Oracle Utilities Application Framework

Administration Guide

Release 4.3.0.0

E52574-01

March 2015

ORACLE

Oracle Utilities Application Framework Administration Guide

Release 4.3.0.0

E52574-01

March 2015

Documentation build: 4.4.2015 6:30:28 [F1_1428154228000]

Contents

Chapter 1

Overview

The Administration Guide describes how to implement and configure the Oracle Utilities Application Framework. This includes:

Defining General Options

Access Groups, Data Access Roles and Users

User Interface Tools

Database Tools

Configuration Tools

To Do Lists

Defining Background Processes

Defining Algorithms

Defining Script Options

Data Synchronization

Attachments

Application Viewer

Defining and Designing Reports

External Messages

Importing Users and Groups

Configuration Migration Assistant (CMA)

Configuring Facts

This guide contains the same content as the Framework Administration section of the online help.

Chapter 2

Defining General Options

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

Defining Installation Options

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

Installation Options - Main

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

Description of Page

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

System Owner will be **Customer Modification**.

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

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

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

CAUTION: In order to improve response times, installation options are cached the first time they are used after a web server is started. If you change the Admin Menu Order and you don't want to wait for the cache to rebuild, you must

clear the cached information so it will be immediately rebuilt using current information. Refer to *Caching Overview* for information on how to clear the system login cache (this is the cache in which installation options are stored).

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

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

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

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

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

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

Installation Options - Messages

Select **Admin** > **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 how the various **Message Types** are used in the system.

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).

Installation Options - Algorithms

Select **Admin** > **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

System Event	Optional / Required	Description
		sending attachments using the Email service. Refer to *Sending Email* for more information. Click *here* to see the algorithm types available for this system event.
Geocoding Service	Optional	Algorithms of this type use Oracle Locator to retrieve latitude and longitude coordinates using address information. Click *here* to see the algorithm types available for this system event.
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. Click *here* to see the algorithm types available for this system event.
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. Click *here* to see the algorithm types available for this system event.
Ldap Import	Optional	Algorithms of this type are called for operations on users, groups, and group memberships after they have been processed. Click *here* to see the algorithm types available for this system event.
Ldap Import Preprocess	Optional	Algorithms of this type are called to preprocess data retrieved from LDAP. Click *here* to see the algorithm types available for this system event.
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. Click *here* to see the algorithm types available for this system event.
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. Click *here* to see the algorithm types available for this system event.
SMS Receive Service	Optional	This type of algorithm is used to provide SMS receive service. Only one algorithm of this type should be plugged in. Click *here* to see the algorithm types available for this system event.
SMS Send Service	Optional	This type of algorithm is used to provide SMS send service. If your installation uses the base algorithm that uses BPEL, you will need to create a feature configuration with the SMS

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

Installation Options - Accessible Modules

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

Description of Page

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

Release ID shows the current release of the application that is installed. This field is used by the system to ensure that the software that executes on your application server is consistent with the release level of the database. If your implementation of the product has developed implementation-specific transactions, you can populate the Release Id for the **Customer Release** entry to define the latest release of implementation-specific logic that has been applied to this environment. In order for this to work, your implementation team should populate this field as part of their upgrade scripts.

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

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

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

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

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

Support For Different Languages

User Language

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

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

The following steps are required to support a new language:

1. **Define a language code and indicate that it is enabled.** For details on this procedure, see *Defining Languages*.
2. **Copy descriptions of all language-enabled tables from an existing translation (e.g., English).** The copied values act merely as placeholders while the strings are translated into the new language. It is necessary to do this as a first step in order to create records using the new language code created in the previous step. Language-based descriptions can be copied using a supplied batch process, *F1–LANG*. The batch copies all English labels in the system.
3. **Apply the language pack.** If the product supplies a language pack with translations for the system metadata descriptions, follow the instructions provided with the language pack to add the translated text.
4. **Translate additional content.** Translatable descriptions and labels for implementation data may be updated / entered in the application. First the user record must be updated to reference the new language. This may be done in one of the following ways:

   a. Switch to the new language using the *Switch Language* zone.

   b. If that zone is not available, navigate to the user page, assign the new language code to your User ID, sign out, and sign back in again.

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

Additional Topics

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

Defining Languages

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

Description of Page

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

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

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

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

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

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

The following are examples of valid locales:

- en_US (this is for American English)
- en_AU (this is for Australian English)
- pt_BR (this is for Brazilian Portuguese)
- fr_FR_EURO (this is for European French)
- ja_JP (this if for Japanese)

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_LANGUAGE*.

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

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

Defining Countries

The topics in this section describe how to maintain countries.

Country - Main

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

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

Description of Page

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_COUNTRY*.

Country - States

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

Description of Page

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

- Enter the standard postal abbreviation for the **State** or province.
- Enter a **Description** for this state or province.

Defining Currency Codes

The currency page allows you to define display options related to currency codes that are used by your system. Use **Admin** > **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. For example, there are two decimal positions for Australian dollars ($5.00), but no decimal positions in the Italian lira (500 L).

The **Currency Position** indicates whether the currency symbol should be displayed as a **Prefix** or a **Suffix** to the currency amount. For example, the US Dollar symbol is a prefix before the amount ($5.00) and the French franc symbol is a suffix after the amount (200.00FF).

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_CURRENCY_CD*.

Defining Time Zones

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

Designing Time Zones

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

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

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

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

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

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

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

Setting Up Time Zones

Refer to *Designing Time Zones* for background information about defining time zones.

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

Description of Page

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

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_TIME_ZONE*.

Setting Up Seasonal Time Shift

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

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

Description of Page

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_SEAS_TM_SHIFT*.

Defining Geographic Types

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

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

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

Description of Page

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

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

Click *here* to see the algorithm types available for this plug-in spot.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_GEO_TYPE*.

Defining Work Calendar

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

Description of Page

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_CAL_WORK*.

Defining Display Profiles

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

Description of Page

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

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

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

Here are some examples of date formats.

Format	Displayed / entered as
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 and the letter ‘H’ is used for dates that fall within the *Heisei* era. For example, for a user whose display profile date format is **Gyy/mm/dd** the Gregorian date 2008/01/01 is shown as **H20/01/01** ; the Gregorian date 1986/03/15 is shown as **S61/03/15**. The setting **GGGGyy** is interpreted as a year entered and displayed using Japanese characters for the era followed by the era number.

  Japanese date limitations are as follows:

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

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

- **hh** or **h** is interpreted as the hour, 1-12; **KK** or **K** is interpreted as the hour, 0-11; **HH** or **H** is interpreted as the hour, 0-23; **kk** or **k** is interpreted as the hour, 1-24. The **h**, **K**, **H**, and **k** options suppress a leading 0.

- **mm** or **m** is interpreted as the minutes. The **m** option suppresses a leading 0.
- **ss** or **s** is interpreted as the seconds. The **s** option suppresses a leading 0.
- **a** is interpreted to mean display **am** or **pm** (only needed when the hour is entered in **hh**, **h**, **KK** or **K** formats). If an **am** or **pm** is not entered, it defaults to **am**.

Here are some examples of time formats.

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_DISP_PROF*.

Additional Hijri Date Configuration

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

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

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

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

Defining Phone Types

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

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

Description of Page

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_PHONE_TYPE*.

Setting Up Characteristic Types & Values

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

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

There Are Four Types Of Characteristics

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

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

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

contact entry. When such a characteristic is defined on an entity, a button can be used to open the URL in a separate browser window.

Searching By Characteristic Values

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

- Some base searches provide an option to search for an object by entering Characteristic Type and Characteristic Value.
- Your implementation may define a customized search for an entity by a characteristic value for a specific characteristic type using a query data explorer.
- Your implementation may require a business service to find a record via a given characteristic value. For example, maybe an upload of user information attempts to find the user via an Employee ID, defined as a characteristic.

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

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

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

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

Characteristic Type - Main

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

Description of Page

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

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

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

- **Predefined value.** Characteristics of this type have their valid values defined in the **Characteristic Value** scroll, below. For each valid value, enter an easily recognizable **Characteristic Value** and **Description**. For example, if you introduce a characteristic type to record the number of volts that are measured by a meter, you would add values for all voltage levels measured by the meters utilized by your organization.
- **Ad hoc value.** Characteristics of this type have an unlimited number of valid values. For example, if you use a characteristic to define the date a meter was ordered, you'd use this classification because you can't define every possible order date. If you use this option, you can optionally define the **Validation Rule** used to validate the user-entered characteristic value. For example, if we carry on with our example, you'd use a validation rule to make sure that what the user entered was actually a date. Click *here* to see the algorithm types available for this plug-in spot.

- **File location value.** Characteristics of this type contain a URL. The URL can point to a file or any web site. Characteristics of this type might be useful to hold documentation/images associated with a given entity. For example, the image of the formal contract signed by a customer could be referenced as a file location characteristic on a service agreement. When such a characteristic value is defined on an entity, the value is suffixed with a button that can be used to display 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 for Internet Explorer version 7 or later, or in Firefox. If the file protocol is used, the browser either does not return properly or an error is thrown (e.g., "Access Denied", which usually results from cross site scripting features added for security reasons).

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

- **Foreign key reference.** Characteristics of this type have their valid values defined in another table. A good example of such a characteristic type would be one used on a premise to define the premise's building manager. Valid values for this type of characteristic would be defined on the person table (as you'd set up a person for the building manager and then reference this person on the premise characteristic). If you choose this option, you must use **FK Reference** to define the table that controls the valid values of this characteristic type. Refer to *Setting Up Foreign Key References* for more information.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_CHAR_TYPE* in the data dictionary schema viewer.

Characteristic Type - Characteristic Entities

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

Description of Page

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

NOTE: The values for this field are customizable using the Lookup table. This field name is **CHAR_ENTITY_FLG**.

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

Setting Up Foreign Key Reference Information

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

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

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

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

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

Information Description Is Dynamically Derived

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

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

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

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

Navigation Information Is Dynamically Derived

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

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

Search Options

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

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

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

Foreign Key Reference - Main

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

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

Description of Page

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

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

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

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

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

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

Refer to *Information Description Is Dynamically Derived* for more information on the info program is used.

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

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

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

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

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_FK_REF*.

Defining Feature Configurations

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

For information about specific features:

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

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

- Review the lookup values for the lookup field **EXT_SYS_TYP_FLG**. If your new option can be logically categorized within an existing feature type, note the lookup value. If your new option warrants a new feature type, add a lookup value to this lookup field.
- Define the feature's option types. If you have identified an existing feature type to add the options to, find the lookup with the name **xxxx_OPT_TYP_FLG** where **xxxx** is the lookup value of **EXT_SYS_TYP_FLG** noted above. If you

decided to create a new feature type (by adding a new lookup value to the **EXT_SYS_TYP_FLG** lookup, you must create a new lookup with the name **xxxx_OPT_TYP_FLG** where **xxxx** is the new value you defined above.

- Flush all caches.

Feature Configuration - Main

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

Description of Page

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

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

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

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

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

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

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

Feature Configuration - Messages

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

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

Description of Page

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

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

Defining Master Configurations

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

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

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

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

Setting Up Master Configurations

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

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

Master Configuration

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

The following functions are available:

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

Master Configuration Details

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

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

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

Miscellaneous Topics

The following sections describe miscellaneous system wide topics.

Module Configuration

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

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

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

- Menu items may not appear. Refer to *Menu Item Suppression* to better understand how menu item suppression works.
- Entire menus may not appear. Refer to *Menu Suppression* to better understand how menu suppression works.
- Tabs on pages may not appear.
- Fields may not appear.
- The system may return an error message when you attempt to use a function (indicating the function is turned off).

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

Your module configuration setup is displayed on the *installations* record.

Menu Item Suppression

The following points describe how your module configuration can suppress *menu items*.

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

Menu Suppression

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

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

Turn Off A Function Module

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

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

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

You may view your module configuration setup on the *installation options* page.

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

Global Context Overview

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

The fields designated as global context for the product are defined using the lookup **F1_UI_CTXT_FLDS_FLG**.

