to support a targeted migration where only specific lookup values within a lookup field should be migrated.
- There are some system data objects where no information in a base delivered record may be modified by an implementation. For these records, the base delivered migration requests include selection criteria to only select CM-owned records (because the base records will always exist in the target region assuming both regions have the same release). An example is Algorithm Type. The **F1-SystemConfig** migration request only includes CM-owned algorithm types. However, many system data objects support custom changes to one or more fields, for example the Zone object allows an implementation to override the zone text or certain parameters. Other system data objects support custom additions to a collection. For example, the Maintenance Object allows an implementation to add algorithms or options. For the migration plans related to these system data objects, all records are included in the base delivered migration requests to allow for any customized configuration to be migrated. It means that during the Import / Compare step many base delivered objects that are not customized will be marked **Unchanged**.
- Many of the integration related maintenance objects that include references to environment-specific data, such as Message Senders. This data should be migrated with extreme care. When appropriate, consider taking advantage of URI substitution. Refer to [Referencing URIs](#) for more information.

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.

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.

Conversion

This section describes the overall conversion process and the tools provided to support it.

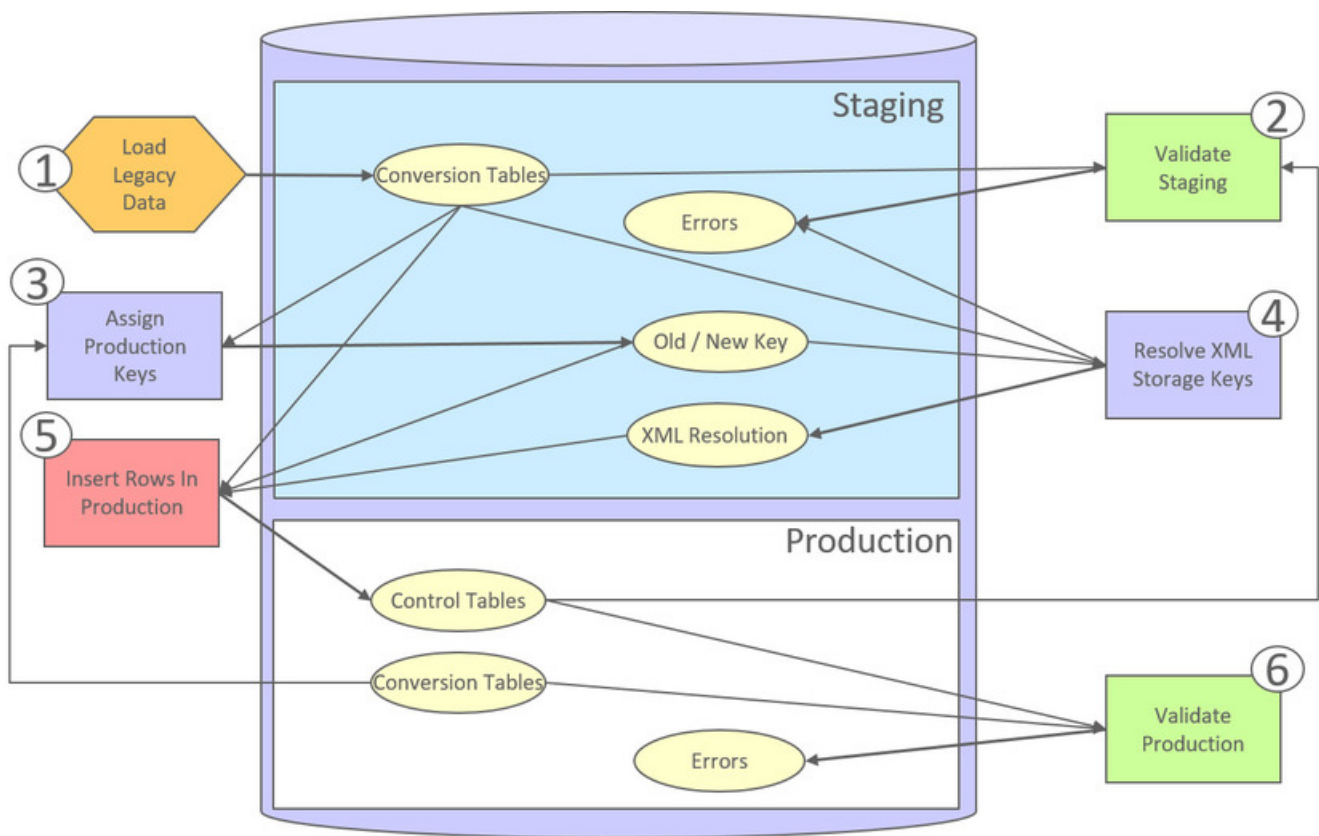
Understanding The Conversion Process

When you're ready to convert data from your legacy system, you will have analyzed your processing requirements according to your business and organizational needs and set up the control tables accordingly. After the control tables are set up, you are ready to load data into the system from your legacy system.

There are two table owners in the system database. We refer to the first owner as **staging** and the second owner as **production**. The staging owner is linked to the tables into which you insert your pre-validated data. The production owner is linked to the tables used by your production system.

NOTE: It's important to notice that **control tables exist only in production**. In staging, they exist as views to the production schema. Refer to [Multiple Owners In A Single Database](#) for more information about table ownership.

The conversion effort involves several steps as illustrated in the following diagram:



The following points briefly outline each of the above tasks:

- **Load Legacy Data.** During this step, your legacy data is loaded into the system. Notice that you are not migrating this data directly into production. Rather, your rows are loaded into tables that are identical to the production tables; they just have a different owner. Mapping legacy data into the system is probably the most challenging part of the conversion process because the system is a normalized database (and most legacy applications are not).
- **Validate Staging.** During this step, the system validates the data you loaded into the staging tables. Note that validation rules validate your staging data using the control tables that have been set up in production.
- **Assign Production Keys.** During this step, the system allocates random, clustered keys to the rows in the staging database.
- **XML Resolution.** During this step, the system resolves legacy keys that may be mapped to an XML storage field with new values that were assigned in the key assignment step.
- **Insert Rows Into Production.** During this step, the system populates your production tables with rows from the staging. When the rows are inserted, their prime keys and foreign keys are reassigned using the data populated in the key assignment and XML resolution steps.
- **Validate Production.** During this step, you rerun the object validation processes, but this time against production. We recommend performing this step to confirm that the insertion programs have executed successfully.

NOTE: Additional steps may be needed to support specific data conversion requirements. Refer to your product's documentation for additional conversion related information.

This section further discusses concepts and guidelines related to the conversion process.

Conversion Entities

Conversion is enabled for selected master and transaction maintenance objects. In the same way, only a subset of a maintenance object's tables are enabled for conversion.

The following is provided for each maintenance object that supports conversion:

- An [entity validation](#) batch control. This batch process validates data in staging and can also be run to validate data in production.
- A [key assignment](#) batch control for each **table** that is eligible for conversion and has a system-generated key.
- An [XML resolution](#) batch control if the maintenance object has at least one table that includes an XML storage field.
- An [insertion to production](#) batch control for every **table** of the maintenance object that is eligible for conversion.

You may review the conversion configuration provided for an entity and more using the [Conversion Entity Dashboard](#) portal.

Conversion Steps

The following sections provide more details about the steps in the conversion process.

NOTE: Additional steps may be needed to support specific data conversion requirements. Refer to your product's documentation for additional conversion related information.

Load Legacy Data Into Staging Tables

This section provides some high level discussion about mapping and loading legacy data to the system's staging tables.

Mass Load Utility

Unless in a Cloud installation, you can use any method you prefer to load data from your legacy application. However, we recommend that you investigate your database's mass load utility (as opposed to using insert statements) as the mechanism to load the staging tables. In addition, we strongly recommend that you disable the indexes on these tables before populating these tables and then enable the indexes after populating these tables.

NOTE: In a Cloud installation, refer to the "Data Conversion Support for Cloud Implementations" for more information about the various tools provided to support database related conversion tasks.

Populating Keys

Some tables in the staging database have keys that are system-assigned random numbers. Special consideration should be given when populating these keys. Refer to [A Note About Keys](#) for more information on how to properly populate these tables. Those tables that don't have system-assigned random numbers have keys that are a concatenation of the parent's prime-key plus one or more additional fields.

Populating Characteristic Tables

There are many maintenance objects that include a characteristic table used to capture miscellaneous information about the object. Most of these tables include an indexed column used when searching by characteristic value called Search Characteristic Value. During conversion and depending on the type of characteristic, this column must be populated as follows:

- **Predefined.** Populate search characteristic value with the contents of the characteristic value column converted to upper case.
- **Ad hoc.** Populate search characteristic value with the first 50 bytes of the ad hoc characteristic value column converted to upper case.
- **Foreign key.** Populate search characteristic value by concatenating the values of each foreign key characteristic value column to a maximum of 50 bytes.

Validate Information In The Staging Tables

During the first validation step, the system validates the data you loaded into the staging tables. This section provides some high level discussion about the validation process.

A Batch Process Per Maintenance Object

Each of the maintenance objects that are eligible for conversion must be validated using its respective entity validation conversion batch process. These are classic background processes that can also be run against production data.

These processes are multi-threaded and have no dependency on other processes.

Validation Mode

By default, each entity is fully validated for schema as well as entity validation rules, at the maintenance object and business object level. The batch process performs the same business rules that are used to validate data added by users in your production system. You may restrict the batch process to perform schema validation only or entity validation rules only using a batch parameter. This may be useful in identifying common data issues in preliminary runs.

Random Sample Mode

By default all records are selected for validation. The "Process Every Nth" batch parameter allows you execute this process in a random-sample mode to highlight pervasive errors. In this mode, you are actually telling the batch process to validate every N records.

We strongly recommend validating each entity in the following steps:

- Execute the validation batch process in random-sample mode.
- Review and correct the errors. Note, you can use the base package's transactions to correct an error if the error isn't so egregious that it prevents the object from being displayed on the browser.
- After all pervasive errors have been corrected; re-execute each object's validation batch process in all-instances mode to highlight elusive, one-off errors.

Reported Errors

Errors encountered during validation are logged onto the conversion Validation Error (CI_VAL_ERR) table. Note that at the start of this job, all rows in the conversion error table for the process maintenance object are deleted.

You can view errors highlighted by the validation process using the [Conversion Entity Dashboard](#) portal.

Recommendations To Speed Up Validation

The following points describe ways to accelerate the execution of the validation process:

- Ensure that statistics are recalculated after data has been inserted into the staging tables.
- Execute the process multi threaded.
- Execute shorter running validation processes (e.g., less records) first so that the error data can be analyzed while other processes are busy running.

- Remember that validation can be run in random-sample mode. We recommend running these batch processes using a large sample value for this parameter until the pervasive problems have been rectified.

NOTE: In a Cloud installation, refer to the “Data Conversion Support for Cloud Implementations” for more information about the various tools provided to support database related conversion tasks.

Another use for these programs

In addition to validating your objects after conversion or an upgrade, the validation programs have another use. Say for example, you want to experiment with changing the validation of a business entity and you want to determine the impact of this new validation on your existing records. You could change the validation and then run that entity’s validation batch process- it will produce errors for each record that fails the new validation.

Allocate Production Keys

During the key assignment step, the system assigns a new key for each legacy key for a table that its prime-key requires a system-generated random key. The conversion process allocates new prime keys to take advantage of the system's parallel processing and data-clustering techniques in the production system (these techniques are dependent on randomly assigned, clustered keys).

The topics in this section provides some high level discussion about the key assignment process and describe the background processes used to assign production keys to the staging data.

The Old Key / New Key Table

It's important to understand that the system does not overwrite the prime-keys on the rows in the staging database, as this is a very expensive IO transaction. Rather, a series of tables exist that hold each row's old key and the new key that will be assigned to it when the row is [transferred into the production database](#). We refer to these tables as the "old key / new key" tables.

The convention "<1st letter of owner flag>K_<table_name>" is used to denote the old key / new key table name. For example, the old key / new key table for CI_ACCT is called CK_ACCT.

The insertion batch process that transfers the rows into the production database use the new key for the main record of the key along with any other record where this key is a foreign key. In the same way, the XML resolution process resolves conversion foreign keys residing in XML storage fields and replaces them with their corresponding new keys from these table.

A Batch Process Per Table

A key assignment batch process is provided for each table that has a system-generated key and belongs to a maintenance objects that is eligible for conversion. The batch process is responsible for populating the corresponding old key / new key table (i.e., you don't have to populate these tables). These processes are single threaded. Refer to a table’s corresponding key assignment batch process for more information.

Key Assignment Dependencies

Most tables with system-generated keys do not inherit part of their key from another table’s key. Their corresponding key assignment batch process have no dependencies and can therefore be executed in any order you please.

Some tables inherit part of their key from another table’s key. Key assignment batch processes for such tables must be executed in key inheritance order. In other words, the key assignment process for a table should be run after the process that generates keys for the entity it depend on. The [Conversion Entity Dashboard](#) portal displays key inheritance dependency information if any exist for a conversion entity.

NOTE:

You may run multiple key assignment batch process in parallel as long as they are independent with respect to key inheritance.

Iterative Conversions

Rather than perform a "big bang" conversion (one where all records of an entity are populated at once), some implementations have the opportunity to go live on subsets of their entity base. If this describes your implementation, please be aware that the system takes into account the existing prime keys in the production database before it allocates a new key value. This means when you convert the next subset of customers, you can be assured of getting clean and unique keys.

Key assignment logic creates the initial values of keys by a manipulation of the sequential row number, starting from 1. After any conversion run, a subsequent conversion run will start with that row number again at 1, and the possibility of duplicate keys being assigned will be higher. The key assignment batch process allows you to specify a starting value for that row number. The purpose of this parameter is to increase the value of row number by the given value, and minimize the chance of duplicate key assignment. This parameter is only used if you are performing conversions where data already exists in the tables in the production database.

Run Type

Key assignment is performed in two steps:

- **Initial Key Generation.** The system allocates new keys to the rows in the staging tables (i.e., it populates the respective old key / new key table).
- **Duplicate Key Resolution.** The system reassigns keys that are duplicates compared to production.

By default, both steps are performed at the same run but you have the option to run them separately by indicating which step to run via a batch parameter. The proper use of this parameter will greatly speed up the key assignment step as described under **Recommendations To Speed Up Key Generation** section.

Recommendations To Speed Up Key Generation

The following points describe ways to accelerate the execution of the key generation programs.

- For a non-Cloud installation:
 - Make the size of your rollback segments large. The exact size is dependent on the number of rows involved in your conversion. Our research has shown that processing 7 million rows generates roughly 3GB of rollback information.
 - Setup the rollback segment(s) to about 10GB with auto extend to a maximum size of 20GB to determine the high water mark
 - A next extent value on the order of 100M should be used.
 - Make sure to turn off all small rollback segments (otherwise Oracle will use them rather than the large rollback segments described above).
 - After the key assignment programs execute, you can reclaim the space by:
 - Keep a low value for the "minimum extent" parameter for the rollback.
 - Shrink the rollback segments and the underlying data files at the end of the large batch jobs.
- Compute statistics on the old key / new key tables after every 50% increase in table size. Key generation is performed in tiers or steps because of the inheritance dependency between some tables and their keys. Although key generation for the inheritance dependency tier currently being processed is performed by means of set-based SQL, computation of statistics between tiers will allow the database to compute the optimum access path to the keys being inherited from the previous tier's generation run.
- Optimal use of the **Run Type** batch parameter.
 - Before any key assignments, alter both the "old key" F1_CX_ID index and the "new key" CI_ID index on the old key / new key tables to be unusable.

- Run all key assignment batch processes in tiers, submitting each job to only perform the **Initial Key Generation** step.
- Rebuild the indexes on the old key / new key tables. Rebuilding the indexes using both the PARALLEL and NOLOGGING parameters will speed the index creation process. Statistics should be computed for these indexes.
- Run all key assignment batch processes in the current tier that were previously run in initial key generation mode, to perform the **Duplicate Key Resolution** step. This will reassign all duplicate keys.

NOTE: In a Cloud installation, refer to the “Data Conversion Support for Cloud Implementations” for more information about the various tools provided to support database related conversion tasks.

XML Resolution

Most system-generated foreign keys are stored in physical fields and characteristic tables and as such their legacy value is replaced with its corresponding new key as part of the insertion to production process. Rarely, maintenance objects may store system-generated foreign keys in an XML storage field, i.e. a field defined with the data type of CLOB or XML. This step is only applicable to such maintenance objects.

During this step, the system resolves convertible system-generated foreign keys that may reside in XML storage fields you may have loaded into the staging tables. This section provides some high level discussion about the XML resolution process.

The XML Resolution Table

It's important to understand that the system does not overwrite the prime-keys on the XML storage fields in the staging database, as this is a very expensive IO transaction. Rather, a corresponding XML resolution table exists for each table that defines an XML resolution field to capture each row's resolved XML storage content, i.e. the content with all the old keys replaced with the new assigned keys.

The convention "<1st letter of owner flag>R_<table_name>" is used to denote the XML resolution table name.

The [insertion](#) batch process that transfers the rows into the production database replaces each XML storage field with its resolved value from the corresponding XML resolution table.

A Batch Process Per Maintenance Object

Each of the maintenance objects that are eligible for conversion and support XML storage fields is provided with an XML resolution batch process. These batch processes must be run to resolves foreign keys that may reside in these XML storage fields.

These processes are multi-threaded and must be executed after the key assignment step has completed and before inserting data to production.

XML Resolution Eligibility

Not all maintenance objects that support XML storage fields actually store convertible system-generated foreign keys in their XML storage field. If none of the business objects associated with the maintenance object involve mapping of such foreign keys to an XML storage field then XML resolution is not needed for any row in the maintenance object. The XML resolution batch process detects such situation and completes right away without storing any rows in any of the maintenance object's XML resolution tables.

NOTE: If the XML resolution batch process detects that the maintenance object is eligible, the maintenance object must reference a physical BO. If a product delivered business object supports system-generated foreign keys in an XML storage field, the product should be supplying a physical BO. If your implementation has system-generated foreign keys in an XML storage field in a custom business object for a maintenance object where the product has not supplied a physical BO, you must [create a physical BO](#) and link it to the maintenance object.

Only Resolved Values Are Captured

XML storage fields typically store large amounts of data. To avoid capturing the same XML content redundantly, the system only stores values in the resolved XML storage fields if the resolved value is different than the original value, i.e. at least one key was resolved.

If for a given record the resolved XML content is the same as the original content then the following rules apply:

- If this is the primary table of the maintenance object a record is inserted into its corresponding XML resolution table for that record with no value in the XML storage field.
- If this is a child table of the maintenance object then no record is inserted into the corresponding XML resolution for that record.

Reported Errors

Errors encountered during XML resolution are logged onto the conversion Validation Error (CI_VAL_ERR) table. Note that at the start of this job, all rows in the conversion error table for the process maintenance object are deleted.

You can view errors highlighted by the XML resolution process using the “Validation Error Summary” page.

Insert Production Data

The topics in this section describe the background processes used to populate the production database with the information in the staging database.

A Batch Process Per Table

An insert to production batch process is provided for each table of a maintenance objects that is eligible for conversion.

The batch process is responsible for transferring to production all the rows from the staging table while replacing all references to old keys with their corresponding new keys as follows:

- Keys residing in prime keys, foreign key fields and characteristics are resolved using the corresponding [old key / new key](#) tables.
- Keys residing in generic foreign key reference tables where each row represents a reference to an entity’s maintenance object and prime key values are also resolved using the corresponding [old key / new key](#) tables. Note that all entities referenced this way must either be convertible or view of production. A reference to an entity that is explicitly marked as not convertible is not supported as it cannot be resolved.
- Keys residing in XML storage fields are resolved using the corresponding [XML resolution](#) tables.

NOTE: Foreign keys residing in any other form of storage are not resolved. For example, any references stored in raw elements, log records, or any other field that is not used as an XML storage field.

All insertion batch processes are independent and may run concurrently. Also note, all insertion batch processes can be run in many parallel threads as described in the next section (in order to speed execution).

Recommendations To Speed Up Insertion

The following points describe ways to accelerate the execution of the insertion batch processes:

- Before running the first insertion batch process:
 - Rebuild the index on the prime key on the old key / new key table.
 - Re-analyze the statistics on the old key / new key table.
 - Alter all indexes on the production tables being inserted into to be unusable.

- After the insertion programs have populated production data, rebuild the indexes and compute statistics for these tables.

NOTE: In a Cloud installation, refer to the “Data Conversion Support for Cloud Implementations” for more information about the various tools provided to support database related conversion tasks.

Validate Production

During this step, you rerun the [object validation batch processes](#), but this time in production. We recommend rerunning these batch processes to confirm that the insertion batch processes have executed successfully. We recommend running these batches in random sample mode (e.g., validate every 1000th object) rather than conducting a full validation in order to save time. However, if you have time, you should run these processes in full validation mode (to validate every object).

A Note About Keys

The prime keys of the tables in the staging database are either system-assigned random numbers or they aren't. Those tables that don't have system-assigned random numbers have keys that are a concatenation of the parent's prime-key plus one or more additional fields. Every table whose prime key is a system-assigned random number has a related table that manages its keys; we refer to these secondary tables as "key tables".

The following points provide more information about the key tables:

- Key tables are used by programs that allocate new keys. For example, before a new key is allocated, the key assignment program checks the corresponding key table to see if it exists.
- Key tables only have two columns:
 - The key of the object.
 - An environment ID. The environment ID identifies the database in which the object resides.
- Key tables are named the same as their primary table with a suffix of "_K". For example: The key table for CI_ACCT is CI_ACCT_K.
- The key table for a table whose prime key is a system-assigned is defined on its table definition record.
- When you populate rows in tables with system-assigned keys, you must also populate a row in the related key table. For example, if you insert a row into CI_ACCT, you must also insert a row into CI_ACCT_K. The environment ID of these rows must be the same as the environment ID on this database's [installation record](#).
- When you populate rows in tables that reference this record as a foreign key, you must use the appropriate key to ensure the proper data relationships. For example, if you insert a row in CI_SA for the above account, the ACCT_ID column must contain the temporary account key.
- When you insert rows into your staging database, the keys do not have to be random, system-assigned numbers. They just have to be unique. A later process, [Allocate Production Keys](#), will allocate random, system-assigned keys prior to production being populated.

Multiple Owners In A Single Database

The conversion process relies on the following table owner configuration in the system database:

- The production owner is linked to the tables used by your production system. These tables have an owner ID of CISADM.
- The staging owner is linked to the tables into which you insert your pre-validated data. These tables have an owner ID of CISSTG.

The staging owner schema is almost identical to the production schema, with the following exceptions:

- Control tables are not actual tables in staging but rather views to the corresponding tables in production.
- Conversion specific tables designed solely to support the conversion process exist only in the staging schema. For example, tables that manage key assignment and XML resolution.

This section provides some high level concepts related to these table owners.

Validation Always Uses Production Control Data

When the validation batch processes run against your staging data, they validate the staging data against the production control tables (and insert errors into the staging error table). This means that the SQL statements that access / update entities needs to use the staging owner whereas the SQL statements that access control tables need to use the production owner. But notice that when these same validation batch processes run against production, the SQL statements will never access the staging owner. Rather, they all point at the production owner.

This is accomplished as follows:

- A separate application server must exist for each owner. Each application server points at a specific database user ID.

NOTE: In a Cloud installation, the application server may only point to a specific database user ID at any point in time. Refer to the “Data Conversion Support for Cloud Implementations” for more information about switching owners.

- The database user ID associated with the staging owner uses CISSTG as the owner for the master and transaction tables, but it uses CISADM as the owner of the production control tables.
- The database user ID associated with the production owner uses CISADM as the owner for all tables.

You may wonder why we went to this trouble. There are several reasons:

- We wanted to reuse the validation logic that exists in the programs that validate your production data. In order to achieve this, these programs must sometimes point at the staging owner, and other times they must point at the production owner (and this must be transparent to the programs otherwise two sets of SQL would be necessary).
- We wanted to let you use the application to look at and correct staging data. This can be accomplished by creating an application server that points at your staging database with the ownership characteristics described above.
- We wanted the validation programs to be able to validate your production data (in addition to your staging data). Why would you want to validate production data if only clean data can be added to production? Consider the following scenarios:
 - After an upgrade, you might want to validate your production data to ensure your pre-existing user-exit logic still works.
 - You may want to conduct an experiment of the ramifications of changing your validation logic. To do this, you could make a temporary change to user exit logic (in production) and then run the validation jobs on a random sample basis.
 - You forget to run a validation program before populating production and you want to see the damage. If you follow the instructions in this document, this should never happen. However, accidents happen. And if they do, at least there's a way to determine the ramifications.

Only Validation Can Work In Both Owners

While the redirection of owner ID's is a useful technique for the validation batch processes, it cannot be used by the key assignment and production insert batch processes? Why, because these processes have to access the same tables but with different owners at the same time. They also need to reference conversion specific tables that do not exist in production. For example, the batch process that inserts rows into a table in production must select rows from the staging version of that table, resolve keys from the conversion old key / new key mapping tables and insert the resolved records into the production version of that same table

This is accomplished as follows:

- In staging, a view to production exists for each eligible to conversion table. These views have hard-coded the database owner to point to production. For example, there is a view called CX_PER that points at person table in production.
- The key assignment and insertion programs use these views whenever then need to access production data.

Conversion Entity Dashboard

This portal provides an overview of a maintenance object's conversion configuration along with information related to its current status and progress within the overall conversion process.

Refer to [Understanding Conversion](#) for more information.

You can access the portal from the **Admin Menu > Implementation Tools > Conversion Entity Query**. You are brought to a query portal with options for searching for a specific maintenance object that is eligible for conversion. Once a maintenance object has been selected you are brought to the conversion entity dashboard portal to review the information for the selected record.

The following zones may appear as part of the portal's **Main** tab page.

- **Tables.** This zone provides information about tables that belong to the maintenance object and are marked as eligible for conversion.
- **Batch Controls.** This zone lists the conversion batch controls defined for the maintenance object in the order they should be executed within the conversion process along with current execution details if any.
- **Error Summary.** This zone provides a summary of all conversion errors related to the maintenance object.
- **Duplicate Keys.** This zone lists duplicate legacy keys that need to be resolved by the key assignment conversion process.

The **Errors** tab page provides a query zone that allows you to review conversion errors related to the maintenance object by various criteria.

Data Export

This section describes concepts and tools related to the various types of data export methods supported by the product.