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

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

System Data Naming Convention

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

Base Product System Data

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

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

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

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

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

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

Implementation System Data

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

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

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

Caching Overview

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

Server Cache

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

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

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

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

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

- **flushAll.jsp?language=<language code>**. This command flushes every cache described below.
- **flushDropdownCache.jsp?language=<language code>**. This command flushes all objects in the dropdown contents associated with a given language (note, the **language code** is defined on the *Language* control table). Use this whenever you change, add or delete values to/from control tables that appear in dropdowns. Unlike the above caches, the objects in the dropdown cache are flushed automatically every 30 minutes from the time they are first built.
- **flushDropDownField.jsp?language=<language code>&key=<field>** If you want to flush the values in a specific drop-down field in the cache, specify the *<field>* using this command.
- **flushMenu.jsp**. This command flushes the menu cache. Use this command if you change menu data.
- **flushMessageCatalog.jsp**. This command flushes the field labels. Use this whenever you add or change any labels (this is typically only done by implementers who have rights to introduce new pages).
- **flushMessaging.jsp**. This command flushes messages.
- **flushNavigationInfo.jsp**. This command flushes the navigation information in the cache.
- **flushPortalMetaInfo.jsp**. This command flushes the portal information cache. Use this command whenever you change any portal zone meta-data.
- **flushSystemLoginInfo.jsp**. This command flushes the system information cache. Use this whenever you change navigation options, cached information from the installation record, or if you change an application service's URL.
- **flushUI_XSLs.jsp**. This command flushes the XSL document cache. Use this command if you change user interface meta-data. Also use this command whenever you change or introduce new zones.

For example, assume the following:

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

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

Client Cache

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

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

NOTE: Automatic refresh of the browser's cache. Each user's cache is automatically refreshed based on the **maxAge** parameter defined in the web.xml document on your web server. We recommend that you set this parameter to **1** second

on development / test environments and **28800** seconds (8 hours) on production environments. Please speak to system support if you need to change this value.

Batch Cache

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

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

Debug Mode

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

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

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

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

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

NOTE: Show User Log button is secured. An application service **F1USERLOG** has been provided for this functionality to allow implementations to restrict user access to this button. Such restriction may be called for in production environments.

System Override Date

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

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

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

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

Advanced Search Options

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

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

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

Chapter 3

Defining Security & User Options

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

The Big Picture of Application Security

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

Application Security

The system restricts access to its transactions as follows:

- An *application service* may be associated with every securable function in the system.
  - All maintenance objects define an application service that includes the basic actions available, typically **Add**, **Change**, **Delete**, and **Inquire**. The base product supplies an application service for every maintenance object.
  - For maintenance objects whose user interface page is not portal-based, the application service also controls whether the menu entry appears. If a user doesn't have access to the maintenance object's application service, the menu item that corresponds with the application service will not be visible.
  - For portal based user interfaces, each main portal defines an explicit application service with the access mode **Inquire**, allowing the user interface to be secured independently of the underlying object security. If a user doesn't have access to the portal's application service, the menu item that corresponds with the application service will not be visible. The base product supplies an application service for every portal that is accessible from the menu.
  - Menu items may define an application service. Use this technique for the following scenarios:
    - Suppress a menu item if the underlying application security for the transaction does not provide enough fine grained control. For example, imagine your implementation creates a special BPA script to add a To Do Entry and would like users to use the special BPA rather than the base supplied Add dialogue for To Do Entry. The underlying security settings for To Do Entry should grant Add access to these users given that the special BPA will still add a record. To suppress the base Add dialogue, link a special application service and access mode for the base supplied menu item for To Do Entry Add. Then define a menu entry for the new special BPA for adding.

    - Suppress the add option if a user does not have add security for the object. By default the product does not suppress the add function if a user does not have add access to the object. Rather, the user is prevented from adding the record at the back-end. If your implementation would like to suppress the icon, link the object's application service and the Add access mode to the Add menu item.
  - Zones define an application service
    - For zones linked to a portal, if a user doesn't have access to the zone's application service, the zone will not be visible on the portal. In most cases the zone defines the same application service as its portal. In special cases, such as the zones on the Dashboard, the product supplies separate application services for each zone allowing implementations to determine at a more granular level which users should have access to which zones.
    - For query zones that are configured on a multi-query zone, if a user doesn't have access to the zone's application service, the zone will not be visible in the dropdown on the multi-query zone. In most cases all zones in a multi-query zone define the same application service as the multi-query zone. The product may supply a special application service for one or more zones in a multi-query zone if the functionality is special to certain markets or jurisdictions and not applicable to all implementations.
    - For zones that are used by business services to perform SQL queries, the product supplies a default application service. Security for these zones is not checked by the product as they are used for internal purposes.
  - Business objects define an application service. If the business object defines a lifecycle, the application service must include access modes that correspond to each state. In addition, the standard maintenance object access modes of **Add**, **Change**, **Delete** and **Inquire** are included. The base product business objects are supplied with appropriate application services.
- Other configuration tool objects are securable but the base product typically does not supply special application services for each object. An implementation may supply custom application services and link them to the appropriate record:
  - BPA scripts that are accessible via a menu define an application service with the access mode **Execute**. The base BPA scripts are typically configured with a default application service, which may be overridden by an implementation.
  - Business Services define an application service with the access mode **Execute**. This is needed for business services that may be executed from an external system, for example via an inbound web service. Base business services are configured with a default application service, which may be overridden by an implementation.
- Users are granted access to application services via *user groups*. For example, you may create a user group called Senior Management and give it access to senior manager-oriented pages and portals.
  - When you grant a user group access to an application service with multiple access modes, you must also define the access modes that are allowed. Often the access modes correspond to an action on a user interface. For example, you may indicate a given user group has **inquire**-only access to an application service, whereas another user group has **add**, **change**, **cancel** and **complete** access to the same service. Refer to *action level security* for more information.
  - If the application service has *field level security* enabled, you must also define the user group's security level for each secured field on the transaction.
  - And finally, you link individual *users* to the user groups to which they belong. When you link a user to a user group, this user inherits all of the user group's access rights.

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

Importing LDAP Users and User Groups

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 all users, a group of users, or a single user. You can resynchronize your LDAP users and groups at any time.

For information on how to set up your system to import users and groups from an LDAP store as well as how to do the import, refer to *Importing Users and Groups*.

Action Level Security

When you grant a user group access to an *application service*, you must indicate the actions to which they have access.

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

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

Field Level Security

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

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

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

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

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

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

Encryption and Masking

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

Only users whose database access user ID has the appropriate encryption key can retrieve decrypted information. Please refer to your database's encryption guidelines for how to encrypt data.

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

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

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

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

Identify the Fields to Be Masked

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

Classify each field into one of the following categories:

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

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

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

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

Create a Security Type For Each Logical Field

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

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

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

- The Control Central - Account Information page contains a map zone that retrieves data by invoking a service script. One of the elements in this script's schema holds the person's social security number and it is subject to masking.

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

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

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

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

Create An Algorithm For Each Security Type

A masking algorithm must be created for each security type.

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

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

- **Masking Character:** Enter the character used to overwrite a field's value for users who can only see masked values.
- **Number of Unmasked Characters:** If some of the field value should remain unmasked at the end of the string, enter the number of unmasked characters.
- **Unmasked Characters**: If some characters in a field value should never be masked, enter them. For example, if a phone number can contain dashes and parenthesis and you don't want these characters masked, you would enter `-()..`
- **Application Service:** Enter the application service that you created above.
- **Security Type:** Enter the security type that you created above.
- **Authorization Level:** Enter the authorization level that a user must have to see this field unmasked. If you followed the recommendations above, this will be **2**.

Create A Feature Configuration For Each Secured Element

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

Add an option for every field that you defined in *Identify The Fields To Be Masked.*

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

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

Schema Based Object Field Masking

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

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

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

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

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

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

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

The option value may look like this: **field="PER_ID_NBR", alg="algorithm name", where="ID_TYPE_CD='SSN'"**

Please note the following important points for schema based masking:

- **Limitation of 'where' field** Although the main use of a 'where' clause for schema oriented elements is to mask certain elements in a list based on a 'type', it is also possible to mask a single field in the schema based on the value of another field. For example, imagine that a customer submits a registration form that defines an ID type and ID value. Although this data is not in a list, the implementation may still want to only mask the ID value if the ID type is "SSN". The framework is only able to mask an element in the schema based on a 'where' clause if the element in the 'where' clause is a "sibling" in the schema.
  - If the element to be masked is in a list, the element in the 'where' clause must be in the same list.
  - If an element to be masked maps to a real column in a table, the element in the 'where' clause must also map to a real column in the table.
  - If an element to be masked maps to the CLOB of a table as a single element, the element in the 'where' clause must map to the same CLOB as a single element.
- **Multiple feature option entries for the same field.** It's possible that different schemas in the system have a similar type of data that may be masked based on different conditions. For example, imagine that an implementation has different schemas that captured or referenced person identifiers in different ways:
  - One schema captures a single person ID without any corresponding "type" record and it should always be masked using Algorithm CM_SSN_MASK:

    ```
    <personSSN mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
    ```
  - One schema captures a person ID and a corresponding ID Type and it should be masked with Algorithm CM_SSN_MASK if the type is "SSN" and masked with algorithm CM_FEIN_MASK if the type is "FEIN".

    ```
    <personIdType mapXML=BO_DATA_AREA mdField=ID_TYPE_CD/>
    <personId mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
    ```
  - One schema captures a person ID and a corresponding ID Type and it has the same masking rules as the previous schema, but a different field name is used for the ID Type code. (This scenario could happen if for example a different label is desired for ID Type on the user interface for this schema.)

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

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

  1. **field="PER_ID_NBR", alg="CM_SSN_MASK"**
  2. **field="PER_ID_NBR", alg="CM_SSN_MASK", where="ID_TYPE_CD='SSN'"**
  3. **field="PER_ID_NBR", alg="CM_FEIN_MASK", where="ID_TYPE_CD='FEIN'"**
  4. **field="PER_ID_NBR", alg="CM_SSN_MASK", where="CM_ID_TYPE='SSN'"**
  5. **field="PER_ID_NBR", alg="CM_FEIN_MASK", where="CM_ID_TYPE='FEIN'"**

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

- If more than one sibling element matches a 'where' clause, a runtime error is issued. For example if a schema has an element that references "mdField=ID_TYPE_CD" and an element that references "mdField=CM_ID_TYPE", this is an error. Additionally, if multiple elements reference mdField=ID_TYPE_CD", this is an error.
- If one and only one sibling element matches a 'where' clause, the value of the element is compared to the values defined in the 'where' clause. If it finds a match on the value, the appropriate masking algorithm is applied. If no match is found (for example, the Person ID Type is "LICENSE") the element is displayed as is.
- If no sibling element matches a 'where' clause and a feature option exists with no 'where' clause (option 1 above), then the masking algorithm of the option with no 'where' clause is applied.

- **Changing the value in the 'where' clause.** If your implementation has some users that are allowed to change records where some data is masked based on a condition, it is recommended to design the user interface to reset the masked value when the value in the 'where' clause changes. For example, if a user is prevented from viewing a person's social security number, but the user is allowed to make updates to the person's record, changing the value of the Person ID Type should reset the Person ID Number. This would ensure that the user does not 'unmask' the social security number by simply changing the ID Type.

Records Maintained Using Page Maintenance

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

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

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

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

Characteristic Data

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

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

Masking Fields in Explorer Zones or Info Strings

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

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

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

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

Search Service Results

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

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

Grant Access Rights To The User Groups

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

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

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

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

Additional Masking Information

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

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

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

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

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

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

When you receive an upgrade:

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

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

The Big Picture of Row Security

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

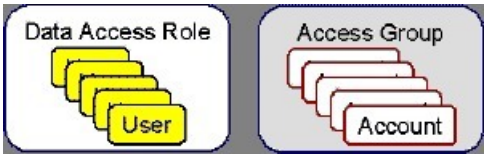
By granting a user access rights to an account, you are actually granting the user access rights to the account's bills, payment, adjustments, orders, etc.

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

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

Access Groups, Data Access Roles and Users

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



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

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

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

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

Defining Application Services

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

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

Application Service - Main

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

Description of Page

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *SC_APP_SERVICE*.

Application Service - Application Security

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

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

This section describes the available zones on this page.

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

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

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

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

- Click the **Description** link to navigate to the User Group - Users page for the user group.

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

Defining Security Types

Security types are used to define the types of *field level security*.

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

Security Type - Main

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

Description of Page

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_SC_TYPE*.

Defining User Groups

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

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

User Group - Main

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

> **CAUTION:** Application services may not be changed or removed from the **ALL_SERVICES** user group. Refer to *The Base Package Controls One User, One User Group, And Many Application Services* for an explanation.

Description of Page

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *SC_USER_GROUP*.

User Group - Application Services

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

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

Description of Page

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

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

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

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

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

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

If you require additional security options, often referred to as "field level" security, then you use **Security Type Code** and assign an **Authorization Level** to each. When a new application service is added, the system will display a message indicating how many security types are associated with this application service. Use the search to define each Security

Type Code and indicate the appropriate Authorization Level for this user group. Refer to *Field Level Security* for more information about security types.

User Group - Users

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

Description of Page

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

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

The following fields are included for each user:

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

> **NOTE:** You can also add a user to a user group using *User - Main*.

Defining Access Groups

> **FASTPATH:** Refer to *The Big Picture of Row Security* for a description of how access groups are use to restrict access to specific objects.

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

Description of Page

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

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

> **NOTE:** You can also use *Data Access Role - Access Group* to maintain a data access role's access groups.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_ACC_GRP*.

Defining Data Access Roles

> **FASTPATH:** Refer to *The Big Picture of Row Security* for a description of how access groups are use to restrict access to specific objects.

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

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

Data Access Role - Main

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

Description of Page

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

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_DAR*.

Data Access Role - Access Group

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

Description of Page

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

> **NOTE:** You can also use *Access Group - Main* to maintain an access group's data access roles.

Defining Users

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

The user maintenance transaction is the same transaction invoked when the user launches ***Preferences*** .

Chapter 4

User Interface Tools

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

Defining Menu Options

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

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

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

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

Menu - Main

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

Description of Page

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

Owner indicates if this menu line is owned by the base package 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.

Long Label is not used at this time.

Menu Description is only enabled for menu types of **Menu**, **Admin** and **Page Action Menu**.

Sequence is only enabled for the **Menu** and **Admin**.

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 package 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.

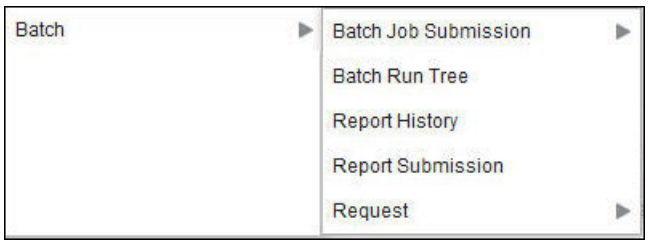
Menu - Menu Items

After 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 package 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 package 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). It is also recommended to define the “search” menu item as sequence 1 and the “add” menu item a 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, the system knows which entry is the Add based on whether a **Image GIF Location and Name** is populated. For new menu entries, to indicate which menu item should be the Add, enter a reference, such as **/images/contextAdd.gif**.
  - **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 (the menu item that does not define an Image GIF) is used to build the menu entry text. The label long on the menu line that defines the Add option is information only.
- The **Override Label** is provided in case you want to override the base-package's label.

- Use **Application Service** and **Access Mode** to easily suppress a menu item for one or more users. Refer to *Application Security* for more information.

The Big Picture of System Messages

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

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

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

NOTE: You cannot change the base package's text. If the message is "owned" by the base package, you cannot change the base package's message or long description (if you could, your changes would be overwritten during an upgrade). If you want your users to see a different message or long description other than that supplied by the base package, display the message on the **Details** tab and enter your desired verbiage in the "customer specific" fields (and flush the cache).

Defining System Messages

The contents of this section describe how to maintain messages that appear throughout the system.

Message - Main

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

Description of Page

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

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

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

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

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

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

- Click the - button to delete a message.
- Click the + button to add a new message. After clicking this button, enter the following fields:

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

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

- **Owner** indicates if this message number is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a message. This information is display-only.
- Click the go to button to enter a detailed description of the message. Clicking this button transfers you to the **Details** tab.
- To change a message, simply overtype its **Message Text** and save the category. Note, you must transfer to the Details tab if you want to update a message's detailed Description.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_MSG*. In addition, messages are used throughout the system for error messages and other system messages.

Message - Details

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

> **NOTE: You don't have to use the scroll.** Rather than scrolling through the messages, you can display a message by clicking the respective go to button in the grid on the main tab.

Description of Page

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

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

For base-package messages, you can use this page as follows:

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

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

- Use **Message Text** to define the message.

  You can use the % *n* notation within the message text to cause field values to be substituted into a message. For example, the message text **The %1 non-cash deposit for %2 expires on %3** will have the values of 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).

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

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

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

The Big Picture of Portals and Zones

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

There Are Three Types of Portals

There are three broad classes of portals:

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

  There is only one dashboard portal. This portal and several zones are delivered as part of the base-package. Your implementation can add additional zones to this portal. Please contact customer support if you need to add zones to the dashboard portal.

Common Characteristics of All Portals

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

Portals Are Made Up of Zones

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

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

Configuring Zones for a Portal

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

- The order in which the zone should appear. An implementation may configure an override sequence to change the order zones on a base delivered portal.
- Whether the zone is visible on the portal. Zones delivered in the base package 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 package portal / zone.

  FASTPATH: Refer to *Zones May Appear Initially Collapsed When a Page Opens* for more information.

FASTPATH: Refer to *Defining Portals* for more information about this configuration.

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

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

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

FASTPATH: Refer to *User — Portal Preferences* for more information about how users configure their zones.

Granting Access to Zones

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

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

Please note the following with respect to zone application security:

- For most base package portals, all the zones for all the portals reference the same application service that is used to grant access to the main (stand-alone) portal for the page. In other words, if the user has access to the page, then he has access to all the zones on all portals for the page. There may be exceptions to this rule for certain portals.
- For a base package multi-query zones, typically the individual query zones and the multi-query zone reference the same application service that is used to grant access to the main (stand-alone) portal for the page. However, there may be individual query zones provided with a unique application service. This may occur when the query option is unusual and

not applicable to all users or even to all implementations. If a user does not have security access to an individual query zone, that option will not be available in the dropdown.

- For base package portals that are configured to show on portal preferences, it is common that the portal contains different types of zones that may be applicable to different types of users. Typically these types of portals will deliver a unique application service for each zone so that an implementation may configure which user groups are allowed to view each zone. For these types of portals, please note the following:
  - A user's *Portal Preferences* page contains a row for a zone regardless of whether the user has access rights to the zone. Because of this, the system displays an indication of the user's access rights to each zone.
  - If a user's access rights to a zone are revoked, the zone will be suppressed when the user navigates to the respective portal.
  - Revoking a user's access rights does not change the user's *portal preferences* (i.e., a user can indicate they want to see a zone even if they don't have access to the zone - such a zone just won't appear when the respective portal appears).

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

Common Characteristics of Stand-Alone Portals

The topics that follow describe addition characteristics specific to *stand-alone portals*.

Putting Portals on Menus

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

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

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

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

Granting Access to A Portal

An *application service* is associated with each *stand-alone portal*. A user must be granted access rights to the respective application service in order to see a portal.

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

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

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

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

Custom Look and Feel Options

The default look and feel of the application can be customized via feature configuration and cascading style sheets. The base package 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 package allows for the conditional inclusion of customer styles into the system style set. The custom style may override any style provided by the base package. 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 of off. By default it is turned on.
- Override the default help image with a custom image
- The location of the help image, either before or after the element.

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

Setting Up Portals and Zones

The topics in this section describe how to set up portals and zones. Please refer to the *The Big Picture of Portals and Zones* for background information.

Defining Zone Types

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

> **NOTE:** It is not very common for an implementation to introduce their own 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 package 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.
- **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 package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a parameter. This information is display-only.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_ZONE_HDL*.

Zone Type Parameter Comments

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

Defining Zones

The contents of this section describe how to maintain zones.

Zone - Main

Select **Admin** > **Zone** > **Search** to maintain a zone.

Description of Page

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

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

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

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

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

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

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

> **NOTE:** Zones on the *dashboard portal* are always the width of the dashboard.

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

Use **Override Zone Help Text** to override the existing embedded help text for this zone.

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_ZONE*.

Zone - Portal

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

Description of Page

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

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

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

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

Additional Zone Topics

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

Zone Help Text

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

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

**This zone summarizes revenue in 4 periods:
**

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

- **** and **** are the HTML tags used to indicate that the surrounded text should be bold
- **** and **** are the HTML tags used to indicate that the surrounded text should be blue.

The following are other useful HTML tags:

- **
** causes a line break in a text string. If you use **

** a blank line will appear.
- **<I>** causes the surrounded text to be italicized

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

You can also use "spans" to customize the look of the contents of a text string. For example, your text string could be **revenue.** This would make the word "revenue" appear as large, bold, Courier text. Please refer to a Cascading Style Sheets (CSS) reference manual or website for more examples.

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

This zone summarizes revenue in 4 periods:

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

- The 2nd period is the period before the 1st period

- The 3rd period is the same as the 1st period, but in the previous year

- The 4th period is the period before the 3rd period

The traffic light's color is determined as follows:

- The ratio of the 1st and 3rd period is calculated

- If this value is between 80 and 100, yellow is shown

- If this value is < 80, red is shown

- If this value is > 100, green is shown

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

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

Common Zone Parameters

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

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/	**NAVIGATION**	Indicates that the action is navigation to a page.

Mnemonic	Description	Valid Values	Comments
	icon/button is clicked. This is ignored when the type=ASIS.	**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 page or executing a BPA script. The mnemonic supports passing multiple values. In each case the target context field or BPA script variable is defined first followed by an equal sign, followed by source data defined using one of the valid values defined in the next column. One or more values may be defined. Each context value is defined separated by spaces. The whole set of context values should be surrounded by square brackets.	FIELD_NAME	Indicates that the value should be taken from the field with this name from portal context, global context or the page data model. The mnemonic sourceLoc is used for defining the source.
		xpath	Indicates that the value should be taken from a schema field, represented by the Xpath, displayed in this zone. This is valid when the zone is displaying a UI Map.
		'constant'	Indicates that the value defined in single quotes should be passed.
sourceLoc=	This mnemonic defines the source of the FIELD_NAME's value in the context mnemonic. If this mnemonic is left blank, the default behavior is as follows: - 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.	**G**	Indicates that the field's value is retrieved from the global context.
		P	Indicates that the field's value is retrieved from the portal context.
		D	Indicates that the field's value is retrieved from the page data model.

Mnemonic	Description	Valid Values	Comments
class=	Use this mnemonic to override the look and feel of the link / icon / button using a different CSS style.	'className1' 'className2'	Enter one or more classes in single quotes. Multiple class names may be provided.
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.

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/userdId)]**
- **type=ASIS asis=['<A class="oraLink" href="www.google.com">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.	FIELD_NAME	
datasource=	This mnemonic defines the source of the filter's pre-populated value defined in the name mnemonic. 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, 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.

Mnemonic	Description	Valid Values	Comments
		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.
		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 and in the input area.	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_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.
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.	nameFromUIMap	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.

Mnemonic	Description	Valid Values	Comments
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.
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]	A valid table name and field name are required. The WHERE_FLD and WHERE_VALUE are optional, but if entered, both are required. Use this to only encrypt the field in FIELD_NAME if another field has a certain value. For example encrypt=[CI_PERSON,PER_ID_NBR,ID_TYPE_NBR,'SSN'].

Examples:

- **label=F1_NBR_DAYS type=NUMBER**
- **label=F1_SHOW_ALL_REQ_FLG type=LOOKUP lookup=F1_SHOW_ALL_REQ_FLG**
- Filter value where a Filter UI Map is defined and Description is one of the filters. **type=STRING xpath=description likeable=S**
  - **type=STRING label=DESCR likeable=S divide=below**
  - **label=REQ_TYPE_CD type=TABLE table=F1_REQ_TYPE**

Hidden Filters

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

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

These parameters are applicable to the zone types

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

A hidden filter is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
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).	**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.

Mnemonic	Description	Valid Values	Comments
	- If still no value, the global context is checked. - If still no value, the zone appears as per the poprule mnemonic.		
poprule=	This mnemonic controls what happens if the hidden filter is not present.	**R**	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.
		O	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**	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.
		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.

Mnemonic	Description	Valid Values	Comments
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.

Mnemonic	Description	Valid Values	Comments
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.
	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.
	Example of a schema:
	`<schema>`
	`<accountInfo type="list">`
	`<accountId/>`
	`<name/>`
	`<amount/>`
	`<process/>`
	`</accountInfo>`
	`</schema>`
	Example of list and context mnemonics.
	list=accountInfo
	context=[accountId=ACCT_ID name=C2 amount=P3 process='O']
		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.
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.

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 package 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 package or by your implementation (**Customer Modification**).

Where Used

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

Defining Portals

This transaction is used to define / change portals.

Portal - Main

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

Description of Page

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

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

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

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

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

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_PORTAL*.

Portal - Options

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

Description of Page

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

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

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

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

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

Defining Display Icons

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

Description of Page

Each icon requires the following information:

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_DISP_ICON*.

Defining Navigation Keys

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

Navigation Key Types

There are two types of navigation keys:

- **System navigation keys** define locations where the URL is derived from other entities within the system. The items that are derived include the program location (i.e., physical location on the server), file name (i.e. program component name), and the file extension (e.g. COB). System navigation keys refer to program components, such as tab menus or tab pages.