Understanding Data Export

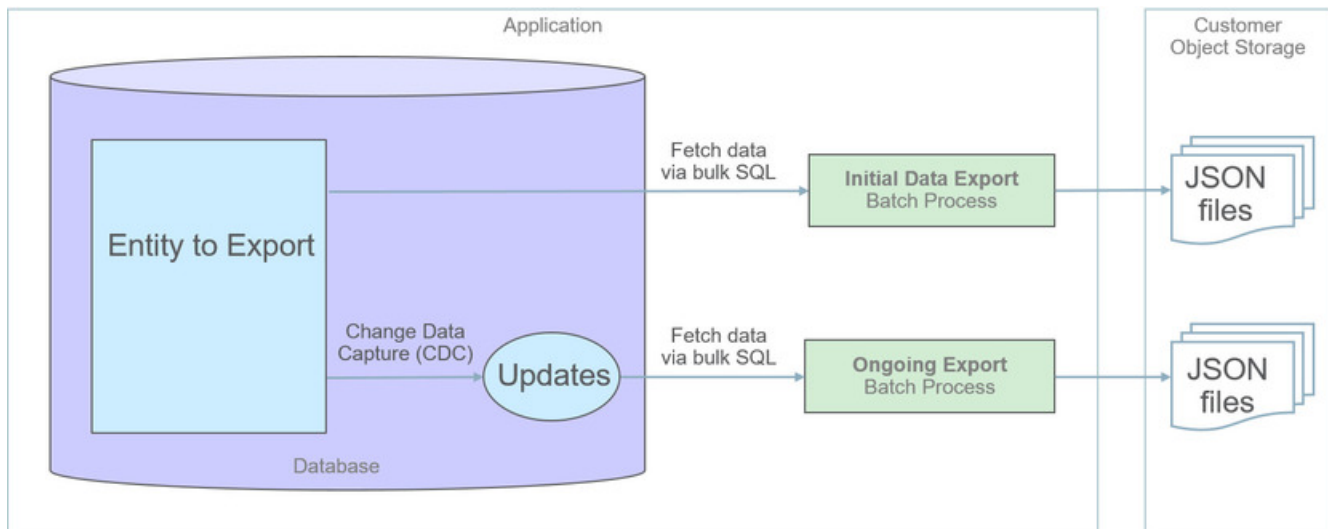
The application supports the following classes of export methods for a maintenance object:

- **Generalized.** This method may be used for maintenance objects that are low to medium in volume and update activity. Most maintenance objects are eligible for this method.
- **Specialized.** This method should be used for maintenance objects that are high in volume or very frequently updated.

This section discusses concepts and guidelines related to data export features.

Generalized Data Export

The generalized data export method provides a file-based export of data in JSON format. Exporting the data of a maintenance object using this method involves an initial export of the entity's entire data followed by an ongoing export of changes as they are made over time. These processes are illustrated in the following diagram:



The overall process involves the following steps:

- Enable entities for export in the [Generalized Export Dashboard](#) portal. You may enable many entities at once. This step also allows you to enable ongoing export of these entities.
- Submit the Generalized Initial Export Initiator (**F1-GEXPI**) batch process to export the current content of all enabled entities. This step submits a separate batch process for each entity you have enabled for export.
- Schedule the Generalized Ongoing Export (**F1-GEEXO**) batch process to run periodically.

The following sections further discuss concepts and guidelines related to the generalized data export method.

Eligibility

By default a maintenance object is eligible for the generalized data export method unless explicitly marked otherwise by the **Data Export Class** option. The option may be used to mark an entity as either not allowed for any type of export or allowed for a [specialized](#) export only. The lack of such explicit option means the maintenance object is eligible for the generalized export.

One Method Fits All

The generalized data export method is designed to accommodate many eligible maintenance objects in a generic way that must also take performance considerations into account. For this reason this method does not support entity specific filtering options nor custom rules of any kind.

The following rules apply to all entities that may be exported using this method:

- Entire data is exported. All fields on all tables that belong to the maintenance object are exported except for “key” and “log” tables if any.
- Same data structure and format is used for all entities. Refer to the **Record Format** section for more information.
- Data is exported to files only. Refer to the **Export To Files Only** section for more information.
- All changes are captured. It is not possible to exclude some changes from being tracked for ongoing export based on custom business rules.
- When data changes, a full snapshot of the data related to that entity is exported to avoid heavy data merging on the target side.

NOTE: Export is not supported for entities that rely on the logical time zone functionality. This type of definition is not commonly used and includes maintenance objects that at least one of their fields is defined as stored in **Logical Standard Time**.

Export To Files Only

The immediate destination of both initial and ongoing export processes are files in a customer specified location. In a Cloud installation files are created in a customer owned Object Storage.

Based on your business requirements you may further consume these files by downstream applications such as data lakes etc.

Enabling Data Export

To enable data export for an eligible maintenance object a **Data Export Control** record needs to be created. The record keeps an indication of whether the initial export for the maintenance object has been completed or not and whether changes made to the entity should be tracked and exported on an ongoing basis. A designated portal allows you to enable the generalized data export for many maintenance objects at once as well as monitor their current export status at a glance. Refer to [Generalized Export Dashboard](#) portal for more information.

NOTE: When applicable, ongoing data export should be enabled before submitting the initial export batch process for the entity. This ensures that changes made during the initial export are properly tracked.

Initial Data Export

Each maintenance object that is eligible for the generalized data export method is associated with an initial data export batch control that is referenced on the maintenance object using the **Export Batch Control** option. The same common batch program is used by all these batch controls to export the entire data of an entity to files. The process is multi-threaded and produces one file per thread by default. An optional batch parameter allows for multiple smaller files to be generated per thread by setting a limit on the number of records written to each file. Refer to the **File Size** section for more information.

Instead of manually submitting the generalized initial export batch process for each maintenance object you have enabled for export, submit the **F1-GEXPI** Generalized Initial Export Initiator batch process to automatically submit them all at once.

The initial export process for each maintenance object updates the corresponding data export control record for the maintenance object to indicate that the process has started and updates it again upon completion. This allows you to view the overall status of the initial export across all enabled entities on the [Generalized Export Dashboard](#) portal.

Note that the initiator batch process only submits batch processes for enabled maintenance objects that according to their data export control record their initial export has not started yet. If you need to export one or more entities again, reset the initial export status indication on their data export control records and submit the initiator batch process again.

The initial export file name is constructed as follows:

```
INIT_EXPORT[_file prefix(optional)]_[maintenance object]_[batch run]_[thread]_[thread count]_[timestamp]
[_file sequence].json.gz
```

The "file sequence" part of the file name is used only when more than one file is created per thread. The sequence suffix is populated starting from the second file as follows `_2`.

Refer to any of the individual generalized initial export batch controls and the initiator batch control for more information.

Ongoing Data Export

Data changes are tracked by the Capture Change for Ongoing Export (**F1-MO-REGCHN**) Maintenance Object Audit algorithm. The algorithm records the prime key of a changed entity in a designated **Data Export Update** table that serves as a queue of changed entities to export. The queue is later on consumed by the ongoing data export batch process.

NOTE: The Maintenance Object Audit plug-in spot is capable of tracking changes made to the entity by any layer of the application. Making changes using direct SQL would bypass this mechanism and therefore should be avoided.

The audit rule is added to a maintenance object when the ongoing data export option on its corresponding data export control record is enabled. In the same way the audit rule is removed from the maintenance object when the ongoing export option is disabled.

NOTE: The ongoing export option has to be enabled on the data export control record for the entity before it is initially exported. This ensures changes made while the initial export process is in progress are not missed.

Unlike the initial export batch process that requires a batch control to be defined for each eligible maintenance object, a single batch process is used to export ongoing changes made to all entities.

The following describes the ongoing export batch process at a high level:

- All changes queued in the Data Export Update table are exported to a designated ongoing export file. Refer to the **Managing a Large Backlog** section for more information.
- A full snapshot of the data related to the changed entity is exported to avoid heavy data merging on the target side.
- Once exported, records are deleted from the Data Export Update table.

The process is multi-threaded and a separate file is produced for each thread by default. An optional batch parameter allows for multiple smaller files to be generated per thread by setting a limit on the number of records written to each file. Refer to the **File Size** section for more information.

NOTE: It is important to schedule the ongoing export batch process to run frequently enough to avoid accumulating a high volume of records in the Data Export Update table. The smaller the table is kept the faster it is for the change data capture mechanism to add records to it.

The ongoing export file name is constructed as follows:

```
INC_EXPORT_[batch run]_[thread]_[thread count]_[timestamp]_[file sequence].json.gz
```

The "file sequence" part of the file name is used only when more than one file is created per thread. The sequence suffix is populated starting from the second file as follows _2.

Refer to Generalized Ongoing Export (**F1-GEEXO**) batch control for more information.

File Size

The initial export batch process produces a single file per thread by default. For some high volume entities this may result in files that are too large to manage. To produce multiple and smaller files for a specific entity, limit the number of records written to each file by setting the corresponding parameter on the initial export batch control defined for the entity. The same parameter may be provided when submitting the Initial Export Initiator batch process, in which case the initiator value is used only when a corresponding value has not been defined on the entity's specific batch control record.

In the same way, the ongoing export batch process produces a single file per thread by default. To produce multiple and smaller files, limit the number of records written to each file by setting the corresponding parameter on the batch control.

To avoid the creation of too many files by a single batch run, the system sets a maximum limit of around 500 files per run. When the requested number of records per file is determined to cause the number of generated files to exceed the system limit, the actual value used is adjusted to meet the system limit.

Managing a Large Backlog

When a large backlog of changes accumulates in the ongoing export queue for some reason, the next batch takes longer to export which delays the downstream import step. A better option is for the backlog to be cleared in smaller "chunks", allowing the downstream process to import them as they are made available.

You may limit the number of records the batch process exports using the **Thread Processing Limit** batch parameter. When specified, the number of records exported by each thread is capped by the specified limit, leaving the remaining records to be consumed by subsequent batch runs. Instead of waiting for the next scheduled batch run, a new batch process may be automatically submitted, if explicitly requested to do so using the **Automatic Backlog Clearance** batch parameter.

It is important to ensure that the backlog clears gradually and that the queue resumes its normal size in a timely manner. The processing limit must therefore be set with an appropriate value that would promote proper clearance of the queue. To do so, at least 20% of the queue, and no less than 100,000 records, must be consumed by each batch run across all threads. If the specified limit does not meet the minimum clearance requirement then the processing limit is adjusted accordingly.

Record Format

Data is fetched and exported directly from the database in bulk read operations for performance reasons. For that reason, the structure used does not reflect the logical data model of the entity but rather its physical list of tables. Data for a given instance is organized by tables followed by rows in each table and for each row its list of fields in JSON format.

The following describes the format and structure used:

```
{
  "OBJ": "<mo name>",
  "TIMESTAMP": "<export time in ISO format for example 2019-07-25T11:06:04.740615Z>",
  "PK1": "<mo pk1 value>",
  "PK2": "<mo pk2 value if any>", # PK2-5 should only be included when applicable
  "PK3": "<mo pk3 value if any>",
  "PK4": "<mo pk4 value if any>",
  "PK5": "<mo pk5 value if any>",
  "DELETED": true, # should only be included when the entity is deleted
  "DATA":
  {
    "<MO table name 1>":
    [
      {<name value pairs of all fields in row 1 in that table>},
      {<name value pairs of all fields in row 2 in that table>},...
      {<name value pairs of all fields in row n in that table>},...
    ],
    "<MO table name 2>":
    [
      {<name value pairs of all fields in row 1 in that table>},
      {<name value pairs of all fields in row 2 in that table>},...
      {<name value pairs of all fields in row n in that table>}
    ],...
    "<MO table name n>":
    [
      {<name value pairs of all fields in row 1 in that table>},
      {<name value pairs of all fields in row 2 in that table>},...
      {<name value pairs of all fields in row n in that table>}
    ]
  }
}
```

Notes about field values and formats:

- All fields for a row are included even when the field is empty or null.
- All string values are trimmed of excess trailing spaces but if a field is empty it exported as an empty string "" and not null.
- An empty array is used for a table node when no records exists for the entity in the table.
- Date/time information is converted to UTC time zone and exported in ISO format for example 2019-07-25T11:06:04.740615Z.
- Deletes have an explicit indicator in the header.

Order of Consuming Files

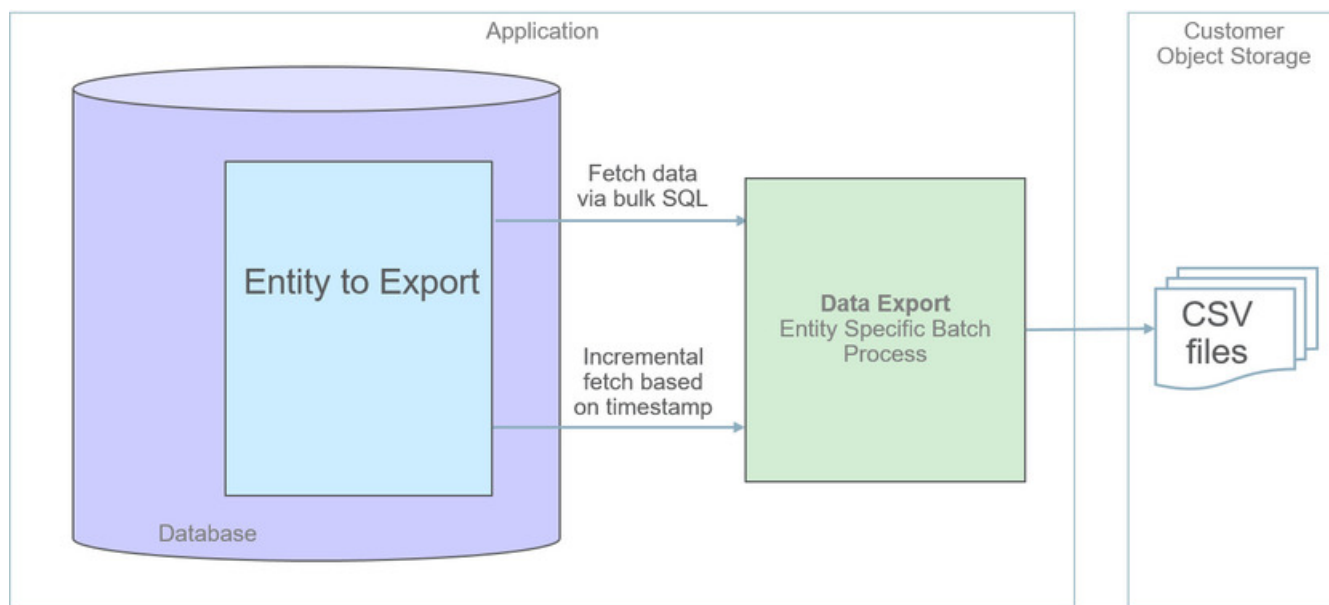
It is important to apply all the initial export files for a maintenance object before applying ongoing change files to ensure data consistency.

The following is a recommended procedure for consuming generalized data export files:

- Apply all initial data export files before ongoing export files.
- Apply initial files for a maintenance object in run number order.
- Apply ongoing export files in run number order.
- If a new initial export is available the file consumption procedure should be changed temporarily as follows:
 - Stop consumption of ongoing export files.
 - Apply the new initial export files.
 - Resume consumption of ongoing export files.

Specialized Data Export

Several performance considerations are called for when exporting the data of a high volume entity or an entity that is updated in extremely high frequency. The generalized method does not fit such entities and a specific batch process is designed for each, aiming to achieve an optimal export method for the entity. The initial run of the process exports the entity's existing data while subsequent runs of the same process exports data incrementally, i.e. it exports records that were added or updated since the previous run based a timestamp field. The overall process is illustrated in the following diagram:



The following sections further discuss concepts and guidelines related to the specialized data export method.

A Specific Process For Each Entity

A maintenance object that is designed to be exported using the specialized method is explicitly marked as such by the **Data Export Class** option and its specific batch process is defined using the **Export Batch Control** option.

All specialized batch processes are designed to export the data in a CSV format as it produces smaller files in size compared to more verbose formats. Refer to each batch process for more information about specific export considerations and parameters.

Enabling Data Export

To enable data export for a maintenance object a specialized type of [Data Export Control](#) record needs to be created. The record serves as an indication that such entity should be exported and keeps track of the date and time through which data has been exported so far, allowing subsequent export run to only export incremental changes.

Incremental Data Export

The specialized export method tracks changes incrementally from prior export run and does not use a change data capture mechanism. It therefore relies on a timestamp field on the primary table of the maintenance object to indicate when data was added or changed. Only entities that were added or changed since the previous export run are exported with each subsequent run. The date and time through which data has been exported is recorded for each maintenance object on its corresponding Data Export Control record.

Order of Consuming Files

Specialized data export files should be consumed in run number order to ensure a consistent and chronological uptake of data.

Defining Data Export Control

This portal is used to maintain Data Export Control records. These records capture your organization's export settings and current export state for a selected maintenance object. For a generalized export it keeps an indication as to whether the initial export for the maintenance object has been completed or not and whether changes made to the maintenance should be tracked and exported on an ongoing basis. For a specialized data export it keeps the date/time through which data was exported.

Refer to [Understanding Data Export](#) for more information.

You can access the portal from the **Admin Menu > Implementation Tools > Data Export Control**. You are brought to a query portal with options for searching for a specific record. Once a data export control record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The **Data Export Control** zone provides information about the selected record.

Generalized Export Dashboard

The number of maintenance objects that are eligible for generalized export is large. This portal allows you to easily select and enable export for many eligible entities at once as well as review the current export state of those already enabled at a glance.

Refer to [Generalized Data Export](#) for more information about the generalized data export method.

You can access the portal from the **Admin Menu > Implementation Tools > Generalized Export Dashboard**.

The following zones may appear as part of the portal's **Main** tab page.

- **Export Entities.** This zone lists entities for which generalized data export control settings have already been set.
- **Eligible Entities.** This zone lists entities that are eligible for the generalized data export method and for which such settings have not yet been set.

Chapter 2

Work and Asset Management Administrative User Guide

The topics in this section describe how to administer Oracle Utilities Work and Asset Management.

General Options

Defining Installation Options

This section provides additional setup information related to your installation options record.

Refer to [Framework installation options](#) for more information.

Formatted Address

Throughout the system address information is formatted for display based on standard address constituents.

NOTE: Logic to format an address resides in an "Address Information" algorithm on the installation options.

Address Geocoding

The scheduler has to be provided with the geocode of any location that may be included on a crew's route. These include locations of assets, service centers, substations, etc.

Unless a geocode is provided, algorithms on the various business entities that capture an address attempt to geocode the address when the entity is added or when the address changes.

The minimum match quality level for address geocoding is configurable. By default, the base system requires an exact match, which is the maximum quality level. You can control the quality level of geocoding for any entity that has an address

by overriding the match quality parameter of the algorithms on each entity with a different minimum quality level, based on your business needs.

NOTE: Logic to geocode an address resides in an "Address Geocoding" algorithm on the installations options.

Geocoding, when used in conjunction with GIS, provides location-based mapping and analysis of assets and activities residing within your infrastructure.

- Map layers display the operational details for services provided by your company.
- Exact location of activities, work orders, and other objects are easily viewed and managed directly from the map.
- New assets created from the map are already plotted with Geocode coordinates.

Access Control

Access control provides a way for your organization to limit or classify record information by “plant,” varying work locations, or work functions. For example, you may have asset types that are only used at your northern location, so you would designate those asset types as restricted to that location. In the same way, you can leverage access control to prevent users in the southern location from accessing work orders related to the northern location (and vice versa). This is referred to as “row level security” in the application framework.

Each location represents an access group and objects in the application may be owned by a specific access group. Access groups are defined in the **Access Group** portal and are assigned to records as an **Owning Organization**.

Refer to [The Big Picture of Row Security](#) in the **Oracle Utilities Application Framework** documentation for more information on data access roles and how they work with access groups.

NOTE: This functionality is optional and not enabled by default. If your implementation needs to restrict access by owning organization, you will need to explicitly enable it as described below.

The following points describe the functionality at a high level:

- Some entities are directly secured by being assigned an owning organization value on the entity itself. For example, the work order entity is directly secured and therefore its owning organization field is populated.
- Some entities are indirectly secured by association to another related entity that is directly secured. Entities secured this way are not assigned an owning organization. For example, work activities are secured indirectly by association to their work order. If a user is allowed to view a work order they are also allowed to view all its work activities. There is no owning organization data security for individual activities.
- All other entities do not support this type of security. For example, operation and technical entities are not used by business users and as such should not be subject to data restrictions. A system administrator needs to be able to review all the data in these entities, unfiltered by owning organization, to properly address issues and exceptions.

The following sections further describe concepts related to data security by owning organization.

Which Entities Are Secured?

Not all entities support this type of data security. The following business entities directly support security by owning organization:

- Asset
- Compatible Unit
- Crew
- Location. This includes all classes of locations stored in the "Node" maintenance object, for example, asset locations, storerooms, etc.

- Purchase Order Header
- Purchase Requisition Header
- Template Work Order
- Work Design
- Work Location
- Work Order
- Work Request

The following business entities indirectly support security by owning organization via association:

- Work Activity. Secured via association to work order
- Activity Reconciliation. Secured via association to work order
- Design Element. Secured via association to work design.
- Expedite Purchase Order. Secured via association to Purchase Order Header.
- Invoice Header. Secured via association to Purchase Order Header.
- Material Request Header. Secured via association to Storeroom (Node).
- Material Return Header. Secured via association to Storeroom (Node).
- Measurement. Secured via association to Asset.
- Physical Inventory Header. Secured via association to Storeroom (Node).
- Warranty. Secured via association to Asset.

The following administrative entities directly support security by owning organization:

- Activity Type
- Approval Profile
- Asset Type
- Business Unit
- Checklist Type
- Cost Adjustment Type
- Cost Center
- Distribution Code
- Expense Code
- Labor Earning Type
- Location Type. This includes all classes of location types stored in the "Node Type" maintenance object, for example, asset location types, inventory location types, etc.
- Planner
- Question
- Specification
- Service Category
- Service Class
- Service Code
- Service History Type
- Work Category

- Work Class

Shared Access

Security by owning organization is optional but when it is enabled the application assumes every record on a directly secured entity is assigned with an owning organization value.

When records are missing a value, the application cannot enforce data restriction and therefore lets these records be accessible to all. While the application treats these records as unrestricted, it assumes they are only temporarily unrestricted until a user or some default rule assigns them with a value.

The correct approach for intentionally sharing records across owning organizations is to designate one or more shared access groups and assign those to appropriate users. For example, if some entities are to be shared between the north and south owning organizations you may want to designate a “north-and-south” owning organization, assign shared entities to this owning organization, and allows users from both locations to have access to it.

Default Owing Organization Assignment Rules

When owning organization security is enabled, entities that are directly secured need to have their owning organization field populated either by the user or process that created them or in absence of an explicit value by a default rule. The application leverages an **Audit** algorithm configured on the entity’s business object to assign it with a default value. While a common default rule for administrative entities is to default the owning organization from the user’s default access group, more complex rules are called for when assigning values for master and transactional entities. To find the rule used for a directly secured entity, find the corresponding audit algorithm configured on its business object.

When creating a new administrative entity that is directly secured by owning organization, the user’s default access group is presented on the user interface prior to saving the record, allowing the user to change the value if needed before creating the record. Note that if the user deliberately clears the defaulted value the application still enforces the default rule and the record would be saved with the user’s default access group.

Default rules for master and transactional objects are more complex and as such need to support a way for your organization to customize them to meet your business requirements. Therefore, the user’s default access group is intentionally not defaulted on the user interface before creating such an entity, allowing the business object audit rule to populate the value as needed.

By default, product provided algorithms for defaulting an entity’s owning organization only resort to using the user’s default access group when a record is created. However, they all support an algorithm parameter that you may configure to enforce defaulting from the user’s default access group also when existing records with no owning organization value populated are updated.

NOTE: If after applying the default rules of any object an assignment has not been made, i.e. the record still has an empty owning organization field, the application is not able to restrict access to this record and therefore leaves it allowed to all. This should not be confused with a configuration that intentionally associates entities with a shared owning organization.

Explicit Owing Organization Assignment Rules

There are times when the appropriate owning organization value can not be determined by the entity’s default rule. In these unique situations, the process that creates the secured entity pre-populates it with the appropriate owning organization value when adding it. The following are examples of such processes:

- When generating preventive maintenance work orders, the application inherits their owning organization from the asset they are created for. The asset is referenced on the activity and therefore a default rule for the work order would not have access to the asset at creation time.
- When generating or interfacing other direct charges, the application inherits their owning organization from their activity’s work order. Here too, the activity is only referenced at the detail record and not available when the charge header record is created..

- When generating follow-up work orders based on service history type configuration, the application inherits their owning organization from the service history's activity's work order.

Refer to the relevant algorithms associated with such processes for more information.

Creating Entities via Integration or in Batch

Typically, the user used to process web service calls or execute batch processes is a common user record that is not affiliated with any owning organization. As such, entities created by this common user should not rely on the user record's default access group for a proper owning organization assignment. For that reason, these common users should not have any default access group defined for them, yet they should be allowed access to all access roles.

Directly secured objects created via web services or in batch are designed to have their value explicitly provided and not rely on the user's default value. For example, a user creates an asset online they can either explicitly populate the asset owning organization or let the application default it to the user's default access group. However, when assets are interfaced from an external system their owning organization should be explicitly populated.

Filtering By Owing Organization

Secured entities, either directly or indirectly, may support an owning organization filter on their respective query portals. The filter allows the user to review data that belongs to a specific owning organization, yet this should not be confused with data restriction. When security by owning organization is enabled, the application restricts the data the user can access to regardless of whether they have used the filter or not. If the user has access to more than one owning organization then a filter may be useful to narrow down the results to a selected group. However, when security by owning organization is not enabled, data is unrestricted and the user may access any record.

Enabling Data Security By Owing Organization

The following points highlight the overall steps needed to enable data security by owning organization customers:

- Design and configure access groups and access roles to meet your business requirements. If some entities should be shared across all owning organization, define a customer shared access group that is linked to all access roles.
- Associate users with access groups and roles:
 - All user records for individuals accessing the application must be assigned with a default access group and may belong to one or more access roles. This access group is used as the default value assigned to some secured objects by various business rules.
 - All user records used for integration and batch processes should not be associated with a default access group but must have access to all access roles. This way data created by these shared user records are not stamped with an arbitrary owning organization yet all data is accessible to these processes.
- For upgrading customers, make sure existing records in the primary tables of directly secured entities have the correct value in the owning organization field. Depending on the volume of each table and the number of records that may need correction in each table you may choose to make these updates online or via a custom plug-in driven batch process.
- Run the **Apply Access Group VPD Policy (W1APLVPD)** batch process to enforce row level security on master and transactional objects that are directly secured by owning organization. Refer to the batch control definition for more information about this batch process.

Three Decimals Places for 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. The currency code will 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.

Oracle Utilities Work and Asset Management supports up to three digits after the decimal for currency.

To configure the system for three decimal currency, set the number of decimal positions when defining currency codes.

Refer to the [Defining Currency Codes](#) section in the **Oracle Utilities Application Framework** documentation for more information.

Defining Colors

Colors can be used to represent transactions and other entities in various portals. For example, colors can be used to distinguish cost categories as they are represented in graphs and pie charts.

The **Color** portal is used to view and maintain colors.

You can access this portal from **Admin**, select **C**, then select **Color**.

The following zones may appear as part of the portal's **Main** tab page:

- **Color List:** This zone lists all the colors. Broadcast a record to display the details of a selected record.
- **Color:** This zone provides information about the selected color.

Documents

Understanding Documents

The product supports several classes of documents that can assist users in performing their work. This section describes topics related to managing and referencing documents.

Attachments

Any electronic file type, such as Word, PowerPoint, JPEG and so on can be uploaded into the system as an attachment. Refer to [Attachment Overview](#) for more information about file attachments.

An attachment may also reference a URL to an externally stored document. When using this type of attachment, the content of the document is not uploaded into the system. Instead, only the link to the document is captured on the attachment record.