- **External navigation keys** define locations simply as a URL. External URLs can be specified as relative to the product web server or fully qualified. External navigation keys always launch in a new instance of a browser window. Examples of external navigation keys include online help links and URLs to external systems.

Navigation Key vs. Navigation Option

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

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

The Flexibility of Navigation Keys

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

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

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

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

Linking to External Locations

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

- Define a *navigation key* that specifies the URL of the location. For example, define an external navigation key that as a URL of **http://www.oracle.com/**.
- Define a *navigation option* that specifies from where in the system a user can go to your external location. For example, define a navigation option with a usage of **Favorites** or with a usage of **Menu**. Your navigation option points to the navigation key you defined above.
- Add your navigation option to the appropriate location within the system. For example, have users add the navigation option to their *Favorite Links* or add the navigation option as an item on a *menu*.

Overriding Navigation Keys

Your implementation may choose to design a program component (e.g., a maintenance transaction or search page) to replace a component provided by the system. When doing this, the new navigation key must indicate that it is overriding

the system navigation key. As a result, any menu entry or navigation options that reference this overridden navigation key automatically navigates to the custom component.

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

To create an override navigation key, you need to:

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

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

Maintaining Navigation Keys

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

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

Description of Page

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

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

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

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

FASTPATH: Refer to *Navigation Key Types* for more information about system and external navigation keys.

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

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

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

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

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

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

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_MD_NAV*.

Defining Navigation Options

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

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

Many navigation options are shipped with the base package 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** > **Navigation Option** > **Search** to maintain a navigation option.

Description of Page

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

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

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

Owner indicates if this navigation option is owned by the base package 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 package.

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 for 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 launch of the **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 ID field for the attachment that will be opened.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_NAV_OPT*.

Navigation Option - Tree

This page contains a tree that shows how a navigation option is used. Select **Admin** > **Navigation Option** > **Search** 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.

Chapter 5

Database Tools

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

Defining Table Options

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

Table - Main

Navigate using **Admin** > **Table** > **Search** .

Description of Page

Table Name is the identifier of this table.

Description contains a brief description of the table.

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

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

Data Group ID is used for internal purposes.

Table Usage is not in use at this time.

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

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

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

Table Volume Type specifies the expected amount of data the table will hold. This is for information purposes only and is not used by any system processing. Valid values are **High Volume, Low Volume, Medium Volume,** and **Unclassified**.

The volume of data in a particular table in the system may differ greatly from one implementation to another based on unique business requirements. The values populated for base product tables are set to volumes that are typical but may not be true for a given implementation. The value may be updated to reflect the situation for a given implementation.

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

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

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

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

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

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

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

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

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

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

Enable Data Dictionary defines if the table is to be included in the *Data Dictionary* application viewer.

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

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

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

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

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

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

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

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

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

Help Text contains additional information about the table.

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

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

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

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

Table - Table Field

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

Description of Page

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

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

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

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

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

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

The **Allow Customization** is only applicable if the table is an Admin System Table. It indicates fields that an implementation is allowed to change for a base owned record. Changes to the field value of one of these types of fields by an implementation are maintained when upgrading to a new version of the product.

Standard Time Type is only enabled if the Table indicates that the Date/Time Data Type is **Local Standard Time**. Each field that represent date/time should define if it uses **Logical Standard Time, Physical Standard Time** or uses a **Referenced Time Zone**.

Sequence is a unique sequence of this field with respect to other fields on the table.

The **Label** column is used to define a special label for this field when it relates to this table if it should be different from the field's label. Note that this only impacts the label on a fixed page user interface. Labels on portal based user interfaces do not use this information.

The **Override Label** is provided if an implementation wants to override the base-product label.

> **NOTE:** If you want the Override Label to be shown in the *data dictionary*, you must regenerate the data dictionary.

Help Text contains any additional information about the field with respect to its use on this table.

Field Usage is not used at this time.

Extract for Translation is only visible in a development environment. For tables marked to extract for translation, each translatable field on the table should indicate **Yes**.

Translation Context is only visible in a development environment. It is used to provide information to a translator about the nature and purpose of the data in this field on this table.

> **NOTE: Changes to base owned table / fields.** Only the following attributes of table / fields that are owned by the product are modifiable: Audit Delete, Audit Insert, Audit Update, Override Label.

Table - Constraints

Select **Admin** > **Table** > **Search** and navigate to the **Constraints** tab to view the constraints defined on the table.

Description of Page

The fields on this page are protected as only the product development group may change them.

This page represents a collection of constraints defined for the table. A constraint is a field (or set of fields) that represents the unique identifier of a given record stored in the table or a field (or set of fields) that represents a given record's relationship to another record in the system.

Constraint ID is a unique identifier of the constraint.

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

Constraint Type Flag defines how the constraint is used in the system:

- **Primary Key** represents the field or set of fields that represent the unique identifier of a record stored in a table.
- **Logical Key** represents an alternate unique identifier of a record based on a different set of fields than the Primary key.
- **Foreign Key** represents a field or set of fields that specifies identifying and non-identifying relationships to other tables in the application. A foreign key constraint references the primary key constraint of another table.
- **Conditional Foreign Key** represents rare relationships between tables where a single field (or set of fields) may reference multiple primary key constraints of other tables within the application as a foreign key.

When **Enable Referential Integrity** is checked, the system validates the integrity of the constraint when a row in the table is modified.

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

Referring Constraint ID is the **Primary Key** constraint of another table whose records are referenced by records stored in this table.

Referring Constraint Table displays the table on which the Referring Constraint ID is defined. You can use the adjacent go-to button to open the table.

Additional Conditional SQL Text is only specified when the constraint is a **Conditional Foreign Key**. The SQL represents the condition under which the foreign key represents a relationship to the referring constraint table.

NOTE: Additional Conditional SQL Syntax. When specifying additional conditional SQL text, all table names are prefixed with a pound (#) sign.

The Constraint Field grid at the bottom of the page is for maintaining the field or set of fields that make up this constraint.

Field is the name of the table's field that is a component of the constraint.

Sequence The rank of the field as a component of the constraint.

The Referring Constraint Field grid at the bottom of the page displays the field or set of fields that make up the **Primary key** constraint of the referring constraint.

Field is the name of the table's field that is a component of the referring constraint.

Sequence is the rank of the field as a component of the referring constraint.

Table - Referred by Constraints

Select **Admin** > **Table** > **Search** and navigate to the **Referred By Constraints** tab to view the constraints defined on other tables that reference the **Primary Key** constraint of this table.

Description of Page

This page is used to display the collection of constraints defined on other tables that reference the table.

Referred By Constraint Id is the unique identifier of the constraint defined on another table.

Referred By Constraint Owner indicates if this constraint is owned by the base package or by your implementation (**Customer Modification**).

Prime Key Constraint Id is the **Primary Key** constraint of the current table.

Prime Key Owner indicates if this prime key is owned by the base package or by your implementation (**Customer Modification**).

Referred By Constraint Table is the table on which Referred By Constraint Idis defined.

When **Enable Referential Integrity** is checked, the system validates the integrity of the constraint when a row in the table is modified.

The grid at the bottom of the page displays the **Field** and **Sequence** for the fields that make up the constraint defined on the other table.

Defining Field Options

The topics in this section describe the transaction that can be used to view information about a field and to change the name of a field on the various pages in the system.

Field - Main

Open this page using **Admin** > **Field** > **Search** .

Description of Page

Field Name uniquely identifies this field.

CAUTION: As described in *System Data Naming Convention* for most system data tables, the base product follows a specific naming convention. However, this is not true for the Field table. If you introduce new fields, you must prefix

the field with **CM**. If you do not do this, there is a possibility that a future release of the application could introduce a new field with the name you allocated.

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

Base Field indicates that the field inherits some of its definitions from another field.

Data Type indicates if the type of data the field will hold. Valid values are **Character**, **Character Large Object**, **Date**, **DateTime**, **Number**, **Time**, **Varchar2** and **XML Type**. This field is protected if the field refers to a Base Field.

Ext Data Type or Extended Data Type is used to further define the type of data for certain data types. Valid values are **Currency Source, Day of Month, Duration, Money, Month of Year, Flag** and **Switch**. This field is protected if the field refers to a Base Field.

Precision defines the length of the field. In the case of variable length fields, it is the maximum length possible. For number fields that include decimal values, the precision includes the decimal values. This field is protected if the field refers to a Base Field.

Scale is only applicable for number fields. It indicates the number of decimal places supported by the field. This field is protected if the field refers to a Base Field.

Sign is only applicable for numbers. It indicates if the data may contain positive or negative numbers.

Value List is only visible for products that had at one point included COBOL. It defines the copybook that includes the list of valid values for the field.

Description contains the label of the field. This is the label of the field that appears on the various pages on which the field is displayed. Note, the field's label can be overridden for a specific table by specifying an Override Label on the *table / field* information. However, this override is not used in portal based user interfaces. It is only applicable if the field is displayed on fixed page user interfaces.

Java Field Name is the reference to this field used in Java code.

Override Label is used if an implementation would like to override the label that appear on user interfaces in the system.

CAUTION: For fixed pages, if the field's label is overridden for a specific table, that override takes precedence. In this is the case the override on the *table / field* page should be used.

Work Field indicates that the field does not represent a database table column.

Help Text is used to provide field level embedded help to this field. If the field is displayed on a user interface that supports display of embedded help, this text may be displayed.

Use **Override Help Text** to override the existing embedded help text for this field.

Extract for Translation is only visible in a development environment. It indicates whether or not the field and its description should be included in an extract of translatable strings when doing a product translation. This flag may be set to “No” for work fields whose label is not visible to a user on any user interface.

Translation Context is only visible in a development environment. It is used to provide information to a translator about the use of the field’s label so that an appropriate translation can be provided.

Field - Tables Using Field

Select **Admin** > **Field** > **Search** and navigate to the **Tables Using Field** tab to view the tables that contain a field.

Description of Page

The grid on this page contains the **Tables** that reference the **Field**. You can use the adjacent go to button to open the *Table Maintenance* transaction.

Defining Maintenance Object Options

A maintenance object is a group of tables maintained together within the system.

Maintenance Object - Main

Select **Admin** > **Maintenance Object** > **Search** to view information about a maintenance object.

Description of Page

Most maintenance objects are provided with the base package. An implementation can introduce custom maintenance objects when needed. Most fields may not be changed if owned by the base package.

Enter a unique **Maintenance Object** name and **Description**. **Owner** indicates if this business object is owned by the base package or by your implementation (**Customer Modification**).

> **IMPORTANT:** If you introduce a new maintenance object, carefully consider its naming convention. Refer to *System Data Naming Convention* for more information.

Service Name is the name of the internal service associated with the maintenance object.

Click the **View XML** hyperlink to view the XML document associated with the maintenance object service in the *Service XML Viewer*.

Click the **View MO** hyperlink to view the definition of the maintenance object in the *Maintenance Object Viewer*.

The grid displays the following for each table defined under the maintenance object:

- **Table** is the name of a given table maintained as part of the maintenance object.
- **Table Role** defines the table's place in the maintenance object hierarchy. Only one **Primary** table may be specified within a maintenance object, but the maintenance object may contain many **Child** tables.
- **Parent Constraint ID** specifies the *constraint* used to link the table to its parent table within the maintenance object table hierarchy.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

Maintenance Object - Options

Use this page to maintain a maintenance object's options. Open this page using **Admin** > **Maintenance Object** > **Search** 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** > **Maintenance Object** > **Search** 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. Click *here* to see the algorithm types available for this system event.
Determine BO	Optional	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. 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. Click *here* to see the algorithm types available for this system event.
Information	Optional	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. 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. Click *here* to see the algorithm types available for this system event.
Revision Control	Optional	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.

System Event	Optional / Required	Description
		Click *here* to see the algorithm types available for this system event.
Transition	Optional	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. Note that some 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. Click *here* to see the algorithm types available for this system event.
Transition Error	Optional	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 transition error somewhere, typically on the maintenance object's log. 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. 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. Click *here* to see the algorithm types available for this system event.

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 Lookup Options

Some special fields have values that are defined by the base-package development group. For example:

- The navigation option usage field has potential values of **Favorites**, **Go To** and **Menu**. On the database, these values are represented by the values of **FAV**, **GOTO** and **MENU**, respectively.
- The access mode (used by application security) defines the actions that are available on various application services, for example **Add**, **Change**, **Delete**, **Complete** and **Cancel**. On the database, these values are represented by the values of **A**, **C**, **D**, **CO**, and **CA** respectively

We call these types of fields "lookup fields" (because we have to "look up" the descriptions on the "lookup" table when they are displayed on a transaction).

The two examples described above represent the two different types of lookup fields.

- The first example (navigation option usage) is a lookup field where you cannot add, remove or change the valid values.
- The second example (access mode) is a lookup field where you are allowed to add valid values.

We differentiate between these two types of lookups because the first type of lookup field (e.g., navigation usage option) controls logic in the system and if you change the valid values, the system will not work and if you add valid values, they will not be used by any system logic. For the second type of lookup field (e.g., access mode), your implementation may define additional values to be used by your customer modifications.

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.

Lookup - Main

Select **Admin** > **Lookup** > **Search** to maintain lookup values.

Description of Page

Field Name is the name of the field whose lookup values are maintained in the grid. If you need to add a new lookup field, you must first add a *Field* with 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 may be modified:

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

The "lookup" table is a single table that holds valid values for many columns on many tables. Standard lookups are used when the element's valid values are limited to a predefined list. These work well when:

- The size of the valid values is 4 characters or shorter.
- The valid values are a simple code / description pair.

However, there can be situations when the valid values exceed 4 digits. For example, when the value is interfaced to another system and this system uses longer values (and you don't want to do a transformation in the interface).

There are also situations when more than just a description is needed for each valid value. For example, what if the value has both a description and a list containing different options based on existing conditions, such as the type of To Do to create based on an activity's service class?

In these situations, extendable lookups can be used to store these types of values. You use the Extendable Lookup portal to create and maintain extendable lookups.

Extendable Lookup - Main

Open this page using **Admin** > **Extendable Lookup** .

Description of Page

In the Extendable Lookup Search zone you can search for lookup values by either Business Object or Description. Click **Refresh** to view the search results.

In the search results, select the **Description** link to go to the Extendable Lookup Value List zone for that record. In the Extendable Lookup Value List zone the following actions are available:

- Click **Add** in the zone’s title bar to add a new extendable lookup value.
- Click **Go To Search** in the zone’s title bar to go back to the Extendable Lookup Search zone.
- Click the Broadcast icon to view the details for the extendable lookup value.
- Click **Edit** to edit the details for the extendable lookup value.
- Click **Delete** to delete the extendable lookup value.

The Big Picture Of Audit Trails

The topics in this section describe auditing, enabling auditing for fields, and auditing queries that you can use to view audit records.

Captured Information

When auditing is enabled for a field, the following information is recorded when the field is changed, added and/or deleted (depending on the actions that you are auditing for that field):

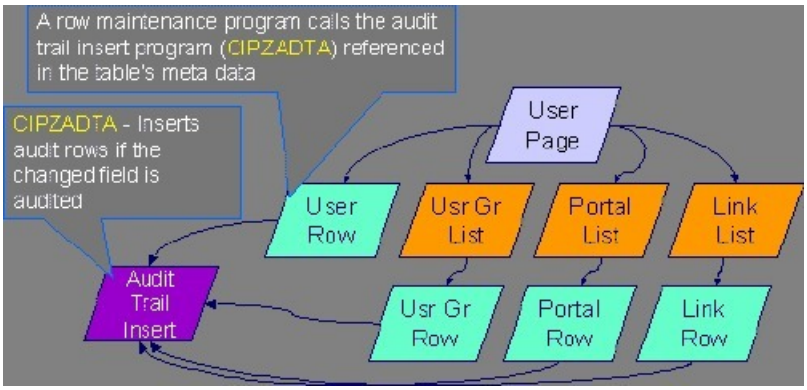
- User ID

- Date and time
- Table name
- Row's prime key value
- Field name
- Before image (blank when a row is added)
- After image (blank when a row is deleted)
- Row action (add, change, delete)

How Auditing Works

You enable auditing on a table in the table's meta-data by specifying the name of the table in which to insert the audit information (the audit table) and the name of the program responsible for inserting the data (the audit trail insert program). Then you define the fields you want to audit by turning on each field's audit switch in the table's field meta-data. You can audit fields for delete, insert and update actions.

Once auditing is enabled for fields in a table, the respective row maintenance program for the table assembles the list of changed fields and calls the audit trail insert program (**CIPZADTA** is supplied with the base package). If any of the changed fields are marked for audit, **CIPZADTA** inserts audit rows into the audit table (**CI_AUDIT** is the audit table supplied with the base package).



NOTE: Customizing Audit Information. You may want to maintain audit information other than what is described in *Captured Information* or you may want to maintain it in a different format. For example, you may want to maintain audit information for an entire row instead of a field. If so, your implementation team can use **CIPZADTA** and *CI_AUDIT* as examples when creating your own audit trail insert program and audit table structures.

The Audit Trail File

Audit log records are inserted in the audit tables you define. The base product contains a single such table (called *CI_AUDIT*). However, the audit insert program (**CIPZADTA**) is designed to allow you to use multiple audit tables.

If you want to segregate audit information into multiple tables, you must create these tables. Use the following guidelines when creating new audit tables (that use the CIPZADTA audit insert program):

- The new audit tables must look identical to the base table (*CI_AUDIT*).
- The new tables must be prefixed with **CM** (e.g., **CM_AUDIT_A**, **CM_AUDIT_B**, etc.).
- The name of the new table must be referenced on the various tables whose changes should be logged in the new table.

NOTE: Interesting fact. It's important to note if you use your own tables (as opposed to using the base package table called **CI_AUDIT**), the SQL used to insert and access audit trail records (in **CIPZADTA**) is dynamic. Otherwise, if the base package's table is used, the SQL is static.

How To Enable Auditing

Enabling audits is a two-step process:

- First, you must turn on auditing for a table by specifying an audit table and an audit trail insert program.
- Second, you must specify the fields and actions to be audited for the table.

The following topics describe this process.

Turn On Auditing For a Table

In order to tell the system which fields to audit, you must know the name of the table on which the field is located. You must specify the audit table and the audit trail insert program for a table in the table's meta-data.

NOTE: Most of the system's table names are fairly intuitive. For example, the user table is called *SC_USER*, the navigation option table is called *CI_NAV_OPT*, etc. If you cannot find the table using the search facility on the *Table Maintenance* page, try using the *Data Dictionary*. If you still cannot find the name of the table, please contact customer support.

To enable auditing for a table:

- Navigate to the *Table maintenance* page and find the table associated with the field(s) for which you want to capture audit information.
- Specify the name of the **Audit Table**.

NOTE: Specifying the Audit Table. You can use the audit table that comes supplied with the base package (**CI_AUDIT**) to audit multiple tables and fields. All the audit logs are combined in a single table (**CI_AUDIT**). However, you can also have a separate audit table for each audited table. Refer to *The Audit Trail File* for more information.

- Specify the name of the **Audit Program** (**CIPZADTA** is the default audit program supplied with the base package).

CAUTION: By default, none of a table's fields are marked for audit. Even though you have enabled auditing for a table, you must still specify the fields and actions on those fields to be audited (see below).

Specify The Fields and Actions To Be Audited

The system only audits actions (insert, update and delete) made to fields that you want audited.

To specify the fields and actions to be audited:

- Navigate to the *Table - Table Field maintenance* page for a table on which you have enabled auditing.
- For each field you want to audit, specify the actions you want to audit by turning on the **Audit Delete**, **Audit Insert** and **Audit Update** switches as appropriate.

NOTE: Note. You can also turn on the audit switches using the Field grid at the bottom of the *Table maintenance page*.

CAUTION: Audit Program Caching! The audit program from the table meta-data is read into a program cache on the application server whenever the date changes or when the server starts. If you implement new auditing on a table, your audit trail does not become effective until this program cache is reloaded. In other words, new audits on tables where the audit program was not previously specified do not become effective until the next day (or the next restart of the application server). However, if you change the fields to be audited for a table where the audit program is already in the cache, your changes are effective immediately.

Audit Queries

There are two queries that can be used to access the audit information.

Audit Query by User

This transaction is used to view changes made by a user that are stored on a given *Audit Trail File*.

CAUTION: The system only audits changes that you've told it to audit. Refer to *The Big Picture Of Audit Trails* for more information.

Navigate to this page by selecting **Admin** > **Audit Query 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 where changed.
- **Audited Field Name** is the name of the field that was changed.
- **Audit Action** indicates whether the row action was **Add**, **Change** or **Delete**.
- **Field Value Before Update** contains the content of the field before the change. This column is blank if information was **Added**.
- **Field Value After Update** contains the content of the field after the change. This column is blank if information was **Deleted**.

Audit Query by Table / Field / Key

This transaction is used to view audited changes made to a given table.

CAUTION: The system only audits changes that you've told it to audit. Refer to *The Big Picture Of Audit Trails* for more information.

NOTE: Most of the system's table names are fairly intuitive. For example, the user table is called *SC_USER*, the navigation option table is called *CI_NAV_OPT*, etc. If you cannot find the table using the search facility on the *Table Maintenance* page, try using the *Data Dictionary*. If you still cannot find the name of the table, please contact customer support.

This transaction can be used in several different ways:

- You can view all audited changes to a table. To do this, enter the **Audited Table Name** and leave the other input fields blank.
- You can view all audited changes to a given row in a table (e.g., all changes made to a given user). To do this, enter the **Audited Table Name** and row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**).
- You can view all audited changes to a given field in a table (e.g., all changes made to all customers' rates). To do this, enter the **Audited Table Name** and the **Audited Field Name**.
- You can view all audited changes to a given field on a specific row. To do this, enter the **Audited Table Name**, the **Audited Field Name**, and row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**).

Navigate to this page by selecting **Admin** > **Audit Query 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 where 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 algorithms types before algorithms.

You can specify the sequence when you define the export bundle or when you import the bundle to the target environment.

Recursive Key References

Recursive foreign keys result when one object has a foreign key reference to another object that in turn has a foreign key reference to the first object.

For example, a zone has foreign keys to its portals, which have foreign keys to their zones. If the objects you want to bundle have recursive relationships, you must create a 'bundling add' business object that has only the minimal number of elements needed to add the entity. A bundling add business object for a zone contains only the zone code and description, with no references to its portals. In the same way, a bundling add business object for a portal defines only its code and description.

When you apply the bundle, the system initially adds the maintenance object based on the elements defined in the bundling add business object. Before committing the bundle, the system updates the maintenance object with the complete set of elements based on its physical business object.

Owner Flags on Bundled Entities

The owner flag of the entities in an import bundle must match the owner flag of the target environment.

If you need to import objects that your source environment does not own, you must replace the owner flag in the import bundle with the owner flag of the target environment.

Configuring Maintenance Objects for Bundling

All base package meta-data objects are pre-configured to support bundling. All other objects must be manually configured.

If a base package maintenance object is pre-configured for bundling, the **Eligible For Bundling** option will be set to "Y" on the Options tab for the maintenance object.

To configure other objects for bundling, review the configuration tasks below and complete all those that apply:

Complete this configuration task...	for...
Make maintenance objects eligible for bundling	All objects to be included in the bundle
Add a foreign key reference	All objects to be included in the bundle
Create a physical business object	All objects to be included in the bundle
Create a bundling add business object	Objects with recursive foreign key references
Add the Current Bundle zone	All objects, if you want the Current Bundle zone to appear on the maintenance object's dashboard. This is not required by the bundling process.
Create a custom Entity Search zone and add it to the Bundle Export portal	All objects, if you want them to be searchable in the Bundle Export portal. This is not required by the bundling process.

Making Maintenance Objects Eligible for Bundling

The "Eligible For Bundling" maintenance object option must be set to "Y" for all bundled objects.

To make maintenance objects eligible for bundling:

1. Select **Admin** > **Maintenance Object** > **Search** and search for the maintenance object.
2. On the Options tab, add a new option with the type **Eligible For Bundling**.
3. Set the value to "Y" and click **Save**.

Adding a Foreign Key Reference

Each maintenance object in a bundle must have a foreign key reference. Bundling zones use the foreign key reference to display the standard information string for the maintenance object.

To add a foreign key reference to the maintenance object:

1. Select **Admin** > **FK Reference** > **Add** and set up a foreign key reference for the primary table of the maintenance object.
2. Select **Admin** > **Maintenance Object** > **Search** and search for the maintenance object.
3. On the **Option** tab, add a new option with the type **Foreign Key Reference**. The value is the name of the foreign key reference you just created.

Creating a Physical Business Object

Each maintenance object in a bundle must have a physical business object. The physical business object's schema represents the complete physical structure of the maintenance object, and includes elements for all fields in the maintenance object's tables. The bundling process uses this schema to generate the XML for the import bundle.

To create a physical business object for the maintenance object:

1. Select **Admin** > **Business Object** > **Add** and specify the maintenance object.
2. Click **Generate** in the **BO Schema** dashboard zone to generate a schema that looks like the physical structure of the maintenance object.
3. Save the physical business object.
4. Select **Admin** > **Maintenance Object** > **Search** and search for the maintenance object.
5. On the **Option** tab, add a new option with the type **Physical Business Object**. The value is the name of the physical business you just created.