Attachments, both *Owned* and *Referenced*, are consolidated into a single zone. A user can click the attachment hyperlink to navigate directly to the attachment record or click the **View** button to launch the file or URL associated with the attachment.

Reference Attachments

When an attachment relates to or describes a specific entity, it is uploaded to the system as owned by that entity. For example, a picture of an asset is uploaded to the system as owned by the specific asset record. These attachments are typically uploaded and maintained on the portal of the owning entity.

An attachment that is owned by one entity may be listed for reference on another entity. For example, you may reference an asset's owned attachment on a work activity planned for that asset. These attachments are referred to as reference attachments. Note that such reference is not a copy of the original attachment but a link to it. When making a change to the content of the attachment being referenced, the changes are reflected everywhere it is referenced.

While many entities support the ability to capture their own attachments, only selected entities support the ability to reference attachments of other entities.

A Document Record Represents a Folder of Attachments

The **Document** entity may be used to organize multiple related attachments into folders when applicable. Each document record represents a folder to which you may upload multiple file attachments and add URL attachments. These attachments are owned by the document record and can be referenced elsewhere as needed.

Standard Notes

Standard notes allow you to create boilerplate text, a note, which can be attached to records wherever a note might be needed to provide additional information.

Notes are displayed in the Standard Notes zone of selected entities, such as stock items, blank contracts, purchase orders, and requisitions.

Configuring Document Options

Follow the steps described in this section to configure document options.

Defining Document Types

The **Document Type** portal is used to view and maintain document types.

Refer to [Understanding Documents](#) for more information.

You can access this portal from **Admin**, select **D**, then select **Document Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Document Type List:** This zone lists all the document types. Broadcast a record to display the details of a selected record.
- **Document Type:** This zone provides information about the selected document type.

Defining Documents

A document record includes basic details about the document, such as the document number, document type, and so forth.

Refer to [Understanding Documents](#) for more information.

The **Document** portal is used to view and maintain documents.

You can access this portal from **Menu**, select **Resource Management**, then select **Document**. You are brought to a query portal with options for searching for a specific document. Once the document has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Document:** This zone displays information about the selected record.
- **Attachments:** This zone lists all the attachments that are related to the selected record.

Defining Standard Notes

A standard note is a text document that can be associated to records wherever additional information is needed. For example, "Ensure that payment terms are clearly visible on the Purchase Order report."

Refer to [Understanding Documents](#) for more information.

The **Standard Notes** portal is used to view and maintain standard notes.

You can access this portal from **Admin**, select **S**, then select **Standard Note**. You are brought to a query portal with options for searching for a specific standard note. Once the standard note has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Standard Note:** This zone displays information about the selected record.

Organizations

Understanding Organizations

Organizations are used for categorization and reporting for locations. For example, you may want to report on only the districts in a certain area or for a particular service. The reporting structure would involve the hierarchy that groups locations by a structure of parent and child organizations.

Refer to [About Asset Locations](#) for more information.

Organizations are defined by the following:

- **Organization Type:** Identifies the types of organizations (Division, Group, and so on) The organization type record identifies valid parent types within the organization. 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.

NOTE: Use caution in creating too many Organization Types as it can become restrictive when placing your Organizations within the hierarchy.

- **Organization Disposition:** The current status of the organization (Active or Inactive).
- **Parent Organization:** The parent organization, if applicable.
- **Main Contact:** A contact for the organization.

Understanding Contact Types

Contact types define the attributes of a specific type of contact. Some examples of contact types might be a *Person* contact type where the person is a residential customer or a passerby, a *Business* contact type where the business contact is the point of contact when interacting with a business, or an *Employee* contact type when receiving information from a coworker.

Refer to [Contacts](#) for more information.

Configuring Organization Options

Follow the steps described in this section to configure organizational options.

Defining Organization Types

The **Organization Type** portal is used to display and maintain organization types.

Refer to [Understanding Organizations](#) for more information.

You can access the portal from **Admin**, select **O**, and then select **Organization Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Organization Type List:** This zone lists all organization type records. Broadcast a record to display the details of the selected record.
- **Organization Type:** This zone displays information about the selected organization type.

Defining Organizations

The **Organization** portal is used to display and maintain organizations.

Refer to [Understanding Organizations](#) for more information.

You can access the portal from **Admin**, select **O**, and then select **Organization**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Organization:** This zone displays information about the organization.
- **Child Locations/Organizations:** This zone lists the child locations or organizations associated with the current record.

The following zones may appear as part of the portal's **Tree View** tab page:

- **Organization Hierarchy:** This zone displays a hierarchy view for the organization.

Configuring Contact Options

Follow the steps described in this section to configure contact options.

Defining Contact Types

The **Contact Type** portal is used to display and maintain contact types.

Refer to [Understanding Contact Types](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Contact Type**. You are brought to the contact type list page.

The following zones may appear as part of the portal's **Main** tab page:

- **Contact Type List:** This zone lists all contact types. Broadcast a record to display the details for the selected record.
- **Contact Type:** This zone displays information about the selected contact type.

Compliance

Understanding Compliance

Compliance is work that has established guidelines or financial incentives to ensure the work gets completed within agreed upon timeframes. Compliance work can also manage timeframes for the rechecking and resolving of issues.

Refer to [Understanding Preventive Maintenance](#) for more information on compliance.

This section describes various compliance features.

Preventive

Preventive compliance refers to work which has guidelines established by regulatory agencies, service level agreements, and manufacturer warranties. These guidelines would include a compliance date that must be met to stay in compliance.

Preventive work usually refers to routine maintenance type work on assets to prevent them from failing. An example of this type of work includes routine and periodic inspections.

Refer to [Understanding Preventive Maintenance](#) for more information.

Corrective

Corrective compliance manages acceptable timeframes for follow-up work resulting from issues that were found during an originating call. The process is driven through the use of service histories which is used to define and create the types of work, including follow-up work, that enables the business to remain in compliance.

Some types of issues that may require follow-up work are repairs or rechecks to water mains after a break or for gas lines after a leak.

Follow-up work is automatically generated using the compliance date definitions on the general service history type. A *Calculate Compliance Date* algorithm is used to calculate the compliance date.

Configuring Compliance Options

Follow the steps described in this section to configure compliance options.

Defining Compliance Categories

The **Compliance Category** portal is used to maintain compliance categories. A compliance category represents a broad category of compliance work, such as leaks, inspections, and preventive maintenance.

Refer to [Understanding Compliance](#) for more information.

You can access this portal from **Admin**, select **C**, then select **Compliance Category**. The user is taken to the search query portal with options for searching for a specific compliance category.

The following zones may appear as part of the portal's **Main** tab page.

- **Compliance Category:** This zone displays information about the selected compliance category.

Defining Compliance Types

The **Compliance Type** portal is used to maintain compliance types. A compliance type represents a broad category of compliance work, such as annual compliance, leak repairs, and regulator inspections.

Refer to [Understanding Compliance](#) for more information.

You can access this portal from **Admin**, select **C**, then select **Compliance Type**. The user is taken to the search query portal with options for searching for a specific compliance type.

The following zones may appear as part of the portals **Main** tab page.

- **Compliance Type:** This zone displays information about the selected compliance type, such as Compliance Date Parameters, Compliance Date Adder, and Preventive Maintenance Interval information.
- **Unlink PM Triggers:** This zone removes the compliance type from the preventive maintenance triggers.

- **Link PM Triggers:** This zone updates preventive maintenance triggers with the compliance type.

NOTE:

The Unlink and Link PM Triggers zones are applicable and available only when *Preventive Maintenance*, NOT *Corrective Maintenance*, is defined for the **Compliance Class** field.

Service History

Understanding Service History Types

Service history types define the main attributes for service history of a certain type. Some examples of service history types might include maintenance, inspection, service, test, measurement, and so on.

Service history types are defined by the following:

- **Service History Business Object:** The business object used to create service history records of this type.
- **Category:** The category to which service history records of this type belong. The base package includes the following categories:
 - Downtime
 - Failure
 - Inspection
 - Maintenance
 - Repair
- **Follow-up Actions:** Enables work orders and work activities to be generated as follow-up actions to the original work order or activity.
- **Service Schedules (ODM Only):** One or more template activity generators and time intervals that should be used to schedule field activity generation for service history records of this type.

Failure Profiles

Failure profiles group failure types, failure modes, failure repairs, and failure components into a profile which can be attached to an asset type to define valid failure information for assets of that type. Effective use of failure profiles can help to ensure that assets run without fail, increase cost effectiveness, improve machine uptime, and help your organization to understand the level of risk you have undertaken with certain assets.

Generally the process includes the following:

- Define Failure Modes Effects and Critically Analysis:
 - 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.
- Define the following according to your business practices:

- **Failure Cause:** Describes what caused the failure. For example, too much exposure to sunlight, too little or too much lubrication, too much moisture, and so on.
- **Failure Mode:** Describes the result of the failure to the asset, such as broken, bent, corroded, and contaminated.
- **Failure Repairs:** Indicates how the failure will be corrected. For example, adjusted, repaired, rebuilt, or cleaned.
- **Failure Component:** Indicates the part of the asset which has failed.
- 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.

Questionnaires

Questionnaires enable a company to assess the condition of an asset.

See [Understanding Questionnaires](#) for more information.

Questionnaire 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.

NOTE: Also note, this question and answer format supports the determination of a rating or score as to the overall condition of the asset.

Questionnaires also support a "fill in the blanks" format. A similar question to that above might appear as follows with "fill in the blanks":

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.

NOTE: The "fill in the blanks" format, unlike the question and answer method mentioned above, does not support a rating for the overall condition of the asset.

Follow Up Work

The service history type can be configured to automatically generate follow-up actions using work orders or work order activities.

For example, if you have a Service History Type of "Grade 2 Gas Leak", users could define a follow-up action of "Recheck" which could be configured with a "30 day" Compliance Type and a follow-up action of "Repair" which could be configured with a "1 year" Compliance Type. Note that you will need to specify a compliance type if a regulatory compliance requirement is associated with your follow-up action.

Refer to [Understanding Compliance](#) for additional information.

Understanding Questionnaires

Questionnaires can be created to guide workers through corrective and preventive inspection processes.

During the completion process, workers are guided through a series of questions assessing the condition of the asset. The answers can be “weighted”, which when calculated, gives an overall score as to the condition of the asset. The calculation is generally a sum (answer points * question weight), but can vary depending on the service history type. The total inspection points are then normalized to an overall grade or score. The condition rating and confidence rating are also calculated and the asset is updated with the appropriate values.

Questionnaires are associated to a service history type which is then associated to an asset type. The Questionnaire Service History Type can also be defined to automatically generate follow up work activities using work requests or work orders.

Answers on the questionnaire can be defined as check boxes, multiple choice, or free-form text.

Configuring Service History Options

Follow the steps described in this section to configure service history options.

Defining Questions

The **Question** portal is used to display and maintain condition assessment questions.

You can access the portal from **Admin**, select **Q**, then select **Question**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Question**: This zone provides information on question and answers for conditional assessments.
- **Service History List**: This zone displays a list of associated service history types.

Defining Failure Causes

The **Failure Cause** portal is used to display and maintain failure causes.

You can access the portal from **Admin**, select **F**, then select **Failure Cause**.

The following zones may appear as part of the portal's **Main** tab page:

- **Failure Cause List**: This zone lists all failure cause records. Broadcast a record to display the details of the selected record.
- **Failure Cause**: This zone provides information about the cause of the failure, such as bent, collapsed, twisted, and so on.
- **Included on Failure Profile**: This zone lists the failure profiles associated with a particular failure.

Defining Failure Modes

The **Failure Mode** portal is used to display and maintain failure modes.

You can access the portal from **Admin**, select **F**, then select **Failure Mode**.

The following zones may appear as part of the portal's **Main** tab page:

- **Failure Mode List:** This zone lists all failure mode records. Broadcast a record to display the details of the selected record.
- **Failure Mode:** This zone provides information about what has failed, such as arcing, binding, backward, and so on.
- **Included on Failure Profile:** This zone lists the failure profiles associated with a particular failure mode.

Defining Failure Components

The **Failure Components** portal is used to display and maintain failure components.

You can access the portal from **Admin**, select **F**, then select **Failure Component**.

The following zones may appear as part of the portal's **Main** tab page:

- **Failure Component List:** This zone lists all failure component records. Broadcast a record to display the details of the selected record.
- **Failure Component:** This zone identifies the part of the asset which has failed, such as instrument, electrical, mechanical drive, and so on.
- **Included on Failure Profile:** This zone lists the failure profiles associated with a particular failure component.

Defining Failure Repairs

The **Failure Repair** portal is used to display and maintain failure repairs.

You can access the portal from **Admin**, select **F**, then select **Failure Repair**.

The following zones may appear as part of the portal's **Main** tab page:

- **Failure Repair List:** This zone lists all failure repair records. Broadcast a record to display the details of the selected record.
- **Failure Repair:** This zone identifies the repair that will be made to the failed asset.
- **Included on Failure Profile:** This zone lists the failure profiles associated with a particular failure repair.

Defining Failure Profiles

The **Failure Profile** portal is used to display and maintain failure profiles.

You can access the portal from **Admin**, select **F**, then select **Failure Profile**.

The following zones may appear as part of the portal's **Main** tab page:

- **Failure Profile List:** This zone lists all failure profile records. Broadcast a record to display the details of the selected record.
- **Failure Profile:** This zone identifies the selected failure profile.
- **Failure Causes:** This zone lists the failure causes associated with this failure profile.
- **Failure Modes:** This zone lists the failure modes associated with this failure profile.
- **Failure Repairs:** This zone lists the failure repairs associated with this failure profile.
- **Failure Components:** This zone lists the failure components associated with this failure profile.

Defining Service History Types

The **Service History Type** portal is used to display and maintain service history types.

You can access this portal from **Admin**, select **S**, then select **Service History Type**. You are brought to a query portal with options for searching for a specific service history type. Once the service history type has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zone may appear as part of the portal's **Main** tab page.

- **Service History Type:** This zone provides information about the service history type, including settings for creating and stopping follow-up actions.
- **Unlink Asset Types:** This zone displays a list of associated asset types that can be removed from the service history type.
- **Link Asset Types:** This zone enables a user to search for an asset type and link it to the service history type. Linked assets will display in the **Unlink Asset Types** zone until removed.

Assets

Understanding Asset Location Types

Asset location types define the details of asset or component storage and install locations.

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

Out of service location types define attributes of types of locations where assets that are out of service are stored and maintained. Examples of out of service locations include warehouses, storerooms, and repair shops.

Out of service location types are defined by the following:

- **Class:** The general class (storeroom, meter shop, etc.) of out of service locations of this type
- **Location Business Object:** The business object used for out of service locations of this type
- **Valid Parent Location/Organization Types:** One or more valid parent locations and parent organizations that can be referenced on out of service locations of this type. Used primarily for reporting.

Location Hierarchy

Locations can be grouped according to a customer's organizational structure and displayed within the application in a hierarchy or tree view.

Understanding Asset Types

Asset types define the attributes for assets and components of a certain type including information such as the valid location types and service history types for the assets and whether or not such assets can have attached components. Also, average cost statistics are graphically displayed on the asset type record.

Asset types are defined by:

- **Component Relationship:** A flag that indicates if components can be installed on assets or components of this type.
- **Asset Business Object:** The business object used for assets or components based on this asset type.
- **Valid Location Types:** A list of location types where assets or components of this type can be located.
- **Valid Attached To Asset Types:** A list of valid asset types where components of this type can be attached (applicable to components only).
- **Valid Service History Types:** A list of valid service history types that can apply for assets or components of this type.

Refer to [Defining Asset Types](#) for more information about setting up asset types.

Asset Classes

Assets are defined by classes, each of which is based on a specific business object, and determines the type of asset: an asset or a component.

The table below lists the asset classes and the business object for each.

Asset Class	Business Object
Asset	W1-AssetType (Operational Device Asset Type)
Component	W1-ComponentType (Operational Device Component Type)

Valid Service History Types

Valid Service History Types are defined on the asset type record and describes the valid service history types that apply to a specific asset or component created from this asset type.

Valid Attributes

For simple use cases where additional details about an asset need to be captured and these details vary by asset type, you may define these details as asset-related characteristic types and associate them with each asset type as needed. With this simplified extension method, the base asset BOs may be used as is without any schema change.

Implemented as a characteristic, the value of such asset attributes may only be of a free text or restricted to a predefined list of characteristic values.

Use the following steps to guide configuration:

- Consider what attributes you want to have recorded for each of your asset types. For example, Pole asset type attributes might include Set Year, Material, Material Class, and Current Height.
- Define a characteristic type of the appropriate type value. Use a Predefined Value Characteristic Type for a list of valid values and an Adhoc Value Characteristic Type for a simple text value. Link the Characteristic Type to the Asset Attributes (W1-AssetAttributes) entity.

- Associate each attribute with the asset types it is applicable to. Assets created from the asset type will display the list of asset attributes in the Asset Attributes zone on the Asset record.

NOTE: Asset Attributes zone will only display on the Asset record if the Asset Type it was created from includes at least one asset attribute.

- Consider whether your business process requires validation algorithms to be plugged in on the asset business object to further validate or enforce other business rules related to these asset attributes.

Valid Measurement Types

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 taken for that asset. An **Asset** can have many measurement types, but the measurement identifiers of the asset must be unique. In other words, if two measurement types share the same measurement identifier, then they will not be allowed on a single asset, at the same time. This rule enforces the notion that when a user enters data, or when a report queries data, there will never be any overlap in measurement history.
- 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 types can be configured to trigger to dos or automatically create work when readings are recorded that fall inside or outside of defined tolerances.

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.

Measurement Identifiers

Measurement identifiers establish the units used to measure assets. Each measurement type is meant to represent a unique combination of Unit of Measure, Time of Use, and a Special Identifier to uniquely describe the set of measurement readings taken for an asset. This might include tire pressure (psi), temperature ratings (degrees), daytime usage (kWh) (peak), and so on. The special identifier is used to uniquely describe unusual measurement configurations, like pressure input vs. pressure output (psi input, psi output) or even gas compositions (% argon, % helium)

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.

Please note the following with regard to measurements:

- Configure the extendable lookup for **Gauge Reason** as needed.
- Measurement readings are entered in the **Measurement** maintenance portal via the **Measurement Quick Entry** zone.

Condition Assessment Class

Assessment classes are used to group and manage a set of similar assets (for example, bridges can be an assessment class and hydrants can be another assessment class, and so on) using their asset type. Each assessment class is managed 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.

An assessment group identifies a grouping of classes in order that the assets associated with those classes can be scored and weighted against other assets of that type. Service history types defined specifically for condition assessments can also be defined on the assessment group record and associated to the assessment class.

Understanding Functional Process Types

Functional process types categorize functional processes of a certain type. Examples of functional process types might include a grouping of asset locations that contribute to Corrosion, Waterfront tanks, waste water, or HVAC.

Refer to [About Functional Processes](#) for more information.

Defining Functional Process Types

The **Functional Process Type** portal is used to display and maintain functional process types.

You can access the portal from **Admin**, select **F**, then select **Functional Process Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Functional Process Type List:** This zone lists all functional process type records. Broadcast a record to display the details of the selected record.
- **Functional Process Type:** This zone provides general information about the selected functional process type.

Refer to [Understanding Functional Process Types](#) for more information.

Understanding Configuration Types

Configuration types define the information required for various asset configurations with other assets, components, firmware, and other data.

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.

Configuration types are defined by the following:

- **Configuration Class:** The class for configurations of this type.
- **Configuration Category:** The category of configurations of this type.
- **Configuration Business Object:** The business object used for configurations of this type.

Configuration Classes

In general configuration classes define the types of connections that are being configured. Each configuration class is based on a specific business object, which is the business object used for configurations of each class and type.

The table below lists the configuration classes and the business object for each.

Configuration Class	Business Object
Single Asset	W1-SingleAssetConfig (Single Asset Configuration)
Asset/Component	W1-AssetCompConfig (Asset/Component Configuration)
Asset to Asset	W1-AssetToAssetCompConfig (Asset to Asset/Component Configuration)
Asset/Component to Asset	W1-AssetCompToAssetConfig (Asset/Component to Asset Configuration)
Asset to Asset/Component	W1-AssetToAssetConfig (Asset to Asset Configuration)
Asset/Component to Asset/Component	W1-AssetCompToAssetCompConfig (Asset/Component to Asset/Component Config)

Configuring Asset Options

Follow the steps described in this section to configure asset options.

Defining Measurement Identifiers

The **Measurement Identifier** portal is used to display and maintain measurement identifiers.

You can access the portal from **Admin**, select **M**, then select **Measurement Identifier**.

The following zones may appear as part of the portal's **Main** tab page:

- **Measurement Identifier List:** This zone lists all measurement identifier records. Broadcast a record to display the details of the selected record.
- **Measurement Identifier:** This zone provides information about the selected measurement identifier.

Defining Measurement Types

The **Measurement Type** portal is used to display and maintain measurement types.

You can access the portal from **Admin**, select **M**, then select **Measurement Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Measurement Type List:** This zone lists all measurement type records. Broadcast a record to display the details of the selected record.
- **Measurement Type:** This zone provides information about the selected measurement type, including upper and lower threshold limits and inner and outer threshold exceeded details.

Defining Asset Location Types

The **Asset Location Type** portal is used to display and maintain asset location types.

You can access this portal from **Admin**, select **A**, and then select **Asset Location Type**.

The following zones may appear as part of the portal's **Main** tab page.

- **Asset Location Type List:** This zone lists all the asset location types. Broadcast a record to display the details for the selected record.

- **Asset Location Type:** This zone provides information about the selected asset location type, including valid breaker location types and valid runtime source location types information.

Defining Out of Service Location Types

The **Out of Service Type** portal is used to display and maintain out of service location types.

You can access the portal from **Admin**, select **O**, and then select **Out of Service Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Out of Service Location Type List:** This zone lists all out of service location type records. Broadcast a record to display the details of the selected record.
- **Out of Service Location Type:** This zone provides information about the selected out of service location type.

Defining Out of Service Locations

The **Out of Service Location** portal is used to display and maintain out of service locations.

You can access the portal from **Admin**, select **O**, and then select **Out of Service Location**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Out of Service Location:** This zone provides information about the out of service location, such as address information, field information and contacts.
- **Current Activities:** This zone lists the current activities associated to the location.
- **Asset Summary:** This zone displays statistical information about assets currently associated with the location.
- **Attachments:** This zone enables users to manage all types of attachments for the specific record.

Defining Condition Assessment Groups

The **Condition Assessment Group** portal is used to display and maintain condition assessment groups.

You can access the portal from **Admin**, select **C**, then select **Condition Assessment Group**.

The following zones may appear as part of the portal's **Main** tab page:

- **Condition Assessment Group List:** This zone lists all condition assessment group records. Broadcast a record to display the details of the selected record.
- **Assessment Group:** This zone provides information about the selected condition assessment group, such as aging factors and service history types.

Defining Condition Assessment Class

The **Condition Assessment Class** portal is used to display and maintain condition assessment classes.

You can access the portal from **Admin**, select **C**, then select **Condition Assessment Class**.

The following zones may appear as part of the portal's **Main** tab page:

- **Assessment Class List:** This zone lists all condition assessment class records. Broadcast a record to display the details of the selected record.

- **Assessment Class Display:** This zone provides information about the selected condition assessment class, such as associated assessment groups, status, and status date and time.

Defining Condition Assessments

The **Condition Assessment** portal is used to display and maintain condition assessment records.

You can access the portal from **Menu**, select **Condition Assessment**, then select **Condition Assessment**. You are brought to a query portal with options for searching for a specific assessment. Once an assessment has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Service History:** This zone displays details, such as consumable information, PM anniversary information, and effective dates, about the selected assessment.
- **Attachments:** This zone enables users to manage all types of attachments for the condition assessment record.

Defining Asset Types

The **Asset Type** portal is used to display and maintain asset types.

You can access the portal from **Admin**, select **A**, then select **Asset Type**. You are brought to a query portal with options for searching for a specific asset type. Once the asset type has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Asset Type:** This zone provides information about the asset type, including various attribute types that can be associated with an asset and a list of locations where assets and components of this type can be attached.
- **Asset Type Statistics:** This zone displays a graphical breakdown of cost and count details by service class.

Refer to [Service Class](#) for more information on service classes.

Configuring Activity Generation Configurations

Follow the steps described in this section to configure activity generation configurations.

Defining Configuration Types

The **Configuration Type** portal is used to display and maintain configuration types.

You can access the portal from **Admin**, select **C**, then select **Configuration Type**. You are brought to a query portal with options for searching for a specific configuration type. Once the configuration type has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Configuration Type List:** This zone lists all configuration type records. Broadcast a record to display the details of the selected record.
- **Configuration Type:** This zone displays required information for various asset configurations, such as asset to asset, asset to components, and so on.

Defining Configurations

The **Configuration** portal is used to display and maintain configurations.

Refer to [Understanding Configuration Types](#) for more information.

You can access the portal from **Admin**, select **C**, and then select **Configuration**. You are brought to a query portal with options for searching for a specific configuration. Once the configuration has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Configuration:** This zone provides information about the configuration, such as asset and specification details.
- **Attachments:** This zone displays attachments related to the specific record.

Defining Configuration Reports

The **Configuration Report** portal is used to display and maintain configuration reports.

Refer to [About Operational Devices](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Configuration Report**. You are brought to a query portal with options for searching for a specific configuration report. Once the configuration report has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Configuration Report:** This zone provides report criteria about the selected configuration report.

Operational Device Management

Understanding Operational Device Management

Oracle Utilities Operational Device Management provides functionality to handle large volumes of assets and to manage the receipt, installation, maintenance, tracking, and removal of those assets. Oracle Utilities Operational Device Management is purchased as a separate license to further enhance Oracle Utilities Work and Asset Management. However, since these two products are typically always sold together, this guide includes seamless documentation for both products. Contact your sales representative or Oracle Support for more information.

Please refer to the [Operational Device Management](#) section in the **Business User Guide** for a conceptual overview of asset functionality.

Device 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. Refer to the [Understanding Asset Types](#) section for general information on configuring assets and components.

Asset and Component Configuration Types

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 Operational Device Management Options

Follow the steps described in this section to configure Operational Device Management options.

Defining Asset Configuration Types

The **Configuration Type** portal is used to display and maintain configuration types.

You can access this portal from **Admin**, select **C**, and then select **Configuration Type**. You are brought to a query portal with options for searching for a specific configuration type. Once the configuration type has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zone may appear as part of the portal's **Main** tab page.

- **Configuration Type List:** This zone lists all configuration type records. Broadcast a record to display the details of the selected record.
- **Configuration Type:** This zone displays required information for various asset configurations, such as asset to asset, asset to components, and so on..

Inventory

Understanding Inventory

As inventory falls and reorder becomes necessary, the Inventory system passes this information to the Purchasing system so that purchases of new inventory can be carried out. Using reorder review, the system can be configured to automatically reorder stock items or notify a specified user when the quantities reach predetermined reorder points.

Each time inventory quantity is changed, the system generates a transaction record, and you have access to this information through related logs and historical information zones.

Managing inventory means balancing the need to order new items against the available supply, maintaining an optimum inventory stock level, and stocking and ordering items for upcoming work and safety requirements.

This section describes how various processes work together to manage inventory throughout the system.

Please refer to the [About Inventory](#) section in the **Business User Guide** for procedural tasks related to these concepts.