Creating a Bundling Add Business Object

If the objects to be bundled have recursive foreign key references, you must create a bundling add business object to avoid problems with referential integrity.

To create a bundling add business object:

1. Select **Admin** > **Business Object** > **Add** and specify the maintenance object.
2. Click **Generate** in the **BO Schema** dashboard zone to generate a schema that looks like the physical structure of the maintenance object.
3. Remove all elements that are not essential. Typically, only a code and description are required.
4. Save the physical business object.
5. Select **Admin** > **Maintenance Object** > **Search** and search for the maintenance object you want to bundle.
6. On the **Option** tab, add a new option with the type **Bundling Add BO**. The value is the name of the bundling add business object you just created.

Adding the Current Bundle Zone

If you want the Current Bundle zone to appear on the maintenance object's dashboard, you must add the Current Bundle zone as a context-sensitive zone for the maintenance object.

To add the Current Bundle zone to the maintenance object:

1. Select **Admin** > **Context Sensitive Zone** and search for the navigation key for the maintenance object.
2. Add the Current Bundle zone, F1-BNDLCTXT, to that navigation key.

Adding a Customized Entity Search Query Zone to the Bundle Export Portal

If you want the maintenance object to be searchable in the Bundle Export portal, you must first create an entity-specific query zone to search for the maintenance object. Then you must create a customized entity search zone that references this new query zone. Finally, you must add the customized entity search zone to the Bundle Export portal.

To make the maintenance object searchable:

1. Create an entity-specific query zone to search for the maintenance object:
   a) Select **Admin** > **Zone** > **Search** and search for one of the base query zones, such as the Algorithm Search zone, F1-BNALGS.
   b) Click the **Duplicate** button in the page actions toolbar.
   c) Enter a name for the new zone.
   d) Click **Save**.
   e) Locate the **User Filter** parameter in the parameter list. Add SQL to search for the maintenance object(s) you want to appear in the zone.
   f) Save the query zone.
2. Create a customized entity search zone:

   This step only needs to be done once. If you already have a customized search zone in the Bundle Export portal go to step 3
   a) Select **Admin** > **Zone** > **Search** and search for the F1-BNDLENTQ Entity Search zone.
   b) Duplicate this zone (as described above).
   c) Remove any references to base query zones.
3. Add the new entity-specific query zone to the customized entity search zone:
   a) Locate the customized entity search zone for your Bundle Export portal. This is the zone created in Step 2.
   b) Locate the Query Zone parameter in the parameter list. Add the name of the query zone you created in Step 1.
   c) Save the entity search zone.
4. Add the customized entity search zone to the Bundle Export portal:

   This step needs to be done only once.
   a) Select **Admin** > **Portal** > **Search** and search for the Bundle Export portal, F1BNDLEM.
   b) In the zone list, add the entity search zone you created in Step 2. (Add the new zone after the base entity search zone).
   c) Save the portal.

Working with Bundles

Use the Bundle Export portal to create an export bundle. The export bundle contains a list of entities to be exported from the source environment. When you are ready to import the objects, use the Bundle Import portal to import the objects to the target environment.

Creating Export Bundles

An export bundle contains a list of entities that can be imported into another environment.

To create an export bundle:

1. Log on to the source environment from which objects will be exported.
2. Select **Admin** > **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** > **Bundle Export** > **Search** 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** > **Bundle Import** > **Add** .
4. In the **Bundle Actions** zone, click **Edit XML**.
5. Paste the contents of the clipboard (or text file if you created one) into the **Bundle Detail** zone.
6. Make any necessary changes to the XML and click **Save**. The status of the import bundle is set to Pending.

   NOTE: Use caution when editing the XML to avoid validation errors.

7. To remove entities from the import bundle or change their sequence, click **Edit**. Enter your changes and click **Save** to exit the Edit dialog.
8. When you are ready to apply the bundle, click **Apply**. The import bundle state is set to Applied and the entities are added or updated in the target environment.

Editing Export Bundles

You can add or remove entities from an export bundle when it is in Pending state. You can also change the sequence of entities.

To edit to an export bundle that has already been bundled, you must change the bundle state by selecting the **Back to Pending** button on the Bundle Export portal.

To edit a pending export bundle:

1. Open the bundle in edit mode.
2. Click **Edit** on the Export Bundle portal.
3. Make any necessary changes on the edit dialog and then click **Save**.

Editing Import Bundles

You can remove entities from an import when it is in Pending state. You can also change the sequence of entities and edit the generated XML.

To edit a pending import bundle:

1. Open the bundle in edit mode.
2. To edit the XML snapshot, click **Edit XML**. Edit the XML code as needed, then click **Save**.

   NOTE: Use caution when editing the XML to avoid validation errors.

3. To remove entities or change their sequence, click **Edit**. Make any necessary changes and click **Save**.

Revision Control

The topics in this section describe the revision control features in the application.

About Revision Control

Revision control is a tool provided for the development phase of a project to allow a user to check out an object that is being worked on. In addition, it captures the version of the maintenance object when users check in an update, maintaining a history of the changes to the object.

If revision control is enabled for an object you must check out the object to change it. While the object is checked out no one else can work on it. You can revert all changes made since checking out an object, reinstate an older version of an object, recover a deleted object, and force a check in of an object if someone else has it checked out.

NOTE: Revision control does not keep your work separate from the environment. Because the metadata for maintenance objects is in the central database, any changes you make to an object while it is checked out will be visible to others and may impact their work.

Many of the maintenance objects used as configuration tools are already configured for revision control, but it is turned off by default. For example, business objects, algorithms, data areas, UI maps, and scripts are pre-configured for revision control.

Turning On Revision Control

Revision control is turned off by default for maintenance objects that are configured for revision control.

To turn on revision control:

1. Add the base package **Checked Out** zone to the Dashboard portal.
   a) Select **Admin** > **Portal** > **Search** .
   b) Search for the portal CI_DASHBOARD.
   c) In the zone list for the **Dashboard** portal, add the zone F1-USRCHKOUT.
2. Set up application security.

   For users to have access to revision control, they must belong to a user group that has access to the application service F1-OBJREVBOAS.
3. Add the revision control algorithm to the maintenance object that you want to have revision control.

   This step must be repeated for each maintenance object that you want to have revision control.
   a) Select **Admin** > **Maintenance Object** > **Search** and search for the maintenance object that you want to have revision control.
   b) On the **Algorithms** tab of the maintenance object, add the revision control algorithm F1-REVCTL.

Configuring Maintenance Objects for Revision Control

Most configuration tool maintenance objects are pre-configured for revision control. You can configure other maintenance objects for revision control, as well.

To configure other objects for revision control:

1. Create a physical business object for the maintenance object.

   A physical business object is one that has a schema with elements for all of the fields for the tables in the maintenance object. Follow these steps to create a physical business object:
   a) Select **Admin** > **Business Object** > **Add** and specify the maintenance object.

b) Use the **BO Schema** dashboard zone to generate a schema that looks like the physical structure of the maintenance object.

c) Save the physical business object.

d) Select **Admin** > **Maintenance Object** > **Search** and search for the maintenance object for which you want to enable revision control.

e) On the **Options** tab of the maintenance object add a new option with the type **Physical Business Object**. The value is the name of the physical business object that you just created.

2. Add a foreign key reference to the maintenance object.

   The revision control zones will display the standard information string for the object based on the foreign key reference. Follow these steps to create a foreign key reference:

   a) Select **Admin** > **FK Reference** > **Add** and set up a foreign key reference for the primary table of the maintenance object.

   b) Select **Admin** > **Maintenance Object** > **Search** and search for the maintenance object.

   c) On the **Options** tab of the maintenance object, add a new option with the type **Foreign Key Reference**. The value is the name of the foreign key reference that you just created.

3. Add the **Revision Control** zone to the maintenance object.

   a) Select **Admin** > **Context Sensitive Zone** and search for the navigation key for the maintenance object.

   b) Add the Revision Control zone, F1-OBJREVCTL, to that navigation key.

4. Add the revision control algorithm to the maintenance object.

   a) Select **Admin** > **Maintenance Object** > **Search** and search for the maintenance object that you want to have revision control.

   b) On the **Algorithms** tab of the maintenance object, add the revision control algorithm F1-REVCTL.

Working with the Revision Control Zones

You use two zones in the dashboard to work with revision controlled objects when revision control is turned on.

The **Revision Control** zone gives you several options for managing the revision of the currently displayed object. This zone also shows when the object was last revised and by whom. This information is linked to the **Revision Control Search** portal which lists all of the versions of the object.

Using the Revision Control zone you can:

- Check out an object in order to change it.
- Check in an object so others will be able to work on it.
- Revert the object back to where it was at the last checkout.
- Force a check in of an object that is checked out by someone else. You need special access rights to force a check in.
- Delete an object.

The **Checked Out** zone lists all of the objects that you currently have checked out. Clicking on an object listed in this zone will take you to the page for that object. The zone is collapsed if you have no objects checked out.

See *Revision Control Search* for more information about Check In, Force Check In, and Check Out one or more records simultaneously.

Checking Out an Object

You must check out a revision controlled object in order to change it.

An object must have revision control turned on before you can check it out.

> **NOTE:** When you first create or update an object a dialog box informs you that the object is under revision control. You can select **OK** to check out the object and save your changes, or **Cancel** to stop the update.

1. Go to the object that you want to work on.
2. Select **Check Out** in the **Revision Control** dashboard zone.

Checking In an Object

You must check in a revision controlled object in order to create a new version of it. Checking in an object also allows others to check it out.

1. Select a link in the **Checked Out** dashboard zone to go to the object that you want to check in.
2. Select **Check In** in the **Revision Control** dashboard zone.
3. Provide details about the version:
   - In the **External References** field state the bug number, enhancement number, or a reason for the revision.
   - In the **Detailed Description** field provide additional details regarding the revision.
   - In the **Keep Checked Out** box specify if you want to keep the object checked out. If you keep the object checked out then your revision is a new version that you can restore later.
   - In the **Add To Bundle** box specify if the object belongs to a bundle.
4. Select **OK** to check in the object.

Reverting Changes

Reverting changes will undo any changes you made since you checked out an object.

To revert changes:

1. Go to the object that you want to revert.
2. Select **Revert** in the **Revision Control** dashboard zone.
3. In the confirmation dialog box select **OK** to confirm the action or **Cancel** to return to the object page.

Once reverted, the object can be checked out by another user.

Forcing a Check In or Restore

You can force a check in if an object is checked out by another user and that person is not available to check it in.

You must have proper access rights to force a check in or restore.

To force a check in or restore:

1. Go to the object that is checked out by another user.

2. Select **Force Check In** or **Force Restore** in the Revision Control zone.

The **Force Check In** option is the same as a regular check in. The **Force Restore** option checks in the object and restores it to the previously checked in version.

Deleting an Object

If revision control is turned on for an object, you must use the **Revision Control** zone to delete it.

The object must be checked in before it can be deleted.

To delete a revision controlled object:

1. Go to the object that you want to delete.
2. Select **Delete** in the **Revision Control** zone.
3. Provide details regarding the deletion.
4. Select **OK** to delete the object.

The system creates a revision record before the object is deleted so that the deleted object can be restored.

Restoring an Object

You can restore an older version of either a current object or a deleted object.

An object must be checked in before an older version can be restored.

To restore an object:

1. Go to the **Revision History** portal for the object.

   If the object was deleted you must search for it by going to **Admin** > **Revision Control** .
2. Select the desired entity by clicking the hyperlink in the **Details** column.
3. Locate the row in the version history that has the version that you want to restore and click **Restore**.
4. In the confirmation dialog box select **OK** to confirm the action or **Cancel** to return to the object page.

Working with the Revision Control Portal

The **Revision Control** portal lists information about each version of a revision controlled object.

You can navigate to the **Revision Control** portal from either a link in the **Revision Control** dashboard zone or by going to **Revision Control** portal through **Admin**.

If you want to find the Revision History entry for an earlier version or deleted object, you must search for the object using the **Revision Control Search** portal. Once you select the desired entry, you can restore a previous version of the object clicking **Restore** in the row for the version that you want to restore. You can also see the details of each version by clicking the broadcast icon for that version.

See *Working with Revision Control Zones* for more information about tasks that can be performed through Revision Control.

Revision Control Search

The **Revision Control Search** portal allows users to search for entities that have a revision history. The **Search By** dropdown provides additional functionality so that users can search for revisions that are associated to theirs or other's user ID. Users can also **Check In**, **Force Check In**, or **Check Out** one or more entities through this portal.