Refer to the [Configuring Inventory Options](#) section in the **Administrative User Guide** for configuration guidelines.

Storerooms

Storerooms define the physical location where stock items are stored.

Valid storerooms might include actual storerooms, warehouses, or any other location as defined by your organization.

Storerooms can be defined in hierarchies with parent and child locations. The parent can define attributes or identifiers common to all child locations, however specific location information must be defined on each location individually. A child location cannot be in an active status if the parent location is inactive. It is also not possible to make a location inactive if it has associated assets or child locations.

Vehicles as Storerooms

Vehicles, just like buildings and warehouses, can be considered a storeroom since they store pieces of equipment that are required to perform certain types of work.

Equipment, such as meters, pumps, clamps, and so on can be stored on a vehicle and made available for work.

Shipment Deficiency

A shipment deficiency is an entry created when there is a problem with a shipment; either in the number of receipt line items that were received, damage to line items or packaging, expired items or items not working properly, and so forth.

Shipment deficiency records can be created for “material” receipt line items only. Select the **Edit** hyperlink in the **Shipment Deficiency** zone to create, edit, comment, resolve, or delete shipment deficiency details.

Once the deficiency is created, buyers are able to view the deficiencies on a variety of records and receive To Dos so they can manage and track the deficiencies as they work to resolve the issue with the vendor.

- Deficiencies are very noticeable as “Deficient” text displays in red in the info string on a variety of records; purchase order header, purchase order line, receipt line, receipt header, and also in the **Shipment Deficiencies** zone on the **Vendor Location** portal and in the **Inventory Transactions** zone on the **Availability** tab of the **Stock Item Detail**.
- A Shipping Deficiency To Do Type can be configured to generate a To Do to the buyer listed on the purchase order. This To Do includes a link that takes the buyer to the receipt line that has the deficiency. The deficiencies can then be edited, added to, or resolved directly from the **Shipment Deficiency** zone on the **Receipt Line** portal. Definition to automatically generate this To Do is defined in the **Receiving Parameter** zone of the **Master Configuration**.

NOTE: Please note that the buyer must be associated with the To Do Role defined for the Shipping Deficiency To Do Type.

Configuring Inventory Options

Follow the steps described in this section to configure inventory options.

Defining Storeroom Types

The **Storeroom Type** portal is used to display and maintain storeroom types.

Refer to [Understanding Inventory](#) for more information.

You can access the portal from **Admin**, select **S**, then select **Storeroom Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Storeroom Type List:** This zone lists all storeroom type records. Broadcast a record to display the details of the selected record.
- **Storeroom Type:** This zone provides information about the selected storeroom type.

Defining Storerooms

The **Storeroom** portal is used to display and maintain storerooms.

Refer to [Understanding Inventory](#) for more information.

You can access the portal from **Admin**, select **S**, then select **Storeroom**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Storeroom:** This zone provides information about the storeroom, such as address information, accounting information, reorder information, and so on.
- **Current Activities:** This zone provides summary information about activities that are currently associated with this location.
- **Asset Summary:** This zone displays statistical information about the assets associated with this location.
- **Attachments:** This zone enables users to manage all types of attachments for the storeroom record.

Defining Shipment Deficiency Types

The **Shipment Deficiency Type** portal is used to display and maintain shipment deficiency types.

Refer to [Understanding Inventory](#) for more information.

You can access the portal from **Admin**, select **S**, then select **Shipment Deficiency Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Shipment Deficiency Type List:** This zone lists all shipment deficiency type records. Broadcast a record to display the details of the selected record.
- **Shipment Deficiency Type:** This zone provides a description of the selected shipment deficiency type.

Purchasing

Understanding Purchasing

Purchase requisitions can be created directly from the **Menu**, through work orders, or through the review of storeroom stocked items.

Following a typical procurement life cycle, a requested item or service is listed on a requisition. Once reviewed and approved, if necessary, the requisition is copied into a purchase order. The purchase order is approved and issued to a vendor, and the requested item is received and inventoried. Throughout the process, different areas within the application are automatically maintained such as storeroom quantities and values depending on the type of item being purchased.

If changes are required for a purchase order that has already been approved, a revision can be created within the purchase order. The system also facilitates expediting purchase orders and full or partial receipt of items.

This section describes how various processes work together to manage purchasing throughout the system.

Refer to the [Purchasing](#) section in the **Business User Guide** for procedural tasks related to these concepts.

Refer to the [Configuring Purchasing Options](#) section in the **Administrative User Guide** for configuration guidelines.

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.

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, and so on. When items of that type are needed, the purchase commodity helps you to identify the buyer with that specialty.

Vendors

Each main Vendor record provides general information about your relationship with the vendor such as active blanket contracts, open requisitions, purchase order history, and a list of vendor locations.

Vendor Locations

One or many vendor locations are defined for each vendor to distinguish between multiple vendor sites. Each location provides more detailed information about the vendor site such as address, capabilities, payment details, contacts, and so on.

For example, it may be valuable to understand which location is shipping the goods that you have ordered from the vendor. Also, some vendors might ship from one site, but your payment terms are set with an alternate site. Capabilities would provide information such as when one location offers pick up at that location whereas another location may not. Commodities describe the types of commodities, such as mechanical, electrical, plumbing and the like, that are associated with a particular vendor at a specific vendor location.

When a vendor has multiple locations, alternate locations are listed as “related” locations.

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.

Freight on Board (FOB)

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.

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.

Understanding Invoices

Please note the following with regard to invoicing:

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.

Automatic Invoicing

In order for the system to complete automatic invoicing against purchase orders, the vendor on the purchase order must be set as an “auto pay” vendor. When the purchase order is created, it must also be set as “auto pay”. After items are received against the purchase order, the system automatically creates an invoice with line items. When an invoice is automatically created, no additional line items can be added to the created invoice and none of the line items can be deleted. If the receipt included more items received than ordered, these invoice line items are placed in “Pending Resolution” status when the invoice is submitted for approval. An authorized user must then change the quantity on the purchase order before the invoice can be reprocessed successfully.

Note that when **Require Receipt for Services** is set to “No” in Purchasing Master Configuration, users cannot create a receipt against a purchase order which was created for services. This also affects automatic invoicing in that when this is set to "No", invoices for services must be created manually even when the vendor is set as an "Auto Pay" vendor.

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 **FI_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’.

If invoice is owned by your ERP system, there is a productized integration to transfer WAM related invoices into WAM, please refer to the ERP integration documentation for more information.

Tax Rate Schedule Selection

A Tax Rate Schedule defines the various tax components for each type of tax (such as state, city, and local tax) that get applied to materials, labor, equipment, and other resource requirements at various instances in times. The tax components for each type of tax identifies the effective date, the sequence in which the tax component is applied, and the factor associated with the specific tax component. If taxes are paid directly to a taxing authority, a cost center and expense code should also be defined.

The actual tax rate depends on the taxing state or taxing city characteristics on the invoice line itself. If the invoice line's taxing state is California, then CA's taxing rate will be applied. But if the invoice line's "Taxing State" is blank, it means state tax is not applicable.

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 one **characteristic type** with an **entity flag** of **Tax Component** and one **characteristic type** with an **entity flag** of **Factor**.

This ensures that the characteristic type will be included on the **Factor Characteristic Types** drop down on **Factors**.

- Introduce a flattened characteristic element for the characteristic type in the Invoice Line business object.

In order to include these elements in the base business object, the corresponding Invoice Info data area included in the business object 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

		Characteristic Entity	Owner
+	🗑️	Factor	Customer Modification
+	🗑️	Invoice Line	Customer Modification
+	🗑️	Purchase Order Line	Customer Modification
+	🗑️	Purchase Requisition Line	Customer Modification
+	🗑️	Tax Component	Customer Modification

Configuring Purchasing Options

Follow the steps described in this section to configure purchasing options.

Defining Blanket Contract Access Lists

The **Blanket Contract Access List** portal is used to display and maintain blanket contract access lists.

Refer to [Understanding Purchasing](#) for more information.

You can access the portal from **Admin**, select **B**, then select **Blanket Contract Access List**.

The following zones may appear as part of the portal's **Main** tab page:

- **Blanket Contract Access List:** This zone lists all blanket contract access records. Broadcast a record to display the details of the selected record.
- **Blanket Contract Access List Display:** This zone displays a list of authorized users for the list being created.

Defining Buyers

The **Buyer** portal is used to display and maintain buyers.

Refer to [Understanding Purchasing](#) for more information.

You can access the portal from **Admin**, select **B**, then select **Buyers**.

The following zones may appear as part of the portal's **Main** tab page:

- **Buyer List:** This zone lists all buyer records. Broadcast a record to display the details of the selected record.
- **Buyer:** This zone displays various information about the buyer, such as email address and primary phone number.

Defining Commodity Codes

This portal is used to view and maintain commodity codes.

Refer to [Understanding Inventory](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Commodity Code**.

The **Commodity Code Hierarchy** zone on the portal's **Main** tab page displays a hierarchical view of all commodity codes.

Defining Purchase Commodities

The **Purchase Commodity** portal is used to display and maintain purchase commodities.

Refer to [Understanding Purchasing](#) for more information.

You can access the portal from **Admin**, select **P**, then select **Purchase Commodity**.

The following zones may appear as part of the portal's **Main** tab page:

- **Purchase Commodity List:** This zone lists all purchase commodity records. Broadcast a record to display the details of the selected record.
- **Purchase Commodity:** This zone displays information about the purchase commodity, such as the default buyer.

Defining Capability Type

The **Capability Type** portal is used to display and maintain capabilities.

Refer to [Understanding Purchasing](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Capability Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Capability Type List:** This zone lists all capability type records. Broadcast a record to display the details of the selected record.
- **Capability Type:** This zone displays information about the capability type.

Defining Free on Board

The **Free on Board** portal is used to display and maintain free on board records.

Refer to [Understanding Purchasing](#) for more information.

You can access the portal from **Admin**, select **F**, then select **Free on Board**.

The following zones may appear as part of the portal's **Main** tab page:

- **Free on Board List:** This zone lists all free on board records. Broadcast a record to display the details of the selected record.

- **Free on Board:** This zone displays free on board information.

Defining Payment Terms

The **Payment Terms** portal is used to display and maintain payment terms.

Refer to [Understanding Purchasing](#) for more information.

You can access the portal from **Admin**, select **P**, then select **Payment Terms**.

The following zones may appear as part of the portal's **Main** tab page:

- **Payment Terms List:** This zone lists all payment term records. Broadcast a record to display the details of the selected record.
- **Payment Terms:** This zone displays various information about payment terms, such as discounts, fixed number of days, etc.

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.

Work Options

Understanding Activity Types

Activity types define the main attributes for activities of a certain type and act as a model and template for **Activity** records. Activity types are created in an *Active* status by default but can be toggled to *Inactive* to prevent the activity type from future use.

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. Refer to [About Work Activities](#) for more information.

This section describes various features controlled by activity type configuration.

Default Labor Requirements

A default set of labor requirements can be defined for each type of activity performed by the company. The system then uses this information to schedule activities of a certain type to the crew shift having the skills to complete the work.

Default labor requirements are defined on the activity type and include the following definitions:

- Craft which typically indicates a special skill, such as electrician, mechanic, etc.
- Quantity refers to the number of each craft that is needed to work the activity.
- Hours refer to the time requirement to work the activity for each craft.

Checklist Types

Checklist types can be used for managing regulations around permit processes or for basic procedural steps in completing work activities. Refer to [Checklists](#) for more information on checklist functionality.

General Checklist Type

Checklists of this type are procedure driven and ensures the user works a pre-defined set of steps in the right order. A checklist type can have one to many line items with each line being marked either mandatory or not. Line items on checklist type are defaulted to a checklist during the checklist's initial creation.

A checklist type also defines the following:

- The maintenance object such checklists are applicable to; for example, Activity or Permit.
- The impact such checklists should have when such a checklist is associated with a particular instance of the applicable maintenance object, for example, an activity, when that object enters certain statuses in its lifecycle and the checklist has not been completed. The impact could be no impact, warning, or stop processing (error).
- The statuses such checklists should impact.

Permit Item Only Checklist Type

Similar to the General Checklist Type but with the following exceptions:

- Each line item on this checklist type must reference a permit template. Each line item on a checklist of this checklist type refers to a permit whose permit template must match that on the corresponding line items on the checklist type.

- Applicable maintenance object is only limited to Activity in the base product.

Planned Service History Types

Users are able to specify a default list of planned service history types for an activity type. When the activity is created, not via a template, the list of planned service history types default onto the corresponding zone on the activity allowing the planner to adjust the list as needed.

Users can also specify on the activity type whether a service history type should be required or optional. If set as required, the service history would need to be completed before the activity or template activity can be closed. This is used when reviewing an asset's service history for preventive maintenance purposes.

Work Categories

A work category represents a broad grouping of work. Some examples of work categories can include Replacements, Repairs, Periodics, Vandalism, Regulatory, and Inspections.

Work categories, in conjunction with work classes, enable users to filter their work by the work category. For example, if a particular work class is selected, a user will only see the related work categories associated with that particular work class. Users are also able to generate reports on each of the work categories.

Work categories can be defined at an organization level using access groups. What this means is that only users associated with a particular organization and who are included as part of that access group will be able to see the values for the work categories associated with that branch of the organization.

Work Classes

A work class represents a broad category of work. Some examples of work classes may be planned, unplanned, or capital. Users can use Work Classes as a way to drive a specific list of available Work Categories.

Some of the main uses for work class are as follows:

- Users are able to filter their work by work class. For example, to see only planned work or capital jobs for that day.
- Reports can be generated for the individual work classes.

Work classes can be defined at an organizational level and by access group so that only users in a particular organization who are included as part of that access group will be able to see a particular work class and its related work categories.

Refer to [Work Categories](#) for more information on work categories.

Service Class

Service class is a broad categorization of activity types, allowing an activity type to belong to more than one service class. Each activity type must belong to at least one service class.

Examples of service classes include a maintenance service class for all activities associated with preventive maintenance services, an operational service class for activities associated with operations, a renewal service class for all activities associated with renewal services, and municipal service class for all activities associated with municipal services.

Some of the main usages of service classes are as follows:

- Crews may not be allowed to perform certain classes of work on specific shifts. Activities of the types included in these service classes would not be assigned by the scheduler to these shifts.
- Supervisors would be able to monitor work of a certain service class.

The scheduler uses service class when scheduling work to shifts based on their defined resource requirements. 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.

Review your activity types and group them to various service classes to meet your business requirements.

Completion Event Types

Completion event types are used to define various business events within the application. 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 and can be defined as optional or required. 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: You will need to add the completion event type to the activity type using the **Activity Type** portal in order to set the completion event types as optional or required. Also, setting a **Default Event Date Time** on the completion event type controls whether or not the completion event can be created automatically without user input.

Work activities that may be completed in the field by crews using the mobile application or by an external system require the following complete event types to be listed on the activity type:

- Update Worked Assets (W1-MobileUpdateWorkedAssets)
- Remove Tracked Asset (W1-RemoveTrackedAsset)
- Remove Non-Tracked Asset (W1-RemoveNonTrackedAsset)
- Install Tracked Asset (W1-InstallTrackedAsset)
- Install Non-Tracked Asset (W1-InstallNonTrackedAsset)
- Attach Component (W1-AttachComponent)
- Create Operational Reading (W1-CreateOperationalRead)
- Create Any Service History (W1-CrAnyServiceHistComplEvtTyp)

NOTE: Removal completion event types must be listed before installation completion even types to support proper processing of asset exchanges.

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 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)

NOTE: Attachments added here are used as part of the completion event processing and may be used to update a picture or other supporting files about an asset.

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.

Understanding Resource Requirements

This section describes various features of Resource Requirements configuration.

Crafts

Crafts, or labor, defines a skill or work specialty for crew members, such as mechanic, electrician, plumber, arborist, and so on. The actual number of labor hours required to work a task is able to be tracked using timesheets.

Every employee as a crew member is defined with a principal craft. This principal craft is used in scheduling activity labor requirements to shifts.

When principal crafts are defined, a secondary craft can also be added as part of the craft definition. Secondary crafts are regarded as additional or complementary skills to the principal craft. While secondary crafts are considered as an available craft during scheduling, their capabilities are given less importance in the schedule.

Likewise, a default set of crafts can be defined in the **Crew Type Labor Profile** zone at the crew type level. Crafts listed in this zone supersede the crafts defined for employees listed on the crew shift and total available hours for a particular craft is multiplied by the number of people.

Equipment

Equipment is another type of resource and is considered to be machinery, tools, or vehicles that are required to work activities. Examples of equipment might include trucks, bulldozers, cranes, hammers, and drills.

The actual number of equipment hours required for the task can be tracked using timesheets.

Miscellaneous Resources

Miscellaneous or “Other” resources are those resources that are not considered to be either craft, equipment, or material and which enable a crew member to work an activity. Some examples of resources that are considered “other” are meals, hotel stays, mileage, and so on.

Materials

Material is another type of resource and can be considered the stock items required for the work.

The actual number of material hours required for the task can be tracked using timesheets.

Refer to [Inventory](#) for more information about stock items.

Understanding Resource Charges

An employee may report time and equipment usage as well as miscellaneous other direct charges against an activity they have worked. The resource charges are then produced and posted to the Oracle Utilities Work and Asset Management application.

These reports are not critical to the activity completion process and as such, not considered as activity completion information. The mobile application sends a separate message for each report when the employee marks it as complete.

Completed reports are locally queued and when the device is connected, the mobile application automatically delivers them to the server.

By default, time and resource usage reports may not be entered against a completed activity unless explicitly permitted to do so on the Work Management Master Configuration.

If the employee is allowed to complete the activity they may report time for the entire crew otherwise they may only enter time for themselves.

NOTE: Time and resource usage reports initiated from the mobile application are yet to be approved as usual by an authorized user using the main application.

Labor Earning Types

Your workforce may have to perform different types of work that affect their base wage rate, such as working with hazardous materials, working in confined spaces, or being the lead person on a job. Labor earning types define these types of earning codes and include *Adder* and *Multiplier* fields so the base wage rates can be automatically adjusted.

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.

Timekeepers

Timekeepers are authorized users who are able to create timesheets or enter and edit timesheet data for one or more employees that are assigned to them.

A timekeeper record associates a user with a list of employees whom the user is responsible for entering time for, regardless of their supervisor or employee relationship.

Timekeepers are able to create and adjust hours, travel time, and a variety of other timesheet elements for an individual worker or for multiple workers at one time. Refer to [Timesheet Quick Entry](#) for more information.

Understanding Projects

This section describes various features of Project configuration.

Projects can be organized into a hierarchical structure.

Projects can be linked to many transactional records and cost centers in the system primarily for cost reporting purposes.

Projects can be configured to require approval processing.

Key data components on a project include the following:

- Estimated and actual start and end dates
- Budgeted and actual amounts
- Project supervisor

Understanding Construction Work

Configuring for this functionality involves setting up the construction work management specific objects discussed in this section.

Refer to [Construction Work Management Setup](#) for the setup sequence and prerequisites.

Property Units

Property units group capital and fixed assets that have similar depreciation characteristics. Deprecation is currently not managed within Oracle Utilities Work and Asset Management. Oracle Utilities Work and Asset Management can interface fixed assets, for example, property unit, quantity, in service date, value, and so on, to an external fixed asset management system.

CU Category

CU Category provides a categorization for compatible units.

Every CU Category must belong to a CU Class. A CU Category can be referenced by one or many CU Equipment Groups (e.g. Pipes, Poles, or Wires).

Equipment Groups

Equipment groups categorize CU Sets. Every CU Set must reference a single equipment group, however, an equipment group can have many CU Sets.

Every equipment group has a single CU Category, but a CU category can have many equipment groups.

CU Usage

Defines usage categories for compatible units. When planners add compatible units to a work design or construction work order, they can choose a CU Usage, which filters the cost centers to use for the CU. For each usage you specify the cost center that is specific for that usage.

Overhead

Overhead defines how to apply expenses against work designs and construction work orders.

There are two classes of overhead, Direct Overhead and Indirect Overhead. Direct Overhead refers to cost charged to a general Construction Overhead activity. For example, a Construction Work Order could have many actual construction work activities, but only one construction overhead activity for people to book the time they spend designing the entire construction project. Direct Overhead cost is distributed among those actual construction activities when they close.

Indirect Overhead refers to cost that's not tracked within WAM, but should be considered as part of the capital expenditure. For example, some construction work is funded by loan and/or bond, which have cost of their own. Indirect Overheads are applied as a percentage of the actual cost of the construction activities.

Compatible Units

Compatible Units define the materials, labor, and equipment necessary to install or remove assets in relationship to construction work. Compatible units can be referenced by other compatible units to build more complex configuration and allow users to properly group resources together.

Compatible Unit Type

The Compatible Unit Type determines how a compatible unit can be used:

- **Compatible Unit Base:** A simple compatible unit only consists of materials and serves as a building block for function compatible units. They cannot be selected directly on a work design and do not reference a property unit. Base compatible units are meant to model basic stock items used to build up a functional or accessory compatible unit. Changes to the stock item will be reflected on the compatible unit.
- **Compatible Unit Function:** The primary compatible unit which is added on work designs. Functional compatible units can have materials, labor and equipment and require a function to install or remove the compatible unit. They can also be defined with a condition and property unit. Functional compatible units are intended to model an asset (such as a pole) and do create fixed or physical assets.
- **Compatible Unit Accessory:** This CU is meant to be used in conjunction with other CUs and is not meant to be capitalized. An accessory can be directly selected on a work design and can be designated with a function and condition, but not a property unit. Accessory compatible units are meant to model optional asset accessories that can be added onto a work design to accompany the build out of an asset but which don't directly contribute to the asset as a whole. A CU Accessory can also inherit from a Base CU if there is a desire to abstract the stock items in the Base CU from the various ways the CU can function as a CU Accessory. Note that CU Accessories do not create fixed or physical assets.

Functional and Accessory compatible units are primarily used in two ways:

- On a work design to determine a quote, and perhaps pick between various design options,
- On a construction work order to initially populate the planned estimates and to drive the assets that will be created.

The accounting information on the compatible unit is used during the construction reconciliation process to produce the true cost of the new asset as well as the appropriate accounting codes.

About CU Sets

CU Sets group related compatible units, usually by factors such as equipment group and size. Every compatible unit must belong to only one CU Set.

The main goal of a CU Set configuration is to make it easy to add CUs to work designs by these groupings. A best practice is to ensure that a CU Set is defined for every equipment group and size combination, and to ensure that a set includes the Functional CUs and Accessory CUs together. This will ease the selection of CUs when creating a work design or construction work order.

About CU Contractor Bids

CU Contractor bids define the details of a contractor's proposal to perform the work for a particular Compatible Unit on a construction project.

Contractors must be defined in the system by a vendor location. Bids can only be added to functional or accessory compatible units and they may or may not include materials. Bids have both an effective date and an expiration date.

Activating the bid will ensure that it can be included on the construction work activity where the compatible unit is referenced.

Understanding Permit Templates

Templates expedite the permit creation process by providing a tool for defining permit classification and safety details so it can be reused on multiple work order permits.

When a planner prepares for an activity, they decide what permits are needed and then they add the permits using the checklist functionality.

Refer to [About Permits](#) for more information.

This section describes concepts related to setting up and maintaining permit templates.

Understanding Completion Events

Completion events are used to execute business events within the application. For example, completion events may be configured for several types of asset processing such as asset update, state transition, replication, service history, and activity creation.

All activities have a set of required and optional completion events that can be executed to see the activity to completion. the Relationship Information section shows any other system objects that are associated to the completed object. For example, if you are looking at the completion event for an activity, the relationship information shows the asset, any components, the location, service history, and so on.

Completion Event Common Attachments

Attachments added here are used as part of the completion event processing. For example, a completion event may be used to update a picture or other supporting file about an asset.

Processing an Activity Completion Message

Activity completion is captured in relation to an asset listed on the activity and includes the following details:

- Was any work performed on the asset?
- Was one or more measurements taken for this asset?
- Was there one or more service histories filled out for this asset?

The completion message is locally queued and when the device is connected, the mobile application automatically delivers it to the server.

Understanding Service Call

Work designs, work requests, work orders, and activities can all be created as a result of a service call.

This section describes various features of service call configuration.

Service Codes

Service categories include a list of valid service codes. These service codes drive the processing of the service call record into a work document.

Service codes can be configured to automatically create follow-up work for service calls that include a service code with one or more of the following definitions:

- To Do is created when the **To Do Type** and the **To Do Role** is defined.
- A work request is created when the **Create Work Request** field is set to “Yes”. Optionally, the **Work Request Approval Profile** can also be defined.
- A work order is created when a **Template Work Order** is defined. Optionally a **Work Order Approval Profile** can also be specified.

Refer to the [Understanding Approval Processing Configuration](#) section of the **Administrative User Guide** for more information on Approval Profiles.

Service Categories

A service category represents a broad grouping of services. Some examples of service categories can include Gas Trouble, Electric Trouble, Water Issues, Vegetation Management, and Customer Requests.

Service categories can be associated with an organization or access group so the utility can segregate the service call categories to specific users. What this means is that only users associated with a particular organization and who are included as part of that access group will be able to see the values for the service categories associated with that branch of the organization.

Configuring Resource Related Options

Follow the steps in the order they are described in this section to configure resource related options.

Defining Unit of Measure Resources

This portal is used to maintain unit of measure resources. A unit of measure is used to quantify a resource.

Refer to [Understanding Resource Requirements](#) for more information.

You can access this portal from **Admin**, select **U**, then select **Unit of Measure-Resource**.

The following zones may appear as part of the portals **Main** tab page.

- **Resource UOM List.** This zone lists all the units of measure. Broadcast a record to display the details for the selected record.
- **Resource UOM.** This zone provides information about the selected resource UOM.

Defining Crafts

This portal is used to maintain craft resources. Crafts are the labor and skill required to work an activity.

Refer to [Understanding Resource Requirements](#) for more information.

You can access this portal from **Admin**, select **C**, then select **Craft**. The user is brought to a query portal with options for searching for a specific craft. Once the craft has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zone may appear as part of the portals **Main** tab page.

- **Craft.** This zone provides information about the selected craft, including scheduling information, secondary crafts, and accounting information.
- **Hourly Rate.** This zone enables an hourly rate and effective date to be defined for the craft.
- **Craft Notes.** This zone displays any notes associated with the craft.
- **Attachments.** This zone enables users to manage all types of attachments for the craft record.

Defining Equipment

This portal is used to maintain equipment resources. Equipment is a resource required to work an activity.

Refer to [Understanding Resource Requirements](#) for more information.

You can access this portal from **Admin**, select **E**, then select **Equipment**. The user is brought to a query portal with options for searching for a specific equipment. Once the equipment has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zone may appear as part of the portals **Main** tab page.

- **Equipment.** This zone provides information about the selected equipment, including accounting information.
- **Unit Price.** This zone enables a unit price and effective date to be defined for the equipment.
- **Notes.** This zone displays any notes associated with the equipment.
- **Attachments.** This zone enables users to manage all types of attachments for the equipment record.

Defining Miscellaneous Resource

This portal is used to maintain equipment resources. Equipment is a resource required to work an activity.

Refer to [Understanding Resource Requirements](#) for more information.