Zone Options

- **Revision History Search** *(default)* allows the user to query for revised entities based on a combination of criteria.
  - In the **User ID** field, enter the user ID that is associated with a revision.
  - In the **External Reference** field, enter an ID from an external system and is associated with a revision.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key 1**, **Key 2**, **Key 3**, **Key 4**, **Key 5** fields, enter the primary identifier(s) for the revised entity. Typically, the entity only requires a single key, but some entities require more than one (for example, Oracle Utilities Customer Care and Billing SA Type require CIS Division and SA Type).
  - In the **Status** dropdown menu, select the entity status for your search.
- **Check In** allows the user to search for entities currently checked out to the logged in user ID and a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them in.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key** field, enter the primary identifier(s) for the revised entity.
- **Force Check In** allows the user to search for entities that are currently checked out by other user IDs (excluding the logged in user ID) based on a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them in.
  - In the **Checked Out By User** field, enter the user ID that has the entity in a Checked Out status.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key** field, enter the primary identifier(s) for the revised entity.
- **Check Out** allows the user to search for entities currently checked in user ID and a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them out.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key** field, enter the primary identifier(s) for the revised entity.

  Please see *Working with Revision Control Zones* for more information about working with individual entities.

Information Lifecycle Management

Information Lifecycle Management (ILM) is designed to address data management issues, with a combination of processes and policies so that the appropriate solution can be applied to each phase of the data's lifecycle.

Data lifecycle typically refers to the fact that the most recent data is active in the system. As time progresses, the same data becomes old and unused. Older data becomes overhead to the application not only in terms of storage, but also in terms of performance. This older data's impact can be reduced by using advanced compression techniques, and can be put into

slower and cheaper storage media. Depending on how often it’s accessed, it can be removed from the system to make an overall savings of cost and performance. The target tables for ILM are transactional tables that have the potential to grow and become voluminous over time.

The Approach to Implementing Information Lifecycle Management

This section describes the product approach to implementing ILM for its maintenance objects (MOs).

NOTE: The term archiving is used to cover any of the possible steps an implementation may take in their data management strategy, including compression, moving to cheaper storage, and removing the data altogether.

Age is the starting point of the ILM product implementation for some of its high volume data. In general "old" records are considered eligible to be archived. In the product solution, maintenance objects (MOs) that are enabled for ILM have an ILM date on the primary table and the date is set to the record's creation 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.

However, 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.

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.

The decision of how old a record is has to be performed before it is evaluated for archive eligibility. This can be determined by a system-wide setting, or uniquely for each maintenance object.

NOTE: The following information outlines the **configuration** for maintenance objects that support ILM in the product. This **is not** a complete summary of the process to implement Information Lifecycle Management. *For more information about implementing Information Lifecycle Management, see the Oracle Utilities Application Framework: Database Administration Guide.*

In order to implement ILM, the following **requirements** must be met:

- **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. *For information about the process to implement referential integrity, see the Oracle Utilities Application Framework: Database Administration Guide.*
- **Partitioning**: This provides a way in which the data can segregate into multiple table partitions and will help in better manageability of the lifecycle of the data. *For information about partitioning, see the Oracle Utilities Application Framework: Database Administrator’s Guide.*
- **Special Table Columns**: Maintenance objects that support ILM include two specific columns: ILM Archive Switch (**ILM_ARCH_SW**) and ILM Date (**ILM_DT**).

There are **additional** configuration points to further define how ILM works within the product:

- **Maintenance Object Options**: There are MO options that include configuration related to ILM.
- **Master Configuration**: An ILM master configuration record defines system wide ILM information.
- **Batch Processes**: Batch processes are defined to execute the ILM eligibility algorithm.

- **ILM Algorithm**: Each maintenance object that is configured for ILM defines an eligibility algorithm that executes the logic to set the ILM Archive switch appropriately.

Maintenance Object Options

Application metadata must be set up for the ILM crawler batch process to successfully manage the lifecycle of the data. The key metadata infrastructure is related to setting up the value of the Maintenance Object (MO) Options types.

The following outlines the MO Option types that are in the base product:

- **ILM Crawler Batch Control**: This option identifies the batch control that is used to manage historic data. This process invokes the MO's ILM eligibility algorithm for entity instances whose retention period has elapsed.
- **ILM Retention Period in Days**: This option is used by the base ILM crawler batch control. Use this option to override the default retention period defined in the ILM master configuration.

Master Configuration

The Master Configuration business object for ILM Configuration defines global parameters for all ILM eligible Maintenance Objects. In addition, the UI for this master configuration record displays summary information about all the maintenance objects that are configured to use ILM.

A Master Configuration record must be added that includes a value in the Default Retention Period field.

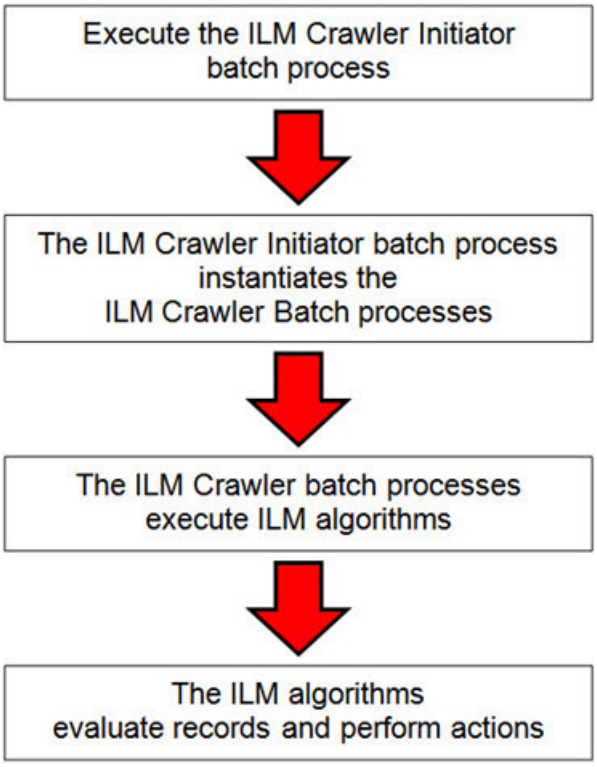
- **Default Retention Period**: The ILM Crawler uses this field’s value (in days) if the retention period is not specified in the MO Options.

Batch Processes

There are two main types of batch controls 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 *parent* 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.

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.

Chapter 6

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. Note that the tools described here are controlled completely by meta-data - no programming required!

Business Objects

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?

The fundamental idea behind a business object is that it should closely match the end user's conception of an object (e.g.; the specific information used to define a customer). This is in marked contrast to an application developer's notion of an object (e.g.; the normalized database tables used to capture generic person information). The business object configuration tool described here is a bridge between the two notions; a business object maps the end user's concept of an object to the physical database structures, and services, used to maintain the information. In other words, a business object is typically a simplification of the maintenance object.

Figure 1: Customer - as a Business Object

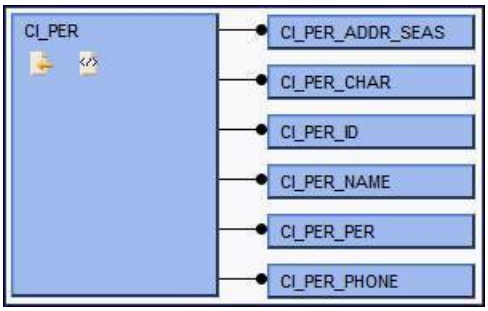


Figure 2: Customer - as Defined in the Database

Another definition of a business object is a structure that allows an implementation to define a simpler view of a maintenance object. For example, a tax management COTS team can set up business objects for individual taxpayer, corporation, and partnership all as simpler views of the Person maintenance object. Yet another use of business objects is for managing market messages: separate business objects can be defined for a multitude of market messages, all of which belong to a single market message MO.

A Business Object Has Properties

A business object has properties (AKA, elements). Properties such as "Social Security Number" and "Home Phone" are applicable to an individual taxpayer, whereas "Corporate Name" is applicable to a corporation. The structure of a business object is defined using an XML schema. The purpose of the schema is to describe the business object's properties and map them to the corresponding maintenance object fields.

```
<schema>
  <name mapField="ENTITY_NAME">
        <row mapChild="CI_PER_NAME">
              <SEQ_NUM is="1" />
              <NAME_TYPE_FLG default="PRIM" />
              <PRIM_NAME_SW  default="true" />
        </row>

  </name>
  <driversLicense mapField="PER_ID_NBR">
        <row mapChild="CI_PER_ID">
              <ID_TYPE_CD is="DL" />
        </row>
  </driversLicense >
   <socialSecurity mapField="PER_ID_NBR">
        <row mapChild="CI_PER_ID">
              <ID_TYPE_CD is="SSN" />
              <PRIM_SW default="true" />
        </row>
  </socialSecurity>
  <email mapField="EMAILID"/>
```

Figure 3: Customer - Business Object Schema (partial)

Keep in mind that many maintenance objects have child table collections (e.g., a collection of phone numbers, or a collection of characteristics on an account) and therefore the definition of where a property resides can be sophisticated. For example, defining a business object property like "Name" requires the implementer to define:

- The child table on the maintenance object that holds names (e.g., CI_PER_NAME)
- The unique identifier of the instance of the collection in which the value resides (e.g., where Name Type Flag = **PRIM(ary)**)
- The name of the field in which the property resides (e.g., ENTITY_NAME)

> **NOTE: Flatten.** We use the terms "flattening" and "flatten out" to describe a business object property that resides in a collection but is defined as a singular element.

Some maintenance objects allow data to be stored as an XML structure field (CLOB) with the entity. Business object properties may reside in the MO's XML extension. You will typically map business object properties to an MO XML extension when the property does not have to be exposed for SQL indexing or joining (e.g., most fields on a tax form or the elements in a market message).

Some business objects may have child tables that allow data to be stored as a CLOB. You can map to these CLOB fields in your schema.

> **CAUTION: CLOB is only for non-indexed / joined fields!** We'd like to stress that putting properties in an XML document (CLOB) field should only be done for properties that will never be indexed or joined in an SQL statement. If you need to "join" to a property you must map it to an indexed field, like a characteristic, rather than to the CLOB. This is because databases do not commonly support indexes for elements in a CLOB field (at least not today).

> **NOTE: Schema Definition Tips.** A context sensitive "Schema Tips" zone appears when you open the *Business Object* page to assist you with the schema definition syntax. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

Scripts and *Inbound Web Services* support interaction with business objects. It is also possible to interact with business objects directly from a Java class.

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, you may use this type of algorithm to default element values.
- 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. Refer to the "Schema Tips" context sensitive zone associated with the Business Object maintenance page for more information on this attribute.

Refer to *Business Object - Algorithms* for more information on the various types of algorithms.

The system applies business object rules (schema based and algorithms) whenever a business object instance is added, changed or deleted. This is only possible when the call is made via the maintenance object service. For example, when made via business object interaction ("invoke BO"), the MO's maintenance page or inbound web services that reference the BO or the MO service, etc. In addition 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 the business object cannot be determined for a maintenance object instance business object rules are not applied.

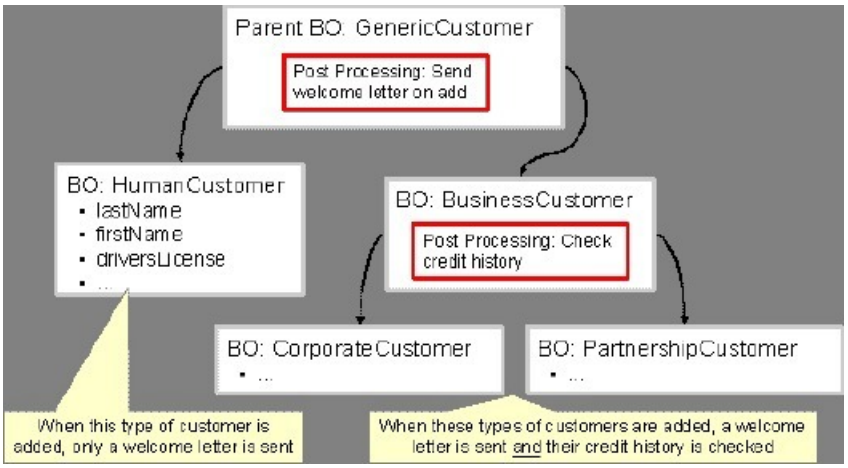
NOTE:
Pre-Processing is special. The pre-processing algorithm plug-in spot is unique in that it only applies during a BO interaction. It is executed prior to any maintenance object processing. It means that when performing add, change or delete via the maintenance object service, the pre-processing plug-in 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.

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.