You can access this portal from **Admin**, select **M**, then select **Miscellaneous Resource**. The user is brought to a query portal with options for searching for a miscellaneous resource. Once the resource has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zone may appear as part of the portals **Main** tab page.

- **Other Resource.** This zone provides information about the selected other resource, including accounting information.
- **Unit Price.** This zone enables a unit price and effective date to be defined for the other resource.
- **Notes.** This zone displays any notes associated with this other resource record.
- **Attachments.** This zone enables users to manage all types of attachments for the resource record.

Configuring Resource Charges Options

Follow the steps described in this section to configure resource charges options.

Defining Labor Earning Types

The **Labor Earning Type** portal is used to display and maintain labor earning types.

Refer to [About Timesheets](#) for more information.

You can access the portal from **Admin**, select **L**, then select **Labor Earning Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Labor Earning Type List:** This zone lists all labor earning type records. Broadcast a record to display the details of the selected record.
- **Labor Earning Type:** This zone provides information about the selected labor earning type, including adder and multiplier values.

Defining Overtime Types

The **Overtime Type** portal is used to display and maintain overtime types.

Refer to [About Timesheets](#) for more information.

You can access the portal from **Admin**, select **O**, then select **Overtime Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Overtime Type List:** This zone lists all labor earning type records. Broadcast a record to display the details of the selected record.
- **Overtime Type:** This zone provides information about the selected overtime type.

Defining Timekeepers

This portal is used to maintain timekeeper records.

Refer to [Understanding Resource Charges](#) for more information.

You can access this portal from **Admin**, select **T**, then select **Timekeeper**.

The following zones may appear as part of the portals **Main** tab page.

- **Timekeeper List.** This zone lists all timekeepers. Broadcast a record to display the details for the selected record.
- **Timekeeper.** This zone provides information about the selected timekeeper, the status of the timekeeper and the list of employees the timekeeper can enter time for.

Configuring Work Options

Follow the steps in the order they are described in this section to configure work related options.

Total Risk Lookups

Work priority and criticality custom lookups require special configuration since they can be searched by range and by greater than and less than values.

The following code descriptions for work priority and asset criticality are delivered as part of the base product:

- **Work Priority:** 09 Emergency
- **Asset Criticality:** 01 Asset Failure Low Impact

The system multiplies the location criticality by the activity work priority to derive a usable total risk value.

To this end, you must configure these lookups, `WORK_PRIORITY` and `CRITICALITY`, 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.

Defining Project Categories

This portal is used to maintain project categories. A project category represents a broad classification for projects such as maintenance, capital, construction, and so on.

Refer to [Understanding Projects](#) for more information.

You can access this portal from **Admin**, select **P**, then select **Project Category**.

The following zones may appear as part of the portals **Main** tab page.

- **Project Category List.** This zone lists all project categories. Broadcast a record to display the details for the selected record.

- **Service Class.** This zone provides information about the selected project category.

Defining Checklist Types

This portal is used to maintain checklist types. A checklist type defines the maintenance object it is applicable to, the impact it should have with its applicable maintenance object, and the statuses it should impact.

Refer to [Understanding Activity Types](#) for more information.

You can access this portal from **Admin**, select **C**, then select **Checklist Type**.

The following zones may appear as part of the portals **Main** tab page.

- **Checklist Type List.** This zone lists all permit classes. Broadcast a record to display the details for the selected record.
- **Checklist Type.** This zone provides information about the selected checklist type, including defaulted checklist items and processing constraints.

Defining Permit Classes

This portal is used to maintain permit classes. A permit class represents a broad category of permits, such as safety permits or permits for right of way.

Refer to [Understanding Permit Templates](#) for more information.

You can access this portal from **Admin**, select **P**, then select **Permit Class**.

The following zones may appear as part of the portals **Main** tab page.

- **Permit Class List.** This zone lists all permit classes. Broadcast a record to display the details for the selected record.
- **Permit Class.** This zone provides information about the selected permit class, including expiration warnings.

Defining Permit Templates

This portal is used to maintain a list of permit templates.

Refer to [Understanding Permit Templates](#) for more information.

You can access this portal from **Admin**, select **P**, then select **Permit Templates**. The user is brought to a query portal with options for searching for a specific permit template. Once the template has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zone may appear as part of the portals **Main** tab page.

- **Permit Template.** This zone provides information about the selected permit. This zone can also include information on the valid checklist types, planned service history, and additional resources, such as labor and material, included on the permit.
- **Notes.** This zone lists notes associated with the permit.
- **Attachments.** This zone displays attachments associated with the permit. Attachments added here are made available as reference information related to the permit and are available on work activity.

Defining Service Classes

This portal is used to maintain service classes. A service class represents a broad class of services.

Refer to [About Service Calls](#) for more information.

You can access this portal from **Admin**, select **S**, then select **Service Class**.

The following zones may appear as part of the portals **Main** tab page.

- **Service Class List.** This zone lists all service classes. Broadcast a record to display the details for the selected record.
- **Service Class.** This zone provides information about the selected service class, including the service class type and cost center.

Defining Work Categories

This portal is used to maintain work categories. A work category describes a broad category of work.

Refer to [Understanding Activity Types](#) for more information.

You can access this portal from **Admin**, select **W**, then select **Work Category**.

The following zone may appear as part of the portals **Main** tab page.

- **Work Category List.** This zone lists all the work categories. Broadcast a record to display the details for the selected record.
- **Work Category.** This zone provides information about the selected work category.

Defining Work Classes

This portal is used to maintain work classes. A work class represents a broad category of work.

Refer to [Understanding Activity Types](#) for more information.

You can access this portal from **Admin**, select **W**, then select **Work Class**.

The following zones may appear as part of the portals **Main** tab page.

- **Work Class List.** This zone lists all work classes. Broadcast a record to display the details for the selected record.
- **Work Class.** This zone provides information about the selected work class.

Defining Completion Event Types

This portal is used to maintain various completion event types.

Refer to [Understanding Activity Types](#) for more information.

You can access this portal from **Admin**, select **C**, then select **Completion Event Type**. The user is brought to a query portal with options for searching for a specific completion event type. Once the completion event type has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zone may appear as part of the portals **Main** tab page.

- **Completion Event Type.** This zone provides information about the selected completion event type, such as inbound communications and attachments.
- **Unlink Activity Types.** This zone enables the removal of an activity type from the completion event type.
- **Link Activity Types.** This zone adds an activity type to the completion event type.

Defining Completion Events

This portal is used to maintain various completion events.

Refer to [Understanding Completion Events](#) for more information.

You can access this portal from **Admin**, select **C**, then select **Completion Event**. The user is brought to a query portal with options for searching for a specific completion event. Once the completion event has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zone may appear as part of the portals **Main** tab page.

- **Completion Event:** This zone displays various details about the completion event, such as assets, components, locations and so on associated with the completion event, as well as any related objects and attachments.
- **Completion Event Related Objects:** This zone displays the related objects associated with the completion event.
- **Attachments:** This zone enables users to manage all types of attachments for the completion event.

Defining Service Codes

This portal is used to maintain a list of valid service codes. Service Codes are used to create the appropriate work record from the service call.

Refer to [About Service Calls](#) for more information.

You can access this portal from **Admin**, select **S**, then select **Service Code**. The user is brought to a query portal with options for searching for a specific service code. Once the service code has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zone may appear as part of the portals **Main** tab page.

- **Service Code.** This zone provides information about the selected service code and enables user functions to manage To Dos, work requests, work orders and work priorities.
- **Unlink Service Categories.** This zone enables the removal of a service code from the valid service code list of service categories.
- **Link Service Categories.** This zone adds a service code to the valid service code list of service categories.

Defining Service Categories

This portal is used to maintain service categories. A service category represents a broad category of services.

Refer to [About Service Calls](#) for more information.

You can access this portal from **Admin**, select **S**, then select **Service Category**.

The following zones may appear as part of the portals **Main** tab page.

- **Service Category List.** This zone lists all service categories. Broadcast a record to display the details for the selected record.
- **Service Category.** This zone provides information about the selected service category, including a list of valid service codes.

Defining Activity Types

This portal is used to maintain activity types. An activity type represents a broad range of activities, such as maintenance, repairs, inspections, etc.

Refer to [Understanding Activity Types](#) for more information.

You can access this portal from **Admin**, select **A**, then select **Activity Type**. The user is brought to a query portal with options for searching for a specific activity type. Once the activity type has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portals **Main** tab page.

- **Activity Type.** This zone defines the main attributes of an activity type. Activity type definitions can include some of the following: Default Labor Requirements, Activity Scheduling information, Planned Service History Completion Event Types, etc.
- **Attachments.** This zone enables users to manage all types of attachments for the activity type.

Configuring Construction Work Options

Follow the steps in the order they are described in this section to configure construction work related options.

Defining Property Units

The **Property Unit** portal is used to display and maintain property units.

Refer to [Understanding Construction Work](#) for more information.

You can access the portal from **Admin**, select **P**, then select **Property Unit**.

The following zones may appear as part of the portal's **Main** tab page:

- **Property Unit List:** This zone lists all property unit records. Broadcast a record to display the details of the selected record.
- **Property Unit:** This zone provides information about the selected property unit.

Defining Compatible Unit Categories

The **Compatible Unit Category** portal is used to display and maintain compatible unit categories.

Refer to [Understanding Construction Work](#) for more information.

You can access the portal from **Admin**, select **C**, then select **CU Category**.

The following zones may appear as part of the portal's **Main** tab page:

- **CU Category List:** This zone lists all compatible unit category records. Broadcast a record to display the details of the selected record.
- **CU Category:** This zone provides information about the selected compatible unit category.

Defining Compatible Units

The **Compatible Unit** portal is used to display and maintain compatible units.

Refer to [Understanding Construction Work](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Compatible Unit**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Compatible Unit:** This zone provides information about the compatible unit.
- **Compatible Unit Hierarchy:** This zone displays a hierarchy tree view of the compatible unit.
- **Referencing Compatible Units:** This zone displays a table view of compatible units that reference the current compatible unit.
- **CUs for the Same CU Set:** This zone displays a table view of compatible units that are in the same compatible unit set.

- **Attachments:** This zone enables users to manage all types of attachments for the compatible unit record.
- **Compatible Unit Notes:** This zone displays note information about the compatible unit.

Defining Compatible Unit Sets

The **Compatible Unit Set** portal is used to display and maintain compatible unit sets.

Refer to [Understanding Construction Work](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Compatible Unit Set**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Compatible Unit Set:** This zone provides information about the compatible unit set.
- **Compatible Units:** This zone displays a table of the compatible units in the set.
- **Compatible Units Hierarchy:** This zone displays a tree view of the compatible units in the set.
- **Attachments:** This zone enables users to manage all types of attachments for the compatible unit set record.
- **Compatible Unit Set Notes:** This zone displays notes about the compatible unit set.

Defining Compatible Unit Usages

The **Compatible Unit Usage** portal is used to display and maintain compatible unit usages.

Refer to [Understanding Construction Work](#) for more information.

You can access the portal from **Admin**, select **C**, then select **CU Usage**.

The following zones may appear as part of the portal's **Main** tab page:

- **CU Usage List:** This zone lists all compatible unit usage records. Broadcast a record to display the details of the selected record.
- **CU Usage:** This zone provides information about the selected compatible unit usage.

Defining Compatible Unit Contractor Bids

The **CU Contractor Bids** portal is used to display and maintain contractor bids.

Refer to [Understanding Construction Work](#) for more information.

You can access the portal from **Admin**, select **C**, then select **CU Contractor Bid**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **CU Contractor Bid:** This zone provides information, such as bid amount and currency, about the contractor bid.
- **Other Bids by Contractor:** This zone displays a list of compatible units and associated contractor bids.
- **Attachments:** This zone enables users to manage all types of attachments for the compatible unit contractor bid record.
- **CU Contractor Bid Notes:** This zone displays notes about the contractor bid.

Defining Equipment Groups

The **Equipment Groups** portal is used to display and maintain equipment groups.

Refer to [Understanding Construction Work](#) for more information.

You can access the portal from **Admin**, select **E**, then select **Equipment Group**.

The following zones may appear as part of the portal's **Main** tab page:

- **Equipment Group List:** This zone lists all equipment group records. Broadcast a record to display the details of the selected record.
- **Equipment Group:** This zone provides information about the selected equipment group.

Defining Overheads

The **Overhead** portal is used to display and maintain overhead details.

Refer to [Understanding Construction Work](#) for more information.

You can access the portal from **Admin**, select **O**, then select **Overhead**.

The following zones may appear as part of the portal's **Main** tab page:

- **Overhead List:** This zone lists all overhead records. Broadcast a record to display the details of the selected record.
- **Overhead:** This zone provides information about the selected overhead, including material overhead, labor overhead, equipment overhead, and other overhead details.

Preventive Maintenance

Understanding Preventive Maintenance

Preventive maintenance programs can be utilized to keep your business in compliance with regulatory requirements and business needs. Users can define compliance rules to ensure the work is completed in a timely manner and to track and report against compliance work.

Please refer to [Understanding Compliance](#) for more information.

The system supports scheduled preventive maintenance and event based maintenance.

Event Based Maintenance

Event based maintenance is work that is performed based on trigger dates. Events differ from maintenance schedules because the system does not manage when to cycle the work. Instead, planners decide upon the next schedule date for the event and enter it on the **Maintenance Event** record. Work performed can include inspections, repairs, replacements, and so on, but it is typically associated to planned outages or other event based maintenance. Work records are generated on the trigger date based on the indicated defaults and templates.

Maintenance Forecasts

The upcoming maintenance for an asset is forecast based on the maintenance schedule and triggers defined for that asset. This information is displayed in the **Asset** portal on the **Asset Maintenance Trigger Forecast** zone.

About Maintenance Specification

Assets inherit their maintenance schedule from their **Maintenance Specification**. One **Maintenance 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”.

Configuration Guidelines

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:

Preventive Maintenance

- A **Maintenance Schedule** is associated with an **Asset Type** and provides a series of predefined **Maintenance Triggers** which define when to perform specific activities on the asset to ensure its continued operation.
- 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.

Refer to [Preventive Maintenance Triggers](#) for more on maintenance triggers.

- Assets inherit their maintenance schedule from their **Maintenance Specification**. One **Maintenance 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”.

Event Based Maintenance

Instead of using scheduled preventative maintenance, you might find that event based maintenance is more appropriate for the asset. Event based maintenance is typically used to execute planned outages.

- Maintenance plan triggers are used to configure a specific work order for a particular location for use by a maintenance event. They define the maintenance plan, template work order, activity type, service history type, work order defaults and asset location for the work being created for the maintenance event, as applicable. The work order template indicated on the trigger defines the pre-planned requirements for the work that is created for the event. The trigger location can either be defined on the work order template specified or on the trigger itself. Specifying an asset location on the trigger allows for the reuse of a common work order template.

2. Plan for the preventative or event based maintenance work by setting up a template work order.

3. Schedule the work plan:

- **Preventive Maintenance** Set up a maintenance schedule and maintenance triggers.

Note that maintenance triggers for compliance work also involves the definition of a **Compliance Category** and **Compliance Type** so that work orders can be generated.

- **Compliance Category** is the grouping of compliance types that are applicable to preventative or corrective maintenance.
- **Compliance Type** links the associated compliance category and compliance class to a specific maintenance trigger.
- **Event Based Maintenance** Set up a maintenance plan and maintenance triggers.

4. Trigger, execute and closeout the work.

Preventive Maintenance

- Monitor algorithms generate **Asset Maintenance Trigger Forecast** records with the date at which the asset is due for the preventative maintenance.
- The trigger creates the preventative 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.

Event Based Maintenance

- Planners decide upon the next schedule date then create or modify **Maintenance Event** records.

5. Evaluate results and plan effectiveness then modify the maintenance work plans as needed.

About Failure Mitigation

Failure profiles group failure types, failure modes, failure repairs, and failure components into a profile which can be attached to an asset type to define valid failure information for assets of that type. Effective use of failure profiles can help to ensure that assets run without fail, increase cost effectiveness, improve machine uptime, and help your organization to understand the level of risk you have undertaken with certain assets.

Generally the process includes the following:

- Define Failure Modes Effects and Critically Analysis:
 - 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.
- 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.
- 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.
- Set up preventive maintenance plans in the same way as described above using maintenance schedules, triggers and linking these to the assets using specifications. Base package **Maintenance Schedule Activity Types** and **Completion Event Types** can be used when setting up schedule plans and work requirements.

Preventive Maintenance Calendars

Preventive maintenance calendars establish calendars with explicit effective dates and schedule dates. These are used to schedule activities created by preventive maintenance triggers. The main function of the calendar is to allow direct control over scheduled work dates.

The effective date is determined using the activity work date with the “desired work date” as the activity work window start date.

Configuring Preventive Maintenance Options

Follow the steps described in this section to configure preventive maintenance options.

Defining Maintenance Managers

The **Maintenance Manager** portal is used to display and maintain maintenance managers.

Refer to [Understanding Preventive Maintenance](#) for more information.

You can access the portal from **Admin**, select **M**, then select **Maintenance Manager**.

The following zones may appear as part of the portal's **Main** tab page:

- **Maintenance Manager List:** This zone lists all maintenance manager records. Broadcast a record to display the details of the selected record.
- **Maintenance Manager:** This zone provides information about the selected maintenance manager.

Defining Preventive Maintenance Calendar

The **Preventive Maintenance Calendar** portal is used to display and maintain preventive maintenance calendars.

Refer to [Understanding Preventive Maintenance](#) for more information.

You can access the portal from **Menu**, select **Preventive Maintenance**, then select **Maintenance Calendar**.

The following zones may appear as part of the portal's **Main** tab page:

- **Maintenance Calendar List:** This zone lists all maintenance calendar records. Broadcast a record to display the details of the selected record.
- **Maintenance Calendar:** This zone provides information about the selected maintenance calendar.

Scheduling

Understanding Scheduling

A schedule manages a scope of work. The schedule takes into consideration various resource requirements:

- Crew type drives craft availability on the daily schedule.
- Assignment of the crew to a shift with exact time span when the crew is available.
- Matches crew capabilities with crew shifts resource requirements.
- Uncompleted work is reassigned to the next available shift.
- Automatically queries the system for unfinished backlog of work.

Employees

Employee records are used to manage human resource level information for the workers within your organization.

Details such as name and personal information, work status, the main craft worked by the employee, their business unit, payroll information, hire date, and timesheet information can all be maintained from the employee record. 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, assign assets (such as computer equipment or vehicles), and define the employee's billable rate.

Mandatory or optional training courses can be specified on the employee record to manage and track the employee's qualifications. Only active course records are presented on the value list when associating a training course to an employee record. However, once a course is associated to an employee record, it is displayed on the employee record regardless of the status of the course. After the course is selected, a mandatory **Effective Date** for the course must be specified (If not specified, the value will default to the current date. All other fields are optional). Courses are defined as an **Extendable Lookup BO (W1-Course)** and provide optional settings for the course, such as enabling the sending of email to the employee and to the employee's supervisor when a course is about to expire; Set **Employee Notification** value to *Yes* if you want to send an email to the Employee and set **Supervisor Notification** value to *Yes* if you want to send an email to Supervisor. Next, set the **Notification Lead Days** for the Employee, for the Supervisor, or for both of how many days before the expiration date the system should send the email.

Employee records are not required to reference a corresponding user profile. When there is no associated user profile, the module can be used to simply store information about employees who do not use the application. Information from the employee portal can be integrated with your external human resources system.

Employee is referenced throughout the system on other records such as crews and crew shifts, contacts, direct charges, timesheets, leave requests, and to indicate which employee is responsible for or in possession of an asset, and so on. These various records can be searched for using employee as the search criteria.

Timesheet Schedule

The individual lines of an employee's timesheet can be configured to automatically generate directly from the **Timesheet Schedule** tab on the **Employee** record for any *active* employee.

- **Auto Timesheet Creation** enables or disables the automatic creation of timesheet rows for the selected employee record.
- **Work Calendar** is an optional field on every employee record. If selected, **Work Calendar** identifies the normal business days for the organization and takes priority over the generation of the rows. For example, if **Work Calendar** is specified for an employee that works on a **Saturday** or a holiday, the timesheet row for those days will not be automatically generated since **Saturdays** and holidays are not normal work days defined in the **Work Calendar**.
- **Start Date** defines the date to generate the timesheet header rows.
- **End Date** defines the date the system is to stop creating the timesheet header rows.

NOTE: Generation of timesheet header rows are only generated up to and including the current date.

- Authorized users are able to make changes to the timesheet information for the selected employee. Such fields as craft, hours, charge types, days of the week, and start and end dates are some of the available information that can be edited for each row.

NOTE: **Start** and **End** dates for the individual rows supersedes the start and end date on the header. This process does not exclude approvals since planners or other authorized users must still validate timesheet information and submit an approval.

The *W1-EMP* batch is used to trigger the *WICREAUTOTS* and *W1-MONTRMDT* algorithms on the *W1-EmployeeInformation* business object's *Active* lifecycle to generate the automatic timesheet rows and monitor the termination date of the employee. The Last Processing Date field enables users to see the last date the timesheet row was generated.

Leave Requests

Employees can request leave or it can be requested on their behalf through the Employee Unavailability portal. Timesheets to cover these time periods of unavailability can be created in advance and then submitted for approval, which tells the system to skip the auto timesheet creation for those time periods. Refer to [Employee Unavailability](#) for more information.

Employee Calendar

The employee calendar is available on the **Employee** record when the employee is assigned to at least one shift or the employee has scheduled leave.

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.

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.

Note that an employee can be assigned to more than one crew.

Employees are defined as primary as well as secondary members of the crew and may access the crew's schedule of work using the mobile application.

Planners

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 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.

Service Area Criteria

A service area defines a company's territorial boundary. This may mean that your organization may define the service area geographically, by region or territory, or by business unit, such as construction, inventory, or purchasing, or by any other business need.

Service areas are used for scheduling and routing purposes:

- Matches the service area defined on the activity to the service area where the crew is allowed to work.

Service area can be defined at the crew level, which is the most common method, or on the employee record or crew shift.

Service Class Criteria

Service class on the crew shift assists the system in scheduling activities:

- Indicates the type of work a shift is able to perform.
- Prompts the system to prefer one service class over another.

A list of service classes can be defined on the crew shift to enable it to be matched with the service class listed on the activity.

Understanding Shift Plan Templates

While crew shifts can always be manually created to address a specific situation, they are typically planned and generated ahead of time based on recurring templates. Templates provide a tool for quickly generating large numbers of shifts based on predefined parameters.

A shift plan template defines a cyclical weekly pattern made up of weekday shift templates.

For example, you might define a weekly template in which the crew works its regular 9-5 shift Monday - Friday every week and also works an on-call shift on Saturday and Sunday every third week. Thus the shift plan template would rotate every 3 weeks.

Resource planners can create shift plan templates for specific crews, or they can create common templates and subscribe multiple crews to one template.

This section describes concepts related to setting up and maintaining shift plan templates.

A batch monitor associated with the subscription template business object performs the shift generation just like the crew specific shift weekly template. The base product implementation of generating shifts for a subscription uses the common template referenced on the subscription along with complementary details from the crew entity.

A Shift Plan Template is Made of Shift Templates

A weekly shift plan template describes general information about the shift plan—the number of weeks in rotation, crew, and effective date. The shift plan template also includes shift templates that describe the recurring daily shift patterns that were defined on the shift plan template for the number of weeks in the rotation. The details associated with each daily shift within the rotation is what is captured on the daily shift template record. For example, a weekly shift plan template might use a different daily shift template on Fridays than the one used on other days.

This section describes concepts related to setting up and maintaining shift templates.

Shift Templates versus Actual Shifts

Each shift template starts with the selection of a Crew Shift Type, and then other details, such as work availability hours, crew members, service classes, and service areas can be populated as needed. These details defined on the shift template is what is used to generate the actual shifts. A Shift Weekly Template Periodic Monitor Process (W1-SFWKT) is used to create the actual shifts based off the template.

The system enforces the close pairing between the shift template and the shift it creates by storing both the actual shift and the shift template records in the same table (this is done so that similar details can be shared). Also, the same crew shift type defines both the business object used to create the actual shifts of this type as well as the business object used to create the shift templates of that type. Additionally, generated shifts maintain a reference to the shift template that was used to generate it to ensure that any changes made to the template may be reapplied to those shifts.

Refer to [Defining Crew Shift Types](#) for more information on how to create crew shift types.

Complete versus Partial Shift Details

If the weekly shift plan template is dedicated to a [specific](#) crew, each daily shift template may include the complete shift information needed to generate a shift.

A weekly shift plan template may be defined as a common template to be [shared](#) by multiple crews. As such, details unique to each crew, such as the crew and the resources allocated to it, cannot be specified on the common template as they differ for each crew. The crew specific details are taken from the crew record at shift generation time.

Refer to [Shift Generation](#) for more information.

Crew Specific Plan Template

Crew specific plan templates are dedicated to a specific crew and cannot be shared with other crews. As such, each weekday shift template is fully defined with everything that the shift generator needs to generate a shift for this crew.

The initial date of the cyclical weekly pattern is the effective date of the template.

This type of plan template references the crew it is designed for, as well as the resources allocated to the crew, on each shift template. This allows for more complex situations where different resources are allocated to the same crew on different days of the week.

This section describes concepts related to setting up and maintaining crew specific plan templates.

Defining and Applying A Template

When you create a new shift plan template, its status is **Pending**. No shifts are generated while the template is in this state allowing you some time to properly define the weekday shift templates and review them as needed.

After you have defined the shift templates, you must apply the crew specific plan template so that actual shifts are generated from it based on the template's effective period. Applying the template changes its status to **Applied** and the system right away generates future shifts starting from the template's effective date (or from now if that date is in the past) and until the template's expiration date, if any.

NOTE: Generating shifts online is limited to a pre-configured number of days in order to avoid the apply action from taking too long and timing out. The parameter is defined on the shift generation algorithm.

Once applied, the shift generation background process manages the generation of shifts going forward.

Refer to [Shift Generation](#) for more information about the generation algorithm as well as the overall process.

Making Changes To An Applied Template

Extending or shortening the effective period of an applied template adjusts future shifts accordingly as soon as you make the change.

If you attempt to add, delete, or modify a shift template for an applied crew specific plan template, the system automatically creates a draft copy of the entire crew specific plan template it belongs to. The changes are made to the draft copy, leaving the original template unchanged. This allows you to make changes in a more controlled way.

You make changes to the draft copy and preview the shifts that would be generated should you accept the draft. If you decide to accept the draft, the original template is updated to match the draft and changes are immediately applied to already generated future shifts accordingly.

Refer to [Draft and Original Templates](#) for more information.

Common Plan Templates

You can set up common plan templates which can be shared across multiple crews. This is helpful for the efficient generation of shifts in organizations where there are sometimes hundreds or possibly even thousands of crews that follow the same weekly shift patterns.

This section describes concepts related to setting up and maintaining common plan templates.

Subscription

The common plan template defines the initial date of the cyclical weekly pattern as well as the details for each shift template. These shift templates cannot include crew specific information, such as the resources allocated to the crew on that shift and so on, as they need to be shared by multiple crews.

You may subscribe multiple crews to a common plan template to use as their template for generating shifts. Shifts generated from the common plan template are exactly the same except for crew specific details which are defaulted from the crew record. Any crew that subscribes to a common template must have these types of details defined on the crew record.

Subscription to a common template is effective dated so that crews can subscribe to follow a common template as of a given date and this relationship can expire. Each crew can subscribe at different times, but their shifts are generated in the same rotation timing as all other subscribed resources, regardless of when they subscribed.

You may also unsubscribe a crew from the common template at any time or change effective dates to affect the creation of future shifts for those crews.

Activation

In a typical workflow, you would create a common plan template and then subscribe multiple crews to the template. When template and subscriptions are ready to be used to generate shifts, you activate the template. Shifts are generated only when the common template is active.

Since a common template may involve a large number of subscriptions, actual shift generation is performed the next time the shift generation batch process runs. Note that you may still test your template and process it online for a subscribed crew without waiting for the next scheduled batch run.

Refer to [Shift Generation](#) for more information.

Making Changes To An Active Template

Changes made to an already active common plan template affect future shifts of all subscribed crews. Since a common plan template may involve a large number of subscriptions, such changes are not applied right away to all these shifts but rather they take effect the next time the shift generation batch process runs. This includes extending or shortening the effective period of an active template, subscribing or unsubscribing a crew, and so on.

If you want to test your recent changes without waiting for the next scheduled batch run you can do so by processing them online for one of the subscribed crews.

Similar to crew specific shift templates, if you attempt to add, delete, or modify a shift template for a shift plan template, the system automatically creates a draft copy of the entire shift plan template, leaving the original template unchanged.

This allows you to make changes in a more controlled way. You make changes to the draft copy and preview the shifts that would be generated should you accept the draft. If you decide to accept the draft, the original template is updated to match the draft. However, unlike crew specific templates, the impact on these changes on future shifts only takes effect the next time the shift generation batch process runs.

Refer to [Draft and Original Templates](#) for more information.

One Effective Plan Template for a Crew

A crew's specific shift plan template and subscription to common templates are either effective for a specified number of weeks starting from their effective date, or they may be perpetual, which means they have an effective date but no end date.

The crew can switch from a specific shift plan template designed just for this crew or subscribe to a common template, and vice versa over time, but their shifts may be generated by a single shift template at any given point in time. In other words, a crew's shift templates must never overlap in time.

Shift Generation

The system generates future shifts for crews based on the shift plan templates with which they are associated.

Common templates do not generate shifts as they are not defined for specific crews. Only a [crew specific](#) plan template and a crew's [subscription](#) to a common template are used to generate shifts for the crew throughout their effective period.

Unlike a crew specific shift plan template that is self contained, a crew's subscription to a common template sets the shift generation effective period, whereas template details as well as the initial rotation date are defined on the common template.

Shifts are generated based on the cyclical pattern of the shift plan template, starting from the shift plan template's effective date (or from now if the effective date is in the past) and up to a predefined number of days known as the planning horizon, or earlier, if the template expires before then.

The planning horizon is defined as a [global configuration](#) parameter and is used by the shift generation process to control the maximum number of days for shifts to generate and be available for planning purposes at any point in time.

If the template references a work calendar, the process skips generating shifts that even partially overlap with a public holiday.

The first few shifts may be initially generated online when you apply a crew specific shift template or verify the setup for a crew's subscription to a common template, but the ongoing process of generating additional shifts to meet the planning horizon is handled by the shift plan template monitor process.

NOTE: The shift generation process relies on the shift plan template being periodically monitored by the **Shift Weekly Template Periodic Monitor** batch process as often as appropriate to meet your organization's needs.

The background process is designed to add new shifts as needed on a daily basis. However, if any shift template was changed, the process would automatically regenerate future shifts that were earlier generated by that template to match its latest version. By default, even shifts that were manually overridden are regenerated to match the latest template version, but you may explicitly indicate to retain these changes when making a shift template change.

Refer to [Overridden Shift Details](#) for more information.

When a shift is about to be generated but happens to overlap with another shift for the crew that was not generated by this template, the generation of the shift is skipped and a user is notified by raising a to do entry that reports the situation.

NOTE: Logic used to generate shifts from a shift template resides in a "Generate Shifts" algorithm on the shift template's business object.

Overridden Shift Details

In a situation where crew shift details differ from their template on a particular date, generate the shift initially from the template and then manually adjust it to override the template settings. This includes canceling a shift that is no longer needed. These modified shifts are called overridden shifts as they no longer match the template from which they were generated.

By default, any changes made to a shift template is automatically reapplied to all future planned but not yet closed shifts that were generated from that template and any manual overrides will be lost. If you wish to retain overridden shifts you may indicate so on the shift template being changed.

Draft and Original Templates

If you attempt to add, delete, or modify a shift template for a shift plan template that has been applied (or activated for common templates), the system automatically creates a draft copy of the entire shift plan template. This allows you to make your changes to the draft and preview the shifts that would be generated if you accepted the draft.

You can also see if any manual edits were made to previously generated shifts. By default, such overrides are lost when the shift template that generated them is changed when a draft is accepted. You may explicitly request to retained overrides when making the shift template change.

If you have reviewed this information and are sure you want to make the proposed changes, you may accept the draft. Otherwise, you may delete the draft copy to discard the proposed changes.

When a draft template is accepted, the original template is updated to match the draft.

- If a shift template was changed, future shifts generated by this template are regenerated to match the changed template.
- If a shift template was deleted by the draft, the corresponding original template is also deleted if not yet used to generate shifts, otherwise it is marked inactive. Future shifts generated by this template are deleted.
- New shifts are generated for any shift templates that were added by the draft.

By default, changes made to a shift template impacts all future planned but not yet closed shifts generated by it, including overridden templates.

NOTE: If the original shift plan template is a crew specific template, changes are reapplied to future shifts right away when you accept the draft. If the original template is a common template, the impact is delayed to the next time the shift generation batch process runs.

Refer to [Overridden Shift Details](#) for more information.

Once a draft has been accepted, any further changes made to the original template will result in a new draft being created.

Configuring Scheduling Options

Follow the steps described in this section to configure scheduling options.

Defining Employee Types

The **Employee Type** portal is used to display and maintain employee types.

Refer to [Understanding Resource Requirements](#) for more information.

You can access the portal from **Admin**, select **E**, then select **Employee Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Employee Type List:** This zone lists all employee type records. Broadcast a record to display the details of the selected record.
- **Employee Type:** This zone provides information about the selected employee type.

Defining Employees

The **Employee** portal is used to display and maintain employees.

Refer to [Understanding Resource Requirements](#) for more information.

You can access the portal from **Admin**, select **E**, then select **Employee**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Employee:** This zone provides information about the employee, such as personal information, employment information, training information, and payroll information.
- **Employee Assets:** This zone identifies the relationship types between assets and employee.
- **Employee Shifts:** This zone provides a calendar view of employee shifts.
- **Billable Rate:** This zone provides hourly rate and effective date information for the selected employee.

The following zones may appear as part of the portal's **Timesheet Schedule** tab page:

- **Timesheet Schedule:** This zone enables or disables the automatic generation of timesheet rows for an individual employee. Also, timesheet and scheduling information can be edited directly from this zone.

Defining Employee Unavailability Types

The **Employee Unavailability Type** portal is used to display and maintain employee unavailability types.

Refer to [Understanding Scheduling](#) for more information.

You can access the portal from **Admin**, select **E**, then select **Employee Unavailability Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Employee Unavailability Type List:** This zone lists all employee unavailability type records. Broadcast a record to display the details of the selected record.
- **Employee Unavailability Type:** This zone provides information about the selected unavailability type.

Defining Crew Shift Types

The **Crew Shift Type** portal is used to display and maintain crew shift types.

Refer to [Understanding Scheduling](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Crew Shift Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Crew Shift Type List:** This zone lists all crew shift type records. Broadcast a record to display the details of the selected record.
- **Crew Shift Type:** This zone provides information about the selected crew shift type.

Defining Crew Types

The **Crew Type** portal is used to display and maintain crew types.

Refer to [Understanding Scheduling](#) for more information.

Refer to [Resource Usage](#) for a better understanding of crew charge rates.

You can access the portal from **Admin**, select **C**, then select **Crew Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Crew Type List:** This zone lists all crew type records. Broadcast a record to display the details of the selected record.
- **Crew Type:** This zone provides information about the selected crew type.
- **Crew Type Labor Profile:** This zone lists the craft resources for the crew type.
- **Crew Type Equipment Profile:** This zone lists the equipment resources for the crew type.

Defining Crews

The **Crew** portal is used to display and maintain crews.

Refer to [Understanding Scheduling](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Crew**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Crew:** This zone provides information about the crew, such as primary and secondary crew members, default crew service classes and crew service areas.
- **Shift Plan Template:** This zone provides details about the shift plan template.
- **Crew Calendar:** This zone displays a calendar view of the crew's shifts.

Defining Planners

The **Planner** portal is used to display and maintain planner records.

Refer to [Understanding Scheduling](#) for more information.

You can access the portal from **Admin**, select **P**, then select **Planner**.

The following zones may appear as part of the portal's **Main** tab page:

- **Planner List:** This zone lists all planner records. Broadcast a record to display the details of the selected record.
- **Planner:** This zone provides information about the selected planner.

Defining Service Areas

The **Service Area** portal is used to display and maintain service area records.

You can access the portal from **Admin**, select **S**, then select **Service Area**.

The following zones may appear as part of the portal's **Main** tab page:

- **Service Area List:** This zone lists all service area records. Broadcast a record to display the details of the selected record.
- **Service Area:** This zone provides information about the selected service area.

Defining Shift Plan Templates

The **Shift Plan Template** portal is used to display and maintain shift plans.

Refer to [Understanding Shift Plan Templates](#) for more information.

You can access the portal from **Admin**, select **S**, then select **Shift Plan Template**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Shift Plan Template:** This zone provides details about the shift plan template, such as shift plan controls and the template's effective period.
- **Shift Templates:** This zone displays a default weekly calendar view.

The following zones may appear as part of the portal's **Review** tab page:

- **Preview Calendar:** This zone provides a calendar view in order to review details of the shift plan.

Defining Shift Templates

The **Shift Template** portal is used to view and maintain a shift template as part of setting up a shift plan template.

Refer to [Understanding Shift Plan Templates](#) for more information.

You can access the portal from the shift templates zone on the shift plan template.

The following zones may appear as part of the portal's **Main** tab page:

- **Crew Shift Template:** This zone provides information about the shift, including the various maintenance actions applicable to its current state.
- **Overridden Shifts:** This zone shows shifts that were generated by this template but their setup has since been manually changed from the setup defined by the template.

Approvals

Understanding 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.

Basic approval processing requires that the person creating or managing a record that requires approval must create an approval request. This is done by clicking the **Submit for Approval** button on the record when it is in a pending approval status and then selecting the appropriate approval profile for the document. Once a record is submitted for approval, the request is emailed to approvers, sent to their To Do List, or both.

- The **Approval Request** shows the related document, the person who submitted it and when, who has approved it so far, and which roles still need to provide an approval decision.
- Approval requests use an **Approval Profile** to determine the properties and requirements for the approval. A common structure defines the approval process, users, and limits for each given functional area that uses approval processing. The **Approval Profile** indicated on the approval request identifies the people who need to approve the record and their role. Sometimes there are multiple approvers needed depending on how the approval profile is set up and your business practices. Often records will have a set of approval profiles which are defined for all records of that type. For example, when a new work request is routed for approval, the options for approval profile are limited to those defined for work requests. Most users only need to know which **Approval Profile** to choose when they submit a record for approval. This selection will depend on your business processes and how your approval profiles are configured. For example, when a user submits an approval request for a timesheet, they should only see approval profiles available for a timesheet. See [Approval Profiles](#) for more information.
- Approvers can access information about approvals that require their attention using the **Approval Dashboard**. Also, the person (requestor) who submitted the record for approval can check the status of the approval from a zone on their home page or by checking the individual record. The requestor is notified by e-mail when the record has been approved or if the request times out.
- Most approvers also have access to the **Approval Request Action** zone on the side **Dashboard** where they can record and submit their decision to reject or approve the approval request. Access to this zone is controlled by your **Portal Preferences** in your user profile.
- The **Log** tab on the record being approved indicates who approved the record and when.

Refer to [About Approvals](#) for more information.

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 can be shared across document types or they can be kept discrete to business process and functionality. This is also true for the **No Approval** user group. There can be one role to be reused across your approval processing configuration, or you can set more roles 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 approvals 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 To Do 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.

No Approval Needed

User groups referenced on **No Approval Required** approval profiles indicate the user groups that can create an approval request in which the system automatically approves the record. Some examples of where a **No Approval Required** approval profile can be used might be for emergency type work which could be generated at anytime or for routine maintenance type work that has a lower impact on business processes.

Example

The following table provides examples of the types of roles you may need to create in order 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 Automatic:</p> <ul style="list-style-type: none"> • Approval Type — Employee Unavailability • Approval Type — ODC • Approval Type — Project • Approval Type — Purchase Order • Approval Type — Purchase Requisition • Approval Type — Timesheet • Approval Type — Work Order • Approval Type — Work Request
To Do Role	<ul style="list-style-type: none"> • General Approver • General Director • General Manager • General Notification Only Approver • General Supervisor • General Work Management Supervisor • General Work Management Manager • General Purchasing Manager • General Purchasing Supervisor
User Groups	<ul style="list-style-type: none"> • Authorized for No Approval
Approval Profiles	<ul style="list-style-type: none"> • Approval Amount <ul style="list-style-type: none"> • General Purchasing Approver • High Value Purchasing Approver • General Work Request Approver • General Work Order Approver • High Value Work Order Approver • Project Approver • Other Direct Charges Approver • No Approval <ul style="list-style-type: none"> • Emergency Work Approval • No Amount <ul style="list-style-type: none"> • Employee Unavailability Approver • Timesheet Approver

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 they approve, but since the General Manager has been set with a mandatory approval type, the General Manager must approve the request for the processing to be considered complete.

Defining Approval Profiles

The **Approval Profile** portal is used to display and maintain approval profiles.

You can access the portal from the **Admin** menu, select **WAM General**, then select **Approval Profile**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Approval Profile:** This zone provides information about the approval profile.
- **Approval Information:** This zone provides additional details about the approval profile, such as the To Do to be used and the hierarchy to use when an approval record is submitted.
- **User Groups:** This zone provides information on what user groups can create the approval request.

Understanding Change Request Types

A change request type defines the related object, for example, Asset, that change requests of this type are for and the applicable approval profiles for such change requests.

Refer to [About Change Requests](#) for more information.

The related object is defined by the extendable lookup "Change Request Related Object". The product comes with support to several key entities but your implementation may extend this configuration to support additional entities.

The following sections describe concepts and features related to change request configuration.

Entity Specific Versus General Change Requests

An "Entity Specific" change request type supports the tracking and approval of changes requested for an existing entity. A general change request type supports all other types of requests, for example:

- A request to update several entities where the request describes in text what should be changed.
- A request to add one of more entities that do not exist yet.

Initiating Change Requests

A dashboard zone allows users to view or create a change request for the entity currently being displayed when at least one active change request type exists for such entity. For example, if a change request type exists for an asset then the dashboard zone would appear when an asset is displayed.

NOTE: Though the product comes with some “Change Request Related Object” extendable lookup values, you can still control which ones to enable by only creating the change request types for the ones you need.

Only one in-progress change request is allowed for a specific entity, for example a specific asset, therefore the button, when applicable, may only be available if such a request does not already exist. If the change request exists, a link to the in-progress request is provided, allowing any user to navigate to it and review the requested changes.

If there is no in-progress change request for that specific instance, a **Create Change Request** button is made available on the **Dashboard** for users with or without “Update” security access to that object.

By definition, this dashboard is not applicable to “General” change requests. Users need to use the menu to add such change requests.

Enabling Change Request For Another Entity

To enable change request for another object, you need to do the following:

- Add a new value to the “Change Request Related Object” extendable lookup.
 - Select the corresponding entity value. For example, Cost Center.
 - Select the corresponding FK reference, which is used to search for a specific instance of the related object. For example, a particular cost center.
- Create the change request types for the new related object.
- Associate the maintenance portal of that entity to the dashboard change request context sensitive zone as follows:
 - Find the maintenance portal by navigating to **Admin**, select **Portal**, and then search for the portal page.
 - Click the **Navigation Option** hyperlink to navigate to the portal’s navigation option.
 - Go to the **Target Transaction** and copy the navigation key.
 - Navigate to **Admin**, click **Context Sensitive Zone**, and then paste the navigation key into the **Navigation Key** field and click **Search**.
 - Add the **Change Request Dashboard** (W1-CGRQDASH) to the **Zone** column for this navigation key.
- If you want to have a nicely formatted UI on the change requests with separate fields for users to enter new values, you need to do the following:
 - Create a “mini” BO for that object.
 - Navigate to the entity’s maintenance object, select the **Options** tab, and select the change request “Valid BO Option Type” for the entity’s maintenance object options.
 - Define the “mini” BO as a BO option of **Option Type** “Change Request BO” on the base BO for that object. Please refer to those objects delivered as product values for the “Change Request Related Object”, except for “**General**” change requests. For example, if you go to BO “W1-Vendor”, you will see such BO options.
- Please Note:
 - The mini BO should simply include a list of elements you would like to include in the change request process but without **Record Actions** and so forth. Please refer to our base “mini” BO’s for an example.
 - Without a “mini” BO, users can only describe the changes they would like to make in the detail description text field. As such, users need to manually make the updates in the system upon approval of the change request and then update the change request to an “Applied” status. Please note that change requests of a “**General**” type are not for any specific object thus, are not a “mini” BO by definition.
- If you want to have audit logs for changes to elements defined in the mini BO, please add the audit algorithm ‘WIGENCGRQLOG’ to the BO’s that the mini BO is added to as a BO option.

- Flush the online cache.
 - Replace the cis.jsp section of the url with flushAll.jsp (case sensitive) and refresh the url.
 - Once finished, it is traditional to close all of your browser windows and log in.

Defining Change Request Types

The **Change Request Type** portal is used to display and maintain change request types.

You can access the portal from **Admin**, select **C**, then select **Change Request Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Change Request Type List**: This zone lists all change type records. Broadcast a record to display the details of the selected record.
- **Change Request Type**: This zone provides general information about the selected change request type as well as valid approval profiles.

Refer to [Understanding Change Request Types](#) for more information.

Financials

Understanding Financials

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 for a variety of situations, but is most generally associated with the purchase or use of resources, such as the time cost of using labor, the invoicing or receipt of materials, and so on. There is always some system transaction 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 using a distribution code and its associated GL Account Detail. The distribution code specifies how the GL Account string is to be constructed using the **GL Construction Algorithm**. The **Financial Master Configuration** specifies which distribution code to use for particular scenarios.

Please refer to the [Financial Transactions](#) section in the **Business User Guide** for a conceptual overview of financial functionality.

Master Configuration

Ensure that you create master configuration settings in **Financial Master Configuration**.

Cost Category

Cost categories group expenses or costs into logical buckets for reporting purposes.

- 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.

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 budget amounts, year to date actual amounts, and committed amounts for the cost center. These amounts represent the fiscal year rather than the calendar year. The cost center record also contains free form account segment fields for building the general ledger (GL) account number. General ledger account numbers are a concatenation of the elements, such as company, business unit, cost center, project, and so on. There are some elements that are not available or configured within the Oracle Utilities Work and Asset Management application but are still able to make up the GL string when financial transactions (FTs) are created. A **Cost Center Monthly Statistics** zone also allows you to display more detailed information by month in a graph.

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 to adjust the debit side).

Business Unit

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. Your implementation will use the business unit to indicate whether to cost timesheets by employee wage rate or craft rate.

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 or employee's business unit.

Rate Calculation

Tax Rate Schedule

A Tax Rate Schedule is the first step in implementing rate calculations and defines the tax components (State tax, City tax) 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..

Tax Components, Characteristic Type, and Characteristic Type Values

Each tax component should be set up as a Characteristic Type and include several associated values. For example, if the Characteristic Type is State, the associated values could include Alaska, Alabama, Arkansas, Arizona, and so on. You will also need to define the list of portals that this Characteristic Type is relevant for. (Only ones that have a Tax Information zone would really be valid) Taxes and tax rates are subject to change over time so each tax component would reference a factor that captures the effective-dated tax percentage.

Factors and Factor Values

Factors are used to calculate tax values related to anything which incurs costs that must be taxed for your organization, such as purchases and resources used to complete activities. For example, you might define different factors for city tax, state tax, and federal tax.

Factor definition references a Factor Characteristic Type and includes only those Characteristic Values that are appropriate. Factor definition also captures an Effective Date/Time and Factor Value which is used to calculate the tax and is able to be edited. The factor is then referenced on a **Tax Rate Schedule**.

Extending the Data Area

Each jurisdiction has its own applicable taxes so implementations will need to extend the system to add the taxes that are applicable for them. One way to extend the system is to use the Data Area (DA) Extension to extend the base Business Object (BO).

- You will need one DA for each business object you want to extend and the element name for every type of tax must match across all the DAs.
 - Find the Data Area that needs to be extended for the BO.
 - Drill into the Maintenance Object to discover the table the Characteristic would reside on.
 - Navigate back to the business object's **Schema** tab and copy the **Data Area** for the **Tax Information** zone.
 - Create a new custom Data Area and indicate what data it is extending (Prefix your element names with "cm" to avoid upgrade issues).
 - Return to the **Schema** tab and enter the schema to match your Fields and Factors.
 - Here is an example Schema:



Here is the schema:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <cmTaxState mdField="CM_TAXST" mapField="CHAR_VAL" uiHint:select="bs:W1-RETCHARVL"
  uiHint:selectIn="characteristicType:'WD-STATE';" uiHint:selectOut="valuePath:results/value;
  descPath:results/description">
    <row mapChild="W1_INVOICE_LINE_CHAR">
      <CHAR_TYPE_CD is="WD-STATE"/>
    </row>
  </cmTaxState>
  <cmTaxCity mdField="CM_TAXCT" mapField="CHAR_VAL" uiHint:select="bs:W1-RETCHARVL"
  uiHint:selectIn="characteristicType:'WD-CITY';" uiHint:selectOut="valuePath:results/value;
  descPath:results/description">
    <row mapChild="W1_INVOICE_LINE_CHAR">
      <CHAR_TYPE_CD is="WD-CITY"/>
    </row>
  </cmTaxCity>
</schema>
```

Please see the **Framework** section of this guide for more information on Data Areas.

ERP Integration

Integration of Oracle Utilities Work and Asset Management with an Enterprise Resource Planning (ERP) application, such as Enterprise Business Suite, can be used to manage financial functions, such as invoicing and procurement functions.

Users can define which application is the owner of the process from the ERP Integration portal . Once ownership is established, the information can either interface using the ERP connector or processed using batch processing.

Refer to the Oracle Utilities Work and Asset Management Connector for Enterprise Resource Planning documentation for more information.

Understanding Cost Adjustment Types

Cost adjustment types define the values for cost adjustments of a certain type including information such as accounting information and a default approval profile.

Cost Adjustment types are defined by:

- **Cost Adjustment Type Business Object:** The business object used for cost adjustments based on this cost adjustment type.
- **Approval Profile:** A list of approval profiles used in the approval process for this cost adjustment type are as follows:
 - Amount Based
 - Role Based
 - Self Approval

NOTE: that the cost adjustment type can have one, and only one, default adjustment profile which can be selected from a list of approval profiles.

- **Offset Cost Center:** Identifies the offset cost center to use for the clearing account.
- **Offset Expense Code:** Identifies the offset expense code to use for the clearing account.
- **Distribution Code:** A list of valid distribution codes used to construct the actual GL account. Each cost adjustment can have one and only one distribution code which controls how the GL account number is constructed by the system.

Refer to [Defining Cost Adjustment Types](#) for more information about setting up cost adjustment types.

Configuring Financial Options

Follow the steps described in this section to configure financial options.

Defining Accounting Calendars

The **Accounting Calendar** portal is used to display and maintain accounting calendars.

Refer to [Understanding Financials](#) for more information.

You can access the portal from **Admin**, select **A**, then select **Accounting Calendar**.

The following zones may appear as part of the portal's **Main** tab page:

- **Accounting Calendar List:** This zone lists all accounting calendar records. Broadcast a record to display the details of the selected record.

- **Accounting Calendar Display:** This zone lists accounting period information about the selected account calendar.

Defining Business Units

The **Business Unit** portal is used to display and maintain business units.

Refer to [Understanding Financials](#) for more information.

You can access the portal from **Admin**, select **B**, then select **Business Unit**.

The following zones may appear as part of the portal's **Main** tab page:

- **Business Unit List:** This zone lists all business unit records. Broadcast a record to display the details of the selected record.
- **Business Unit:** This zone displays information about the selected business unit.

Defining Cost Adjustment Types

The **Cost Adjustment Type** portal is used to display and maintain cost adjustments.

Refer to [Understanding Cost Adjustment Types](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Cost Adjustment Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Cost Adjustment Type List:** This zone lists all cost adjustment type records. Broadcast a record to display the details of the selected record.
- **Cost Adjustment Type:** This zone displays information about the selected cost adjustment.

Defining Cost Categories

The **Cost Category** portal is used to display and maintain cost categories.

Refer to [Understanding Financials](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Cost Category**.

The following zones may appear as part of the portal's **Main** tab page:

- **Cost Category List:** This zone lists all cost category records. Broadcast a record to display the details of the selected record.
- **Cost Category:** This zone displays information about the selected cost category.

Defining Cost Centers

The **Cost Center** portal is used to display and maintain cost centers.

Refer to [Understanding Financials](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Cost Center**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Cost Center:** This zone provides financial information for the cost center as well as account segment details used to create the general ledger account number.

- **Cost Center Monthly Statistics:** This zone displays a graphical view of monthly statistics for the selected cost center.
- **Valid Expense Code:** This zone displays valid expense codes for the selected cost center.

Defining Distribution Codes

The **Distribution Code** portal is used to display and maintain distribution codes.

Refer to [Understanding Financials](#) for more information.

You can access the portal from **Admin**, select **D**, then select **Distribution Code**.

The following zones may appear as part of the portal's **Main** tab page:

- **Distribution Code List:** This zone lists all distribution code records. Broadcast a record to display the details of the selected record.
- **Distribution Code:** This zone displays information about the selected distribution code.

Defining Expense Codes

The **Expense Code** portal is used to display and maintain expense codes.

Refer to [Understanding Financials](#) for more information.

You can access the portal from **Admin**, select **E**, then select **Expense Code**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Expense Code:** This zone provides information about the expense code, such as resource class, cost category, and account segment values.

Defining Factors

The **Factor** portal is used to display and maintain factors.

Refer to [Understanding Financials](#) for more information.

You can access the portal from **Admin**, select **F**, then select **Factor**.

The following zones may appear as part of the portal's **Main** tab page:

- **Factor List:** This zone lists all factor records. Broadcast a record to display the details of the selected record.
- **Factor:** This zone displays information about the selected factor.
- **Factor Values List:** This zone displays the factor values that are applied to the factor, the date and time the factor goes into effect, and the tax component used to calculate the tax values.

Defining Tax Rate Schedules

The **Tax Rate Schedule** portal is used to display and maintain tax rate schedules.

Refer to [Understanding Financials](#) for more information.

You can access the portal from **Admin**, select **T**, then select **Tax Rate Schedule**.

The following zones may appear as part of the portal's **Main** tab page:

- **Tax Rate Schedule List:** This zone lists all tax rate schedule records. Broadcast a record to display the details of the selected record.
- **Tax Rate Schedule:** This zone displays information about the selected tax rate schedule.

Integrations

Understanding 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:

- **Esri ArcGIS:**
GIS applications are widely used to manage assets, locations and other spatial data. The Oracle Utilities Work and Asset Management GIS Integration interfaces with an Esri ArcGIS Portal/Server to allow users to view asset layers on a map, and to create and view work requests and work orders on the map.
Refer to [Understanding GIS Integration](#) for more information and configuration details.
- **Enterprise Resource Planning (ERP):** Oracle Utilities Work and Asset Management can be integrated to an Enterprise Resource Planning (ERP) application, such as Enterprise Business Suite, to manage purchasing, financial, inventory, and human resource functional areas.
Refer to the Oracle Utilities Work and Asset Management Connector for Enterprise Resource Planning documentation for more information.
- **Oracle Utilities Analytics:**
Oracle Utilities Analytics Dashboards provide analysis of data from Oracle Utilities Work and Asset Management or Oracle Utilities Operational Device Management using Oracle Business Intelligence Enterprise Edition built-in metrics. Non-spatial analytics, information that is not tied to geography, is represented in a series of dashboards showing tables, bar graphs, pie charts, and gauges.
Refer to the Oracle Utilities Analytics documentation for more information.
- **Oracle Utilities Integration for Device Operations:**
(ODM only). Oracle Utilities Operational Device Management integration with Oracle Utilities Meter Data Management.
Refer to the Oracle Utilities Integration for Device Operations documentation for more information.

Understanding Web Services

This section provides a high-level description of web services and outbound messages provided by the product to support various types of integrations.

Refer to the [Inbound Web Service](#) portal for more information about each inbound service.

Refer to the [Outbound Message Type](#) portal for more information about each outbound message type.

Refer to the appropriate Oracle Utilities Integration documentation for information about specific integrations.

Work Order API

Work Order APIs enable an external system to receive work orders and to exchange messages related to these work orders.

Work Order API involves the following *Inbound Web Services*:

- **W1-IntTMSDtl** (Interface Timesheet Details):
Web Service involved in capturing the crew time charged to an activity. This Web Service manages the capture of an employee time sheet details report for an activity sent in from the mobile application or from an external system.
- **W1-IntODCDtl** (Interface Other Direct Charge Details):
Web Service involved in capturing equipment charges and other costs. This Web Service captures equipment and other direct charges for an activity sent in from the mobile application or from an external system.
- **W1-MblActCom** (Mobile Activity Completion):
Web Service involved in capturing activity completion details sent in from the mobile application or from an external system.
- **W1-WOActivityDetails** (Work Order Activity Details):
This Web Service enables an external system to request the most current information on an activity already interfaced to that system.

Work Order API involves the following *Outbound Web Services*:

- **W1-WOACTAMSG** (Work Order Activity Active Message Type):
Outbound web service used to send or resend a work order activity, including its details, to an external system.
- **W1-WOACTFMSG** (Work Order Activity Final Message Type):
Outbound web service used to notify the external system when an activity is finalized (canceled or completed) by WAM.

NOTE: If the activity is completed by the external system, the completion in WAM does not trigger a message.

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.

Understanding GIS Integration

GIS applications are widely used to manage assets, locations, and other spatial data. Oracle Utilities Work and Asset Management supports an integration with ESRI ArcGIS that includes data synchronization between the systems and a GIS map viewer that allows users to view asset layers on a map, create work related to these assets and more.

NOTE: This integration requires several installation steps to be performed in Esri ArcGIS. Refer to the *Installation Guide* for more information.

The following sections describe concepts and configuration options related to GIS integration.

GIS Owns Non-Tracked Assets

Non-tracked assets may be loaded into the system from the GIS system and synchronized on an ongoing basis. A GIS feature defined in the GIS system is synchronized into the system as an Asset Location and an Asset installed at that location. GIS feature details that describe the location are mapped to the Asset Location entity while others that are more asset related are mapped to the Asset entity.

Only features of specific classes configured as eligible for synchronization are considered. Each feature class that is eligible for synchronization needs to be defined in the **GIS to WAM Sync Mapping** extendable lookup. This definition controls the following:

- Exclusion of features that belong to the eligible feature class based on a specific GIS field name and value combination.

- Determining the asset location type and asset type to use for each synchronized feature based on specific GIS field name and value combinations.
- Mapping of location related feature details to corresponding elements on the asset location entity.
- Mapping of asset related feature details to corresponding elements and attributes on the asset entity.

Which System Owns Asset Related Details?

GIS is the system of record for location related details and as such these are designed to be synchronized into the system continuously, reflecting changes made to these details over time.

By default, asset related details are synchronized once from the GIS system when the asset is created and maintained internally by the product thereafter. Depending on business requirements, each asset related detail may be explicitly configured with either of the following synchronization options:

- Continuously synchronized from the GIS system. This is indicated on the **GIS to WAM Sync Mapping** extendable lookup record for the feature class.
- Continuously synchronized back to the GIS system when data changes in the system over time. This is indicated in the **WAM to GIS Sync Mapping** extendable lookup record for the asset type. Changes made to such asset details are detected using standard audit rules on the asset entity and a synchronization request is created to send an update message to the GIS system.

NOTE: The same asset detail cannot be continuously synchronized both ways.

A Feature's Global Identifier Is Key To Synchronization

The GIS feature's unique global identifier serves as a cross reference between the systems. When a feature is synchronized from the GIS system its global identifier is stored on the asset location entity as a unique GIS identifier. The ongoing synchronization process relies on this cross reference to uniquely identify assets to update when changes are made to their corresponding features in the GIS system.

Asset synchronization is implemented at the feature class level, not layers within a feature class, allowing assets to move between layers of the same class. Moving a feature to another class causes it to be deleted from the original feature class and added as a new feature of the new class with a newly assigned global identifier. In this situation, the next synchronization process would detect the original global identifier as being deleted and the a new feature as being added thus losing the ability to correlate them as the same feature. It is therefore not recommended to move assets between feature classes in the GIS system if retaining the cross reference is important for reporting or any other business purposes.

Similarly, when a user creates a new work location using the GIS Map Viewer, a corresponding new feature of "Work Location" class is created in the GIS system. The unique global identifier of the new feature is captured as a GIS identifier of the new work location entity for cross reference.

Asset Synchronization from GIS is Version Based

Ongoing data synchronization of GIS features into the system relies on Esri's Geodatabase versioning capability. The latter allows for various long-running edits and designs to use designated copies of the data concurrently with the current default version.

The ongoing data synchronization process leverages this capability as follows:

- The current default version represents features as built.
- Initially, all eligible features in the default version are synchronized into the system and a "WAM" version is created to represent the already synchronized version of the data.

- The next ongoing synchronization process compares between the default version and the “WAM” version to detect delta changes made in the default version since the last synchronization.
- The results of the comparison is a list of previously synchronized features that were since deleted or updated as well as newly added features that are eligible for synchronization.
- Based on this list, new assets are added to the system and changes are applied to corresponding existing assets using the predefined **GIS to WAM Sync Mapping** mapping rules.
- With each ongoing synchronization process a new “WAM” version is created to represent the most recently synchronized version of the data.

Asset Synchronization Mode

Initial and ongoing synchronization processes rely on different [master data views](#) when handling the mapping between the external and internal identifiers of an entity. The system keeps track of whether the synchronization is at the initial mode or not on the **Esri WAM Integration** master configuration record. At the end of initial synchronization process the record is updated to mark the initial step as complete thus indicating that the synchronization is at the ongoing mode.

Asset Removal

When a previously synchronized GIS feature is deleted in the GIS system, the next synchronization process detect this change as a **Removal** operation that results in the following changes:

- The corresponding asset location is marked inactive.
- The GIS global identifier reference on the asset location is removed since a feature with that identifier no longer exists in the GIS system.
- The corresponding asset is retired and moved to the default out of service location defined on the **Esri WAM Integration** master configuration record.

Asset Abandonment

When a previously synchronized GIS feature is marked as abandoned in the GIS system, the next synchronization process detect these changes as an **Abandonment** operation that results in the following changes:

- The corresponding asset location is marked inactive.
- The corresponding asset is retired.
- If the asset is not allowed to be left in place based on current location’s type the asset is moved to the default out of service location defined on the **Esri WAM Integration** master configuration record.

NOTE: For the synchronization process to detect any change made to a previously synchronized feature the latter must retain its unique global identifier as it is the only cross reference to the corresponding entity in the system. This includes abandonment. Therefore the abandoned feature in the GIS system must be marked as abandoned in such a way that it does not lose its unique global identifier. Moving an abandoned feature to another class would cause it to be assigned a new global identifier and therefore is not a supported method for abandonment. The feature can be moved between layers of the same feature class as needed as this technique retains its global identifier.

The synchronization process looks for a designated “abandonment” GIS field and a corresponding “abandonment” value on the GIS feature to mark the feature as abandoned. The “abandonment” GIS field and value combination are configured on the **Esri WAM Integration** master configuration record.

Reference The Original Asset

The system provides a designated identifier on the asset location entity to track a new asset's reference to the original asset it may have been associated with in the GIS system. This reference may be useful to track references in situations like asset splits etc. If your GIS system configuration supports such an attribute you may reference it using the **GIS to WAM Sync Mapping** mapping rules.

Populating such identifier on an asset location entity allows a user to navigate to the original asset location record online if such record exists in the system.

GIS Map Viewer

The GIS Map Viewer interfaces with Esri ArcGIS to present assets and work information on a geographic map. Refer to the [GIS Map Viewer](#) for information about GIS Map Viewer.

The following describes web map related configuration aspects of the GIS Map Viewer:

- A web map consist of a base map plus optional feature layers and controls.
- The default web map defined in the **Esri Maps** master configuration is displayed when GIS Map Viewer is initially launched.
- Additional web maps may be configured in Esri, allowing the user to toggle between them.
- Once defined in Esri, all web maps need to be registered in the **Esri Web Maps** extendable Lookup.
- When launched from specific portals that maintain an asset related entity, the default web map may be overridden for specific asset types. These overrides are defined in the **Esri WAM Integration** master configuration record.

The Drop Pin feature of the GIS Map Viewer allows a user to create a new work location for the purpose of creating work in that location. The work location business object to use is defined in the **Esri WAM Integration** master configuration record.

Configuring GIS Integration

Follow the configuration steps described in the following sections to enable the GIS Integration functionality. Refer to [Understanding GIS Integration](#) for more information.

NOTE: This integration requires several installation steps to be performed in Esri ArcGIS prior to the configuration steps described here. Refer to the *Installation Guide* for more information about the Esri ArcGIS integration components.

Asset Details Mapping

The product can be configured to support various mapping approaches between GIS features and assets. The following are high level guidelines to consider when designing such mapping:

- Analyze your GIS feature classes
 - Determine which feature classes should be synchronized.
 - Determine if subsets of a synchronized feature class should be skipped.
 - Determine which GIS fields should be synchronized in each direction.
 - Determine if value mapping is needed for some GIS values.

- Extend the base asset location business object if custom GIS location details need to be supported.
- Use asset attributes when possible to store GIS asset related details
- Define corresponding non-tracked Asset Types and Asset Location Types.
- For each synchronized feature class define the following:
 - Custom value mapping extendable lookups for specific GIS field values that may need to be mapped to different values.
 - A **GIS to WAM** extendable lookup.
 - A **WAM to GIS** extendable lookup if some asset details should be continuously synchronized back to the GIS system.

Map Viewer Setup

The GIS Map Viewer presents web maps defined in Esri ArcGIS. Register the unique identifier of each web map in the **Esri Web Maps** extendable Lookup.

Define the default projected coordinated system in the **GIS Projected Coordinate Systems** extendable lookup to ensure that the proper coordinate model is used in your map.

Configuring Outbound Messages

This section provides guidelines for configuring the components needed to support outbound messages to the GIS system.

Message Senders

Define a message sender for each of the outbound message types associated with the **GIS Integration** web service category. Each sender should be defined using the following guidelines.

Field / Context	Value
Invocation Type	Real-time
Message Class	SOAP: RHTTPSNDP (Send post messages via HTTP real-time)REST: RTJSONSNDR (Sender for real-time HTTP/JSON messages)
MSG Encoding	UTF-8 message encoding
HTTP Method	POST
HTTP URL 1	The URL of the corresponding service in Esri.
HTTP Header	REST:Content-Type:application/x-www-form-urlencodedAccept:application/jsonapplication/json
HTTP Form Data	adds=@XMLMSG@

External System

Configure an external system record for the GIS system your company sends messages to. Refer to [External System](#) for more information.

Provide an entry for each of the outbound message types associated with the **GIS Integration** web service category following these guidelines:

Field	Value
Outbound Message Type	The provided outbound message type

Field	Value
Processing Method	Real-time
Message Sender	The corresponding message sender for this type of message
Date/Time Format	XSD
Message XSL	SOAP: W1-CommonGISRequestForSOAP.xslREST: W1-CommonGISRequestForREST.xsl
Response XSL	SOAP: W1-CommonGISResponseForSOAP.xslREST: W1-CommonGISResponseForREST.xsl

Master Configuration Setup

Configure the **Esri WAM Integration** [master configuration](#) record.

Configure the **Esri Maps** [master configuration](#) record.

Asset Synchronization Batch Processes

Inbound synchronization can be performed for an initial synchronization or for delta updates and changes.

Synchronization request records are processed by the synch request monitor batch process, and it is at that time that the actual outbound message is prepared and sent to the external system.

No batch needs to run explicitly for outbound synchronization as the interface dynamically updates the GIS system.

The following sections describe the background processes involved in asset synchronization from the GIS system.

NOTE: Before initiating the inbound synchronization, make sure that configuration is done in **Master Data Synchronization** master configuration record. Refer to [Understanding Data Synchronization](#) for more information.

Initial Synchronization

The following batch processes need to be run in sequence to complete the initial load of all eligible assets from the GIS system.

NOTE: The system keeps track of the status of the initial conversion process in the **One-time Data Conversion** field of the **Esri WAM Integration** master configuration record. Until the process is complete, the field shows *Convert all Esri data*.

Step	Batch Code	Description	Additional Parameters (if any)
1	W1-GISID	Retrieve GIS ID's	
2	W1-PRGIS	Process GIS Data	
3	W1-CMSYN	Composite Sync Request	State: PENDING BO: W1-CompositeSyncReqGISAsset
4	F1-SYSRQ	Sync Request Monitor (Deferred)	State: PENDING BO: W1-InitialSyncRequestLocation
5	F1-SAKRQ	Sync Request Allocate Keys Monitor	State: DATA-TRANSFD BO: W1-InitialSyncRequestLocation

Step	Batch Code	Description	Additional Parameters (if any)
6	W1-SIKSP	Initial Sync Request - Resolve Keys Loc	State: KEY-ALLOCATD BO: W1-InitialSyncRequestLocation
7	W1-SILSP	Initial Sync Request - Load Data Location	State: KEYS-RESOLVD BO: W1-InitialSyncRequestLocation
8	W1-SILDD	Initial Sync Request - Loaded	State: LOADED BO: W1-InitialSyncRequestLocation
9	F1-SYSRQ	Sync Request Monitor (Deferred)	State: PENDING BO: W1-InitialSyncRequestAsset
10	F1-SAKRQ	Sync Request Allocate Keys Monitor	State: DATA-TRANSFD BO: W1-InitialSyncRequestAsset
11	W1-SIKAS	Initial Sync Request - Resolve Keys Asset	State: KEY-ALLOCATD BO: W1-InitialSyncRequestAsset
12	W1-SILAS	Initial Sync Request - Load Data Asset	State: KEYS-RESOLVD BO: W1-InitialSyncRequestAsset
13	W1-SILDD	Initial Sync Request - Loaded	State: LOADED BO: W1-InitialSyncRequestAsset
14	W1-DELVR	GIS Delete Version	

Ongoing Synchronization

After the initial conversion is complete, the following batch processes need to be run periodically and in the specified sequence to uptake ongoing delta changes to eligible assets from the GIS system.

Step	Batch Code	Description	Additional Parameters (if any)
1	W1-GISID	Retrieve GIS ID's	
2	W1-PRGIS	Process GIS Data	
3	W1-CMSYN	Composite Sync Request	State: PENDING BO: W1-CompositeSyncReqGISAsset
4	W1-SIOPE	Ongoing Sync Request - Pending	State: PENDING BO: W1-OngoingSyncRequestLocation
5	W1-SIOPE	Ongoing Sync Request - Pending	State: PENDING BO: W1-OngoingSyncRequestAsset
6	W1-DELVR	GIS Delete Version	

Understanding Data Synchronization

Types of Synchronization Requests

Data synchronization is performed using synchronization requests sent from the external system using 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**, click **Business Object**, and search for “W1%sync%” to locate business objects used for data synchronization.

Base Package Synchronization Request Business Objects

Navigate to **Admin**, click **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** and click **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, and so on).

“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, and so on) 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 (all key allocation jobs must finish)
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** and click **XAI Inbound Service** to review or modify master configuration settings.

Processing of the Synchronous Request

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 list 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

Configuring Inbound Communication Options

Follow the steps described in this section to configure inbound communication options.

Defining Communication Types

The **Communication Type** portal is used to display and maintain communication types.

Refer to [Understanding Integrations](#) for more information.

You can access the portal from **Admin**, select **C**, then select **Communication Type**.

The following zones may appear as part of the portal's **Main** tab page:

- **Communication Type List:** This zone lists all communication type records. Broadcast a record to display the details of the selected record.
- **Communication Type:** This zone provides information about the selected communication type, such as attachment suppression, communication data mapping, message validation and exception handling.

Maintaining Inbound Communications

This portal is used to view and maintain inbound communications.

Refer to [Understanding Integrations](#) for more information.

You can access the portal from **Admin**, select **I**, and then select **Inbound Communication**. You are brought to a query portal with options for searching. Once your record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page:

- **Inbound Communication:** This zone displays information, such as activity, asset, location, and event information, about the inbound communication.
- **Communication Tree:** This zone displays a tree view of communications and events for the selected inbound communication.

Enterprise Resource Planning Integration

Oracle Utilities Work and Asset Management can be integrated with an Enterprise Resource Planning (ERP) application, such as Enterprise Business Suite to manage human resources and procurement processes.

Refer to the Oracle Utilities Work and Asset Management Connector for Enterprise Resource Planning documentation for more information.

Creating Application Links in External Applications

When integrating Oracle Utilities applications with other external applications, you can create links from external applications to specific portals and pages within the Oracle Utilities applications (also known as "deep linking"). These links typically leverage Business Process Assistant (BPA) scripts that direct the user to the appropriate target portal or page for a specific entity such as an account or service point. Refer to [Creating Application Links in External Applications](#) in the *Application Framework Administrative User Guide* for more information about creating links.

Note: When creating application links, the use of the following parameters will help speed up launching the target page

- **initNav=false:** skips launching the default home page for the user
- **minDashboard=true** minimizes the dashboard for the user

GoTo Portal Script

The Framework provides a generic script (F1-GoToPrtl) that can be used to navigate to the maintenance portal for a specified maintenance object and entity. Refer to [Navigating to a Given Record's Maintenance Portal](#) in the *Application Framework Administrative User Guide* for more information about using this script.

WAM Examples

The table below provides examples of using the F1-GoToPrtl script to open specific portals used by Oracle Utilities Work and Asset Management.

Script	MO	ID	Example
F1-GoToPrtl	mo=W1-ACT	Activity ID	&script=F1-GoToPrtl&mo=W1-ACT&pkValue1={your Activity ID}
F1-GoToPrtl	mo=W1-ASSET	Asset ID	&script=F1-GoToPrtl&mo=W1-ASSET&pkValue1={your Asset ID}
F1-GoToPrtl	mo=W1-NODE	Location ID	&script=F1-GoToPrtl&mo=W1-NODE&pkValue1={your Location ID}
F1-GoToPrtl	mo=W1-WORKORDER	Work Order ID	&script=F1-GoToPrtl&mo=W1-WORKORDER&pkValue1={your Work Order ID}

Full URL Example:

```
https://<ENV_BASE_URL>/ouaf/cis.jsp/?minDashboard=true&initNav=false&script=F1-GoToPrtl&mo=W1-ACT&pkValue1={your Activity ID}
```

Configuration Options

Content Migration Assistant (CMA)

The Content Migration Assistant (CMA) provides customers with a flexible, extensible facility for migrating their configuration data from one environment to another, for example, from a development environment to a production environment.

Data that Cannot be Migrated Using CMA

Content Migration Assistant does not currently support migration of objects with system generated IDs. The following maintenance object is an example if an object that is not supported by CMA:

- Shift Templates that are not common templates, such as crew-specific templates or subscriptions to common templates.

NOTE: Above list is strictly an example and is not intended as a complete list of all objects not supported by CMA.

Global Configuration Setup

Many aspects of the application rely on setting up the **Global Configuration** master configuration record.

Refer to [Framework master configuration portal](#) for more information.

Analytics Configuration

This portal provides a bird's eye view of the configuration information for Oracle Utilities Analytics. It provides links and guidelines for the areas that need configuration to successfully run the extract transfer and load processes from Oracle Utilities Analytics.

Refer to the Oracle Utilities Analytics Administration Guide for more information.

You can access the portal from the **Admin**, select **A**, and then select **Asset Analytics Configuration**.

The following zones may appear as part of the portal's **Main** tab page

- **Analytics Extract Parameters:** This zone displays the extract data to set up.
- **Bucket Configuration List:** This zone displays bucket configuration to set up.

The following zones may appear as part of the portal's **ODM** tab page

- **BI-Oriented Extendable Lookup:** This zone displays extendable lookups to set up.
- **BI-Oriented Master Configuration:** This zone displays master configuration to set up.
- **Bucket Configuration Tree:** This zone displays tree view of analytics data.

Contextual Insights Configuration

Contextual Insights is a feature to allow for the configuration of widgets that highlight information that you want to bring to the attention of the user. The following information describes the configuration of **BADGE** insights for **Asset Condition Score** and **Location Criticality** information in the **Asset** portal.

To enable these two insights, you will need to take the following steps:

1. Configure Asset Condition Score Bucket if you haven't done so already.

- Navigate to **Admin**, select **B**, select **Bucket Configuration**, and then select **Add**.
- Select *Asset Condition Score Bucket Configuration (WI-AssetConditionScoreBuckets)* as the Bucket Configuration Business Object and enter the desired values.

Example:

Asset Condition Score Bucket Configuration

Main ⊕

BUCKET CONFIGURATION *

DESCRIPTION *

Bucket Value Ranges

		SEQUENCE	START RANGE	END RANGE	DESCRIPTION
+	🗑️	* <input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="1"/>	* <input type="text" value="Unrated"/>
+	🗑️	* <input type="text" value="2"/>	<input type="text" value="1"/>	<input type="text" value="2"/>	* <input type="text" value="Very Good"/>
+	🗑️	* <input type="text" value="3"/>	<input type="text" value="2"/>	<input type="text" value="3"/>	* <input type="text" value="Good"/>
+	🗑️	* <input type="text" value="4"/>	<input type="text" value="3"/>	<input type="text" value="4"/>	* <input type="text" value="Average"/>
+	🗑️	* <input type="text" value="5"/>	<input type="text" value="4"/>	<input type="text" value="5"/>	* <input type="text" value="Poor"/>
+	🗑️	* <input type="text" value="6"/>	<input type="text" value="5"/>	<input type="text" value="25"/>	* <input type="text" value="Very Poor"/>

2. Configure Asset Location Criticality Bucket if you haven't done so already.

- Navigate to **Admin**, select **B**, select **Bucket Configuration**, and then select **Add**.
- Select *Asset Criticality Bucket Configuration (W1-AssetCriticalityBuckets)* as the Bucket Configuration Business Object and enter the desired values.

Example:

Asset Criticality Bucket Configuration

Main ⊕

BUCKET CONFIGURATION *

DESCRIPTION *

Bucket Value Ranges

		SEQUENCE	START RANGE	END RANGE	CRITICALITY CATEGORY	DESCRIPTION
+	🗑️	* <input type="text" value="1"/>	<input type="text" value="0"/>	<input type="text" value="4"/>	* <input type="text" value="Low (W1LW)"/> ▼	* <input type="text" value="Low"/>
+	🗑️	* <input type="text" value="2"/>	<input type="text" value="4"/>	<input type="text" value="7"/>	* <input type="text" value="Medium (W1MD)"/> ▼	* <input type="text" value="Medium"/>
+	🗑️	* <input type="text" value="3"/>	<input type="text" value="7"/>	<input type="text" value="9"/>	* <input type="text" value="High (W1HI)"/> ▼	* <input type="text" value="High"/>
+	🗑️	* <input type="text" value="4"/>	<input type="text" value="9"/>	<input type="text" value="100"/>	* <input type="text" value="Critical (W1CR)"/> ▼	* <input type="text" value="Critical"/>

3. Configure the corresponding Insight Types and Insight Algorithms.

- Configure an insight type for asset condition and another for asset criticality by navigating to **Admin**, select **I**, select **Insight Type**, and then select **Add**.
 - Select a **Default Insight Color Option** that displays the text and background in a color.
 - Enter severity values and insight color options in the **Override Visual Settings** field if the insight algorithm is designed to return different severity values based on conditions.
- Configure an insight algorithm for asset condition and another for asset criticality using the W1ASCNLOCRT algorithm type.
 - Select the appropriate algorithm for each insight type, either asset condition or asset criticality.

- Algorithm **Parameter** and **Sequence** values must match the **Severity Values** defined on the appropriate insight type in order to have the badge information display on the **Asset** portal.
4. Set up one Insight Group for asset condition as follows:
- Navigate to **Admin**, select **I**, select **Insight Group**, and then select **Add**.
 - Select *Asset Condition(WIAC)* as the **Insight Class**.
 - Select *Badge Insight* as the **Valid Visual Structure** and click **Save**.
 - Click *Edit* in the **Related Insight Types** zone and add the asset condition insight type you just created to the **Insight Type** list on the insight group.
5. Set up one Insight Group for asset location criticality as follows:
- Navigate to **Admin**, select **I**, select **Insight Group**, and then select **Add**.
 - Select *Asset Location Criticality (WILC)* as the **Insight Class**.
 - Select *Badge Insight* as the **Valid Visual Structure** and click **Save**.
 - Click *Edit* in the **Related Insight Types** zone and add the asset location criticality insight type you just created to the **Insight Type** list on the insight group.

Home Page Configuration

The **Home Page** is the user's main page for reviewing and managing their work and can be configured as the first page launched when you log in to the application on the **User** portal's **Main** tab.

Users can personalize the arrangement of zones and how the zones are initially displayed on the **Home Page** from the **Portal Preferences** tab under the **User** profile page.

Additional **Home Page** configuration can be defined from **Master Configuration** portal.

Refer to [Understanding Home Page Configuration](#) in the **Administrative User Guide** for more information about **Home Page** configuration.

Refer to [About Home Page](#) in the **Business User Guide** for general information about the **Home Page**.

Understanding Home Page Configuration

Please refer to [About 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 [Understanding Portals and Zones](#), or the [User - Portal Preferences](#) section under [User Preferences](#) in the **Oracle Utilities Application Framework** documentation for more information on using portal preferences.

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 [Understanding Portals and Zones](#) in the **Oracle Utilities Application Framework** documentation.

Defining Statistics

The **Statistics** portal is used to display and maintain statistics.

Refer to [About Statistics](#) in the Framework documentation for more information on statistics.

You can access the portal from **Admin**, select **S**, then select **Statistics**.

The following zones may appear as part of the portal's **Main** tab page:

- **Statistics List:** This zone lists all property unit records. Broadcast a record to display the details of the selected record.
- **Statistics Display:** This zone displays information, such as statistical details and captured date and times for the selected statistic.
- **Statistics Log:** This zone displays log information about the statistic.

Understanding Online / Batch Loaders

Online / Batch Loaders enable the quick loading of data into the system. Data can be loaded for any business object as it uses the BO's schema to load data into the application.

Two types of loaders are supported:

- **Online Loader** easily uploads smaller amounts of data. Use this type of upload process when uploading business objects without complex lifecycles and BO rules. For example, stock items, stock item detail, crafts, crews, and more. The data is loaded through the UI to create the target objects.
- **Batch Loader** should be used for larger volumes of data and for complex business objects such as Organization, Asset, Commodity Codes, and so on. The Batch Loader uses a two-step process that first creates the inbound sync request and uploads the file to a staging table and secondly, uploads the sync request from the staging table to either create the target objects or create another dedicated inbound sync request that handles the complexity of a particular object.

Processing uses the creation of a request type to facilitate the upload process:

- For the **Online** request type, the actual data is stored within and processed by the request.
- For the **Batch** request type, two options for batch load are supported; from a file located on the server or from an attachment in the application. Either the file name (file on server) or the attachment id (attachments) is stored on the request, the data is loaded into the generic inbound sync requests, and the request acts as a header record for all the generic inbound sync requests.

A csv file is the format used for data uploads. The generic loader has very strict requirements as to the format of the file. Please see the “embedded” help in the **Online Loader Request Data** section for format and additional tips.

Creating CSV Files

While data can be manually entered into the system, the most efficient method for uploading the data is to use one of the CSV data templates already available for all appropriate business objects.

The first step is to create a .csv file (comma delimited) from the business object of the portal being loaded. The following is an example using expense codes:

1. Navigate to **Admin**, select **E**, select **Expense Code**, and then select **Search**.
2. Expand the **Record Information** zone and click the *Expense Code* hyperlink for the business object.

- Click the *View Schema* hyperlink on the **Main** tab and then copy the elements from the business object to the CSV file and create your data.

1	Sequence	Primary Key	Field 01	Field 02
2	1	Expense Code	description	Cost Category
3	2	expenseCode	description	costCategory
4	3	WD-Inventory	200 – Inventory	WD-Stock
5	4	WD-LaborPremium	120–Labor - Premium	WD-Labor

- Columns in the top row match the elements in the list of the request BO (header).
- The row with the lowest sequence number (1) is the description of xpaths to help users read the file and is ignored by the processing.
- The BO processing treats the row with the second lowest sequence number (2) not as data, but as the xpath to move data to the target BO, in this case, W1-ExpenseCode.
 - Note that the configuration of some BOs mean that this cell may start with a “+” sign. Also note that the file always treats this cell as a formula and adds an “=” sign to the front of the data string, as indicated by “#NAME?” (For example: =+costCategory/expenseCode). You must reformat the cell to “Text” and remove the “=” sign.
- Rows with sequence number three and up are treated as data.

Other Formatting Issues

The following information addresses known formatting issues in working with the CSV data template:

- Dates need to be formatted as yyyy-m-dd (2020-10-27).
- Date and Time should be formatted as yyyy-mm-dd-hh.mm.ss.
- Columns based on lookup values that have leading zeroes will need to match the values in the CSV file. For example, the TD_PRIORITY_FLG value should be “030”, not “30”.
- A link needs to be created when you want to include the “parent”, which has a system generated ID, in the same file with the “child”. For example, you can load Organizations but if you want to populate the Parent Organization’s ID in the same file, you would need to insert a link (for example, @3, which is the @ symbol plus the Sequence #) in the row where the system generated ID would reside.
- If multiple rows contain the same **Primary Key**, only the last row is loaded. All values for the **Primary Key** need to be on the same row.

Initiating Requests

Facilitate the upload process by creating the request type and the request.

1. Create the request type

- Navigate to **Admin**, select **R**, select **Request Type**, and then select **Add**.
- Enter a meaningful description for the request type. For example, **Online Loader Request Type** or **Batch Loader Request Type**.
- Select the request type business object that corresponds to your request type. For example, either “Online Loader Request Type” or “Batch Loader Request Type”.
 - Note that for **Batch Loader Request Type**, the user will need to define whether the contents from the attachment record will be deleted once the upload is finished.

2. Create the request

Online Requests

- Navigate to **Menu**, select **Tools**, select **Request**.
- Select your *Online* loader request type and then click **OK**.
- Select the business object you want to create.
- Select your CSV file to upload or enter the data directly into the **Data** zone and then click **Save**.
 - Note: If examples of online requests are already available in your system, search for the appropriate request, select **Edit**, and either use the existing data if it applies, or select **CSV File To Upload** to upload your already formatted and populated CSV file, or you can select **Download To CSV File** and manually populate the data into the file and then select **CSV File To Upload** to upload the file.
- Click on **Load Data** to submit the request.
 - Check the **Request Log** for requests that do not transition to a “Loaded” status. You may need to reset the request back to a created status and click **Edit** to fix the issue before resubmitting the request.

Batch Requests

- Navigate to **Menu**, select **Tools**, select **Request**.
- Select your *Batch* loader request type and then click **OK**.
- Select an external system.
- Select the business object you want to create.
- Enter a commit frequency.
- Enter the file name or select the attachment that should be in the request and click **Save**.
 - For **Batch File Upload** — *Filename* is specified in the request. Comma delimited file should be uploaded / available in the server. Run the batch job W1-GBUPL to create the sync request. Parameter File Name and File Path should match the file name and path of the file. Once job has ended and the request is loaded, run batch job F1-SYSRQ to process the created sync request.
 - For **Batch Attachment Upload** — *Attachment ID* is specified in the request. Comma delimited file is uploaded as an attachment. Run the batch job W1-GAUPL to create the sync request. Parameter Request ID should match the created request. Once the job has ended and the request is loaded, run batch job F1-SYSRQ to process the created sync request.
- The F1-SYSRQ parameters are as follows:
 - **Maintenance Object:** F1-SYNCREQIN
 - **Restrict by Batch Code:** True
 - **Restrict by Business Object:** W1-GenericDataLoaderSync
 - **Restrict by Status Code:** PENDING

Table Setup Sequence

OUAF Admin Setup

This section lists entities 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.

Level	Entity	Comment
1	Defining Countries	Your organization's country.

1	Defining Currency Codes	Your organization's 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.
1	Defining Display Profiles	Controls how dates, times, and numbers are displayed.
1	Defining Languages	The language to use for this implementation.
1	Defining Time Zones	Your organization's base time zone.
1	Defining To Do Roles	Used to associate users with To Do entries.
1	Defining Work Calendar	The work calendar for your organization which identifies your public holidays.
1	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 for assets.
1	Defining 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.
1	Defining 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.
1	Defining Outbound Message Types	Prerequisites: Business Object and Schema
1	Global Context Overview	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.
2	Defining Installation Options	Controls various aspects of the system. You must set up 'Address Information' and 'Global Context' algorithms.
2	Defining To Do Types	Used to define types of To Do Entries.
2	Defining Feature Configurations	Prerequisites: The Checkbox Feature Configuration needs the custom UI Maps. Add a feature configuration of feature type 'Schema Constants' and set the value for 'Default Country'.
2	Characteristic Types	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.
2	External Systems	If you are using an integration you must configure the external system in this portal.

General Data Setup

This section lists entities 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.

Level	Entity	Comment
1	Defining Communication Types	

1	Defining Statistics	
1	Defining Colors	
2	Maintaining Inbound Communications	
3	Defining Attachments	It is recommended that you use Document Types and Documents instead of attachments in Oracle Utilities Work and Asset Management.

Approval Setup

This section lists entities related to approval management functionality in the order they should be defined.

Level	Entity	Comment
1	Defining Approval Profiles	Prerequisite: To Do Type, To Do Role, and Access Group. 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.

Asset Management Setup

This section lists entities related to asset management functionality in the order they should be defined.

Level	Entity	Comment
1	Defining Condition Assessment Questions	Condition Assessment
1	Defining Service Area	
1	Defining Contact Types	
1	Defining Document Types	
1	Defining Employee Types	
1	Defining Organization Types	
1	Defining Service Classes	
1	Defining Storeroom Type	
2	Defining Configuration Types	Prerequisites: Configuration Type BOs and Lookups, Configuration Class, and Configuration Category
2	Defining Configurations	(ODM) Dependency
2	Defining Asset Location Type	
2	Defining Measurement Identifiers	Prerequisites: Resource Class (custom lookup), Unit of Measure (UOM) (characteristic type), Time of Use (characteristic type), and Special Identifier (lookup)
2	Maintaining Contacts	
2	Defining Documents	Prerequisites: Document Category (lookup)
2	Defining Service History Types	Prerequisites: Condition Assessment Question (if using Base BO's), Service History BO (if need custom), and Service History Category (Lookup)
2	Defining Business Units	Prerequisites: Cost Center

2	Defining Out of Service Location Types	
2	Defining Out of Service Locations	
2	Defining Unit of Measure Resources	Prerequisites: Resource Unit of Measure Class (Lookup)
3	Maintaining Service History	Prerequisites: Activity, Failure Profile, Deficiency (custom lookup), General Weighted (custom lookup), Percent of New (custom lookup), Asset Disposition (custom lookup), and Downtime Reason (custom lookup)
3	Defining Assets and Component Types	Prerequisites: Failure Profile
3	Maintaining Specifications	Prerequisites: Specification BO, Manufacturer, BOM ID, and Vehicle Detail (custom lookup) (fleet Specification)
3	Defining Condition Assessment Group	
3	Maintaining Asset Locations	Prerequisites: Site Location (custom lookup) and Parent Asset Location
3	Defining Organizations	Prerequisites: Parent Organization (optional)
3	Maintaining Warranty Terms	Prerequisites: Vendor
3	Maintaining Bill of Materials	Prerequisites: Stock Item
3	Defining Measurement Types	
4	Defining Configuration Reports	
4	Defining Condition Assessment Classes	
4	Maintaining Operational Readings	Prerequisites: Measurement Numeric Gauge Reason (lookup)
4	Maintaining Warranties	
5	Defining Condition Assessments	
5	Defining Employees	Prerequisites: User ID, Craft
6	Maintaining Assets, Components, and Operational Devices	

Construction Work Management Setup

This section lists entities related to contractor management functionality in the order they should be defined.

Level	Entity	Comment
1	Defining Property Units	
1	Defining Cost Centers	Capital Account Flag
1	Defining Compatible Unit Categories	Compatible Unit Class (custom lookup)
1	Maintaining Work Locations	
2	Defining Equipment Groups	
2	Defining Compatible Unit Usages	
2	Configuring Resource Type Options	Related Work Management Master Configuration must be configured with a Resource Type for Contractor Service .
2	Defining Overheads	Resource Types needed on overheads are meant for indirect overhead. They are used to create Other Direct Charges to account for the overhead adjustments.
2	Defining Service Classes	Construction or Construction Overhead Used to categorize activity types, identify renewal work, and defaults cost center on construction work activity if populated.

Cost Center on Service Class is typically the default Work In Progress (WIP) account.

3	Defining Crew Types	
3	Defining Activity Types	Related Financial Master Configuration must be configured with a Reconciliation Distribution Code.
4	Defining Compatible Unit Sets	Prerequisites: CU Class (custom lookup)
4	Maintaining Vendors	Prerequisites: Country, Currency, Location Class (Lookup), Free On Board, Payment Terms, and Capability Type
5	Defining Compatible Units	Prerequisites: Asset Type, CU Condition (lookup), CU Account Usage (lookup), CU ID Type (lookup), Stock Item, and Stock Item Detail. Identifies the materials, labor, and equipment necessary to install or remove assets in relationship to construction work.
6	Defining Contractor Bids	Defines the materials, labor, and equipment necessary to install or remove assets in relationship to construction work.
7	Maintaining Work Designs	Prerequisites: Work Design Type (lookup), Work Design Priority (lookup), Work Design Phase (lookup), Price Type (lookup), Responsibility Party (lookup), and Service Position (lookup)
8	Maintaining Construction Work Orders	Prerequisites: Planner, Criticality (custom lookup), Work Class (custom lookup), Work Category (custom lookup), and Work Priority (custom lookup)
8	Maintaining Construction Work Activities	Prerequisites: Storeroom, Asset, Project, Phase (custom lookup), Crew (as applicable), and Backlog Group (custom lookup)
9	Maintaining Activity Reconciliations	

Financial Setup

This section lists entities related to financial functionality in the order they should be defined.

Level	Entity	Comment
1	Defining Accounting Calendars	
1	Defining Cost Centers	
1	Defining Factors	
1	Defining Distribution Codes	Prerequisites: GL Account Construction Algorithm
2	Defining Cost Categories	Prerequisites: Color
3	Defining Expense Codes	
3	Defining Tax Rate Schedules	
3	Defining Cost Adjustment Types	

Inventory Setup

This section lists entities related to inventory functionality in the order they should be defined.

Level	Entity	Comment
1	Maintaining Manufacturers	

1	Defining Commodity Codes	
1	Defining Free On Boards	
1	Defining Payment Terms	
1	Defining Capability Types	
2	Maintaining Vendors	Prerequisites: Country, Currency, and Location Class (lookup)
3	Defining Purchase Commodities	Prerequisites: Buyer
3	Maintaining Stock Items	Prerequisites: Resource, Unit of Measure (UOM), Usage and Inventory Expense Codes, and Stock Item Category (custom lookup)
4	Defining Storerooms	Prerequisites: Storeroom Type
4	Maintaining Stock Item Details	Prerequisites: Cost Center, Buyer, and Bin Information
4	Maintaining Reorder Reviews	
4	Maintaining Material Request Templates	Prerequisites: Expense Code

Purchasing Setup

This section lists entities related to purchasing functionality in the order they should be defined.

Level	Entity	Comment
1	Defining Buyers	
2	Defining Blanket Contract Lists	
3	Maintaining Blanket Contracts	Prerequisites: Stock Items, Vendor, Payment Terms, and Cost Center
4	Maintaining Purchase Requisitions	Prerequisites: Vendor and Stock Item
5	Maintaining Purchase Orders	Prerequisites: Vendor
6	Maintaining Expedite Purchase Orders	
6	Maintaining Invoices	
7	Maintaining Invoice Groups	

Work Management Setup

This section lists entities related to work management functionality in the order they should be defined.

Level	Entity	Comment
1	Time Periods	
1	Defining Crew Shift Types	
1	Defining Labor Earning Types	
1	Defining Overtime Type	
1	Defining Failure Profiles	
1	Defining Preventive Maintenance Calendars	
1	Defining Project Categories	
2	Defining Maintenance Managers	
2	Maintaining Maintenance Plans	Prerequisites: Event Category (lookup)
2	Maintaining Preventive Maintenance Schedules	Prerequisites: Asset Type
2	Defining Planners	
3	Maintaining Template Work Orders	Prerequisites: Planner, Approval Profile, Work Class (custom lookup), Work Category

(custom lookup), and Work Priority (custom lookup)

3	Defining Activity Types	
3	Maintaining Preventive Maintenance Triggers	Prerequisites: Maintenance Trigger Business Object, Service History Type, Service Class, Approval Profile, Outage Type (custom lookup), Backlog Group (custom lookup), Asset Condition, and Asset Location.
3	Defining Completion Event Types	Prerequisites: Service History Type
4	Defining Miscellaneous Resources	Prerequisites: Unit of Measure and Resource (non-material class items)
4	Defining Employee Unavailability Types	Prerequisites: Cost Center and Expense Code
4	Defining Equipment	Prerequisites: Cost Center and Unit of Measure
5	Maintaining Crew Shifts	Prerequisites: Crew
5	Defining Shift Plan Templates and Template Activities	Prerequisites: Resources, Crew, and Schedule Plan Type
5	Maintaining Work Orders	Prerequisites: Cost Center, Project, Crew, Planner, Criticality (custom lookup), Work Class (custom lookup), Work Category (custom lookup), and Work Priority (custom lookup)
5	Maintaining Maintenance Plan Triggers	
5	Defining Crew Types	Prerequisites: Craft
6	Defining Crews	Prerequisites: Employee
7	Maintaining Work Activities	Prerequisites: Resources, Storeroom, Asset, Project, Service Class, Cost Center, Phase (custom lookup), and Backlog Group.

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 configured in the order listed. These are recommended settings, not requirements, based on your configuration of the system 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](#) in the **Oracle Utilities Application Framework** documentation 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>	Hourly.

Batch Control	Description	Suggested Freq.
	<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>	
<p>W1-APVRQ Approval Request Periodic Monitor Process</p>	<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 moves the approval request to the next approver.</p>	<p>Daily. Approver time out is defined as number of days.</p>
<p>W1-ACRR Activity Resource Requirement</p>	<p>Users can indicate a future 'Reservation' date on Material Requirements and the system will try to reserve the material when that date becomes current.</p>	<p>Run several times a day.</p>
<p>W1-ACT Activity MO Periodic Monitor Process</p>	<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 caused by stock re-order, material issue or return.</p> <p>This monitor also auto-activates approved activities based on the Activation Date/Time on an activity which defines a future activation date. The system will activate the activity once it is both approved and the 'Activation Date' becomes current.</p>	<p>Daily. Available inventory changes by the hour, however your implementation should avoid running this batch job too often, as it places a heavy burden on the system.</p>
<p>W1-WO Work Order MO Periodic Monitor Process</p>	<p>Causes the system to automatically close completed work orders after the indicated number of days after completion.</p> <p>This is relevant for auto close and service schedules.</p>	<p>Daily.</p>

Batch Control	Description	Suggested Freq.
W1-TMST Timesheet MO Periodic Monitor Process	Posts approved timesheets to create financial transactions and entries in the general ledger.	Daily or weekly (if time period on timesheets is weekly).
W1-ODCTR ODC MO Periodic Monitor Process	Posts approved direct charges to create financial transactions and to the general ledger.	Daily.
W1-COSTM Cost Center Periodic Monitor Process	Calculates monthly and year to date statistics per cost center. Note that users can also click a button online to calculate the statistics on demand.	Daily.
W1-ASTYP Asset Type Monitor Process	Calculates asset type average cost statistics.	Daily.
W1-CRSHF Crew Shift Periodic Monitor Process	Maintains crew shift status.	Daily.
W1-ASSET Asset Periodic Monitor Process	Monitors preventive maintenance triggers to create the related work orders. This monitor also creates PM Forecast records for the Asset. Your implementation can define the number of days to generate the future forecast, or planning horizon. For example, 1825 days would be 5 years of PM forecasts for all triggers. The planning horizon is controlled by a parameter on the W1-ASTMNTMON algorithm.	Daily.
W1-BCHDR Blanket Contract Header Monitor Process	Transitions blanket contracts that are awaiting activation to Active when future blanket contracts become current.	Daily.
W1-SID Stock Item Detail MO Periodic Monitor Process	Causes the stock item detail to monitor itself to check it's reorder levels and creates the reorder review as necessary. This batch control also monitors lot expiration dates. Note that W1-SID is the batch job for reorder review, not W1-RR. W1-RR is the standard monitor batch control delivered with the reorder review maintenance object, however, it does not perform the processing of creating reorder review records.	Daily.
W1-BSRR Batch Stock Reorder Monitor Process	Creates the purchasing documents based on reorder review records.	Daily. (after W1-SID)
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 Financial Master Configuration .	Daily.
W1-STAT Statistic Monitor Process	Calculates home page statistics.	Daily.

Batch Control	Description	Suggested Freq.
W1-SFWKT Shft Wkly Templt Periodic Monitor Process	Creates new shifts based on templates.	Daily.
W1-COMIN Inbound Communication Periodic Monitor Process (both ODM and WAM)	If an external system sends inbound communications to WAM, e.g. to complete activity, this batch job needs to be run to process those inbound communications. Base BO's have a deferred monitor algorithm on the initial state, which moves the inbound communication to the next state and is only executed when the batch is run. The reason why that monitor is a deferred is to avoid error(s) upon creation of the message.	As often as needed.
W1-MTRDP Material Disposition Monitor Process	Closes material dispositions after the assets have been processed (e.g. the new asset has been installed and the old asset have been either repaired and returned to a storeroom or retired).	Daily.
W1-SSWO Svc Sched WO MO Periodic Monitor Process	ODM only— this monitor is only relevant to activity generators in 'Template' state. Create activity generator, from a template, based on the schedule defined on service history.	Daily.
W1-MNTSC Maintenance Schedule Periodic Monitor Process	Calculate preventive maintenance statistics. This batch job must run in order to generate the statistics that are displayed on the Analytics tab in the PM Schedule portal.	Daily.
W1-MNTTR Maintenance Trigger Periodic Monitor Process	Calculate failure statistics.	Daily.
W1-WRNTY Warranty Periodic Monitor Process	This process monitors the warranty BO's active status to update the warranty to an "expired" status when the terms of the warranty is reached and to create a To Do when the warranty is about to expire.	Daily.
W1-CKLST Activity Checklist Monitor Process	Monitors the Activity Checklist by looking at all instances of the Checklist MO whose current state has monitor algorithms defined and, if so, invokes those monitor algorithms. Associated algorithm checks that all <i>Mandatory</i> Checklist Line Items are completed prior to allowing the Checklist to enter the <i>Complete</i> state.	Daily.
W1-CKLTY Activity Checklist Type Monitor Process	Monitors all instances of the MO whose current state has monitor algorithms defined and, if so, invokes those monitor algorithms.	Daily.
W1-RTM	Monitors the Resource Types (Craft, Stock Item, Equipment, and Other) in order to	Daily.

Batch Control	Description	Suggested Freq.
Resource Type Maintenance Object Monitor Batch Process	<p>maintain the Average Unit Price (AUP) field on the Stock Item, NOT the Stock Item Detail.</p> <p>The AUP on the Stock Item is the average unit price for that stock item across all storerooms.</p>	
W1-CUCTR CU Contractor Bid Periodic Monitor Process	<p>Moves CU Contractor Bids through various lifecycle states including activating the bid once it reaches its effective date and expiring it once the expiration date is reached.</p>	Daily.
W1-FIXAS Fixed Asset Extract	<p>For use with construction work management functionality: An external Fixed Asset Management system which manages the depreciation of fixed assets can interface with Oracle Utilities Work and Asset Management to retrieve fixed assets. This batch code creates a flat file to send these fixed assets to the external system. This batch code must also be referenced on the Construction Parameter Work Order Master Configuration to complete the following for fixed assets:</p> <ul style="list-style-type: none"> • send fixed assets to an external system • to add a retired fixed asset entry to the general process table • to add an accept or return line entry to the general ledger for fixed assets • to add a cancel line entry to the general ledger for a stock item that is a capital spare 	Daily.
W1-GAUPL Batch Attachment Upload	<p>This batch control uploads data from a .csv file that is stored in the application as an attachment. This batch control is designed specifically for the Batch Loader Request process and has some format requirements for the data in the file:</p> <ul style="list-style-type: none"> • Information cell in the Excel spreadsheet is treated as a formula. Reformat the cell to <i>Text</i> and remove the “=” sign that was automatically added by Excel. <p>Batch Loader Request can be accessed by navigating to Menu, click Tools, and then click Request.</p> <p>For each data row in the file, this batch creates a generic inbound sync request record which acts as a staging record. Submit batch F1–SYSRQ to create the records in the application.</p>	As needed.

Batch Control	Description	Suggested Freq.
W1-GBUPL Batch File Upload	<p>This batch control uploads data from a .csv file on the server. This batch control is designed specifically for the Batch Loader Request process and has some format requirements for the data in the file:</p> <ul style="list-style-type: none"> Information cell in the Excel spreadsheet is treated as a formula. Reformat the cell to <i>Text</i> and remove the “=” sign that was automatically added by Excel. <p>Batch Loader Request can be accessed by navigating to Menu, click Tools, and then click Request.</p> <p>For each data row in the file, this batch creates a generic inbound sync request record which acts as a staging record. Submit batch F1–SYSRQ to create the records in the application.</p>	As needed.

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.

W1-ACPTL	Accept Line MO Periodic Monitor Process
W1-ACTRR	ARR MO Periodic Monitor Process
W1-ASCLS	Assessment Class Periodic Monitor Process
W1-ASGRP	Assessment Group Periodic Monitor Process
W1-ASCDU	Asset Current Disposition Update
W1-BCLIN	Blanket Contract Line Monitor Process
W1-BUYER	Buyer MO Periodic Monitor Process
W1-CEVT	Completion Event MO Periodic Monitor Process
W1-CMSYN	Composite Sync Request
W1-SICER	Composite Sync Request - Error
W1-CFG	Configuration Periodic Monitor Process
W1-CFGRP	Config Report MO Periodic Monitor Process
W1-CVNR	Create Vendor from Characteristic
W1-CREW	Crew MO Periodic Monitor Process
W1-EMP	Employee MO Periodic Monitor Process
W1-EU	EU MO Periodic Monitor Process
W1-IA	IA MO Periodic Monitor Process
W1-INVHD	Invoice Header Monitor Process
W1-INVAP	Invoice AP Extract Batch
W1-MNTEV	Maintenance Event Monitor

W1-MNTMG	Maintenance Manager MO Periodic Monitor Process
W1-MSRMT	Measurement Periodic Monitor Process
W1-MTRQH	Material Request Header Monitor Process
W1-MTRQT	Material Request Template Monitor
W1-MTRTH	Material Return Header Monitor Process
W1-ODCDT	ODC Detail MO Periodic Monitor Process
W1-PIMSR	Physical Inventory Count Line Periodic Monitor Process
W1-SIOER	Ongoing Sync Request - Error
W1-SIOPE	Ongoing Sync Request - Pending
W1-PLANR	Planner MO Periodic Monitor Process
W1-POHDR	PO Header Monitor Process
W1-POLNE	Purchase Order Line Periodic Monitor Process
W1-PRHTR	PR Header MO Periodic Monitor Process
W1-PRJ	Project MO Periodic Monitor Process
W1-RCPTH	Receipt Header MO Periodic Monitor Process
W1-RCPTL	Receipt Line MO Periodic Monitor Process
W1-RTM	Resource Type MO Periodic Monitor Process
W1-RPINV	Reprocess Invoice
W1-RTNL	Return Line MO Periodic Monitor Process
W1-SVHST	Service Hist MO Periodic Monitor Process
W1-SYRQI	Inbound Sync Request MO Monitor Process
W1-SRSER	Sync Request Seeder - Error
W1-TMPW	Template WO MO Periodic Monitor Process
W1-TMPWO	<p>Template WO MO Periodic Monitor Process</p> <p>Retrieves all the template activity resources for the template work order and updates the unit price on the template activity resource.</p> <p>Run process as needed.</p>
W1-TSDTL	Timesheet Detail MO Periodic Monitor
W1-VNDRL	Vendor Location Periodic Monitor Process
W1-WR	Work Request MO Periodic Monitor Process