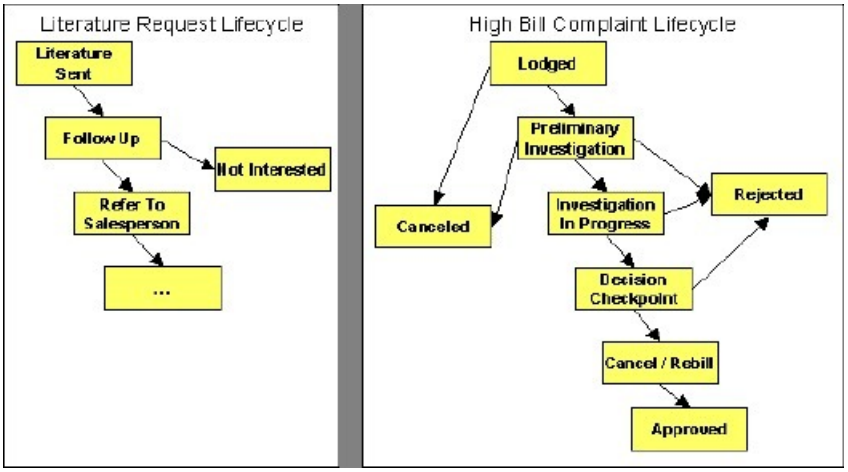
NOTE: Data structures are not inherited. While you can declare schemas on parent business objects, their children will not inherit them. A good practice is to design child business object schemas to **include** their parent business object's schema.

NOTE: 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.

Logic To Take Place When Entering A State

You can define algorithms that execute before a business entity enters a given state. For example, you could develop an algorithm that requires a cancellation reason before an entity is allowed to enter the *Canceled* state.

You can also incorporate state auto-transitioning logic within this type of algorithms. Refer to *auto-transition* for more information.

Logic To Take Place When Exiting A State

You can define algorithms that execute when a business entity exists a given state. For example, you could develop an algorithm that clears out error messages when a given entity exits the *Error* state.

Monitor Rules

You can define algorithms to monitor a business entity while it is in a given state. This type of logic is typically used to check if the conditions necessary to *transition* the entity to another state exist (and, if so, transition it). For example, transition an entity to the *Canceled* state if it's been in the *Error* state too long. Another common use is to perform ancillary work while an entity is in a given state. For example, update statistics held on the object while it's in the *Active* state.

Monitor algorithms are invoked when a business entity first enters a state and periodically after that in batch. You have the option to defer the monitoring of a specific state until a specific monitoring batch job runs. You do so by associating the state with a specific monitoring process. In this case the system will only execute the monitoring rules of this state when that specific batch process runs. This may be useful for example in a market-messaging world where you do not want an inbound message processed when it is received; rather, you want to wait until a later point in time (maybe at the end of the day).

A monitor algorithm can carry out any business logic. In addition it can optionally tell the system to do either of the following:

- Stop monitoring and transition to another state. The system will not call any further monitoring algorithm plugged in on the state and attempt to transition the entity to the requested new state.
- Stop monitoring. Same as above except that no transition takes place. You may want to use this option to prevent transitions while some condition is true.

If none of the above is requested the system keeps executing subsequent monitoring algorithms.

FASTPATH: Refer to *Business Object - Lifecycle* for more information on how to set up state-specific algorithms.

Inheriting Lifecycle

If a business object references a parent business object, it always *inherits* its lifecycle from the highest-level business object in the hierarchy. In other words, only the highest-level parent business object can define the lifecycle and the valid state transitions for each state. Child business objects, in all levels, may still extend the business rules for a given state by introducing their own state-specific algorithms.

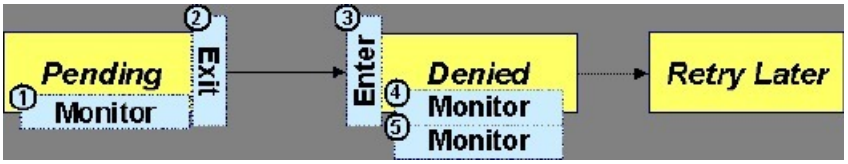
The system executes all state-specific algorithms at all levels in the inheritance chain starting from the highest-level parent business object moving on to lower levels.

Auto-Transition

In a single transition from one state to another, the system first executes the **Exit** algorithms of the current state, transitions the entity to the new state, executes the **Enter** algorithms of the new state followed by its **Monitor** algorithms. At this point if a **Monitor** algorithm determines that the entity should be further automatically transitioned to another state the remaining monitoring algorithm defined for the current state are NOT executed and the system initiates yet another transition cycle.

Notice that an **Enter** algorithm can also tell the system to automatically transition the entity to another state. In this case the remaining **Enter** algorithm as well as all **Monitor** algorithms defined for the current state are NOT executed.

The following illustration provides an example of an auto-transition chain of events.



In this example a business entity is in a Pending state. While in that state a **Monitor** algorithm determines to auto-transition it to the Denied state. At this point the following takes place:

- No further **Monitor** algorithms of the Pending state are executed
- Pending state **Exit** algorithms are executed
- The system transitions the entity to the Denied state
- Denied state **Enter** algorithms are executed. No further auto-transition is requested.
- Denied state **Monitor** algorithms are executed. No further auto-transition is requested.

Keeping An Entity In Its Last Successful State

By default, any error encountered while transitioning a business entity from one state to another rolls back ALL changes leaving the entity in its original state.

When applicable, the Maintenance Object can be configured to always keep an entity in its last successful state rather than rolling all the way back to the original state. This practice is often referred to as "taking save-points". In case of an error, the entity is rolled back to the last successfully entered state and the error is logged on the *maintenance object's log*. Notice that with this approach no error is returned to the calling process, the error is just logged!

The logic to properly log the error is in a **Transition 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

The base package provides a periodic monitoring batch process for each maintenance object that supports soft state transition. The process periodically executes the monitoring algorithms associated with the current state of an entity, excluding states explicitly referencing a deferred monitoring batch process.

A deferred monitoring process works in the same way but only considers entities whose current state references this particular batch control as their monitor process. Deferred monitoring is only needed for certain states based on business requirements.

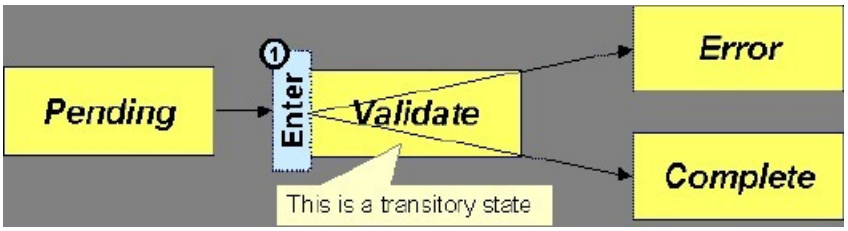
Your business rules will dictate the execution frequency of each monitoring process and the order in which they should be scheduled.

> **NOTE: Updates to the business object.** When the monitor algorithms indicate that the business object should transition, the monitor batch processes are responsible for ensuring the business object is transitioned appropriately and that the appropriate exit, enter and monitor algorithms are executed. Please note that the business object is not updated using a call to the maintenance object service and therefore the *business rules* plugged in to the business object are not executed.

Transitory States

You can define a state as **Transitory** if you do not wish the business entity to ever exist in that particular state.

The following illustrates a lifecycle with a transitory Validate state.



In this example, the business entity is saved still not validated in the Pending state. At some point, the user is ready to submit the entity for validation and transitions it into a transitory state Validate whose **Enter** rules contain the validation logic. The responsibility of the transitory state's **Enter** algorithms is to decide if the entity is valid or in error and then transitions it into the appropriate final state. In this scenario, you may not ever want the business entity to exist in the Validate state.

Let's also assume that the maintenance object in this example is practicing " *save-points*" and requires the entity to be kept in its last successful state. If an error were to occur during the transition from **Validate** to the next state, the system would roll back the entity back to Pending, and not Validate even though the entity has successfully entered the Validate state. Refer to the *Auto Transition* section for more information.

State Transitions Are Audited

Most Maintenance Objects that support soft lifecycle definition also have a log to hold significant events throughout a business entity's lifecycle. For example, log entries are created to record:

- When the business entity is created (and who created it)
- When its status changes (and who changed it)
- If a transition error occurred (and the error message)

- References to other objects created throughout the entity's lifecycle. For example, if a To Do entry is created as part of processing the entity, the To Do Entry is referenced in the log.
- Manual entries added by a user (think of these as "diary" entries)

When a business entity is first created and when it transitions into a new state the system calls **Transition** algorithm(s) plugged in on the *Maintenance Object* to record these events. If the maintenance object supports a log these events can be captures 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.

Defining Reasons 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.
- 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 or to mark status reasons as optional. With this configuration, the framework will automatically get the list of valid reasons for that state 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.

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.

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.

Business Object - Main

Use this page to define basic information about a business object. Open this page using **Admin** > **Business Object** > **Search**

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.

Click the **View Schema** hyperlink to view the business object's expanded schema definition. Doing this opens the *schema viewer* window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

Click the **View MO** hyperlink to view the maintenance object in the *Maintenance Object Viewer*. You may find it useful to leave the application viewer window open while defining your business object schema.

The options grid allows you to configure the business object to support extensible options. Select the **Option Type** 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*.

Where Used

Follow this link to open the data dictionary to view the tables that reference *F1_BUS_OBJ*.

Business Object - Schema

Use this page to maintain a business object's schema. Open this page using **Admin** > **Business Object** > **Search** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the business object.

Click the **View Schema** hyperlink to view the business object's expanded schema definition. Doing this opens the *schema viewer* window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

The ***Schema Designer*** zone allows you to edit the business object's schema. The purpose of the schema is to describe the business object's properties and map them to corresponding maintenance object fields.

NOTE: Generating a Schema A context sensitive "Generate Schema" zone is associated with this page. The zone provides a button that allows the user to generate a basic schema that includes all the fields for all the tables for the BO's maintenance object.

NOTE: Schema Definition Tips. A context sensitive "Schema Tips" zone is associated with this page. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

Business Object - Algorithms

Use this page to maintain a business object's algorithms. Open this page using **Admin** > **Business Object** > **Search** 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	Algorithms of this type may be used to audit certain changes made to business object instances. 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. It is the responsibility of the algorithms to record corresponding audit information. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to *Business Object inheritance* for more information. Click *here* to see the algorithm types available for this system event.
Information	Optional	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. 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. Click *here* to see the algorithm types available for this system event.
Post-Processing	Optional	Algorithms of this type may be used to perform additional business logic after a business object instance has been processed. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to *Business Object inheritance* for more information. Click *here* to see the algorithm types available for this system event.
Pre-Processing	Optional	Algorithms of this type further populates a request to maintain a business object instance right before it is processed. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to *Business Object inheritance* for more information. Click *here* to see the algorithm types available for this system event.
Validation	Optional	Algorithms of this type may be used to validate a business object instance when added, updated or deleted. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to *Business Object inheritance* for more information. Click *here* to see the algorithm types available for this system event.

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** > **Business Object** > **Search** and then navigate to the **Lifecycle** tab.

Description of Page

The **Status** accordion contains an entry for every status in the object's *lifecycle*. The entry appears differently for a child business object as it can only extend its inherited lifecycle by introducing algorithms and options of its own.

Use **Status** to define the unique identifier of the status. This is NOT the status's description, it is simply the unique identifier used by the system. Only the highest-level business object can define lifecycle statuses. For a child business object the inherited status description is displayed allowing navigation to the corresponding entry on the business object defining the lifecycle.

Use **Description** to define the label of the status. This field is hidden for a child business object.

Use **Access Mode** to define the action associated with this status. Refer to *Access Rights* for the details of how to use this field to restrict which users can transition a business entity into this state. This field is hidden for a child business object.

Enter a **Monitor Process** to defer the monitoring of entities in this state until the specific batch process runs. Refer to *Monitor Rules* for more information. This field is hidden for a child business object.

The **Status Reason** dropdown indicates if users should be prompted to provide a specific reason when the business object enters this state. This field appears only if the Status Reason Field is configured as an option on the business object's maintenance object. Valid values are blank, **Optional**, and **Required**. The default value is blank (users are not prompted to provide a status reason). See *Configuring Status Reasons* for more information about status reasons.

Use **Status Condition** to define if this status is an **Initial**, **Interim** or **Final** state. Refer to *One Initial State and Multiple Final States* for more information about how this field is used. This field is hidden for a child business object.

Use **Transitory State** to indicate whether a business entity should ever exist in this state. Only **Initial** or **Interim** states can have a transitory state value of **No**. Refer to *transitory states* for more information. This field is hidden for a child business object.

Use **Alert Flag** to indicate that being in this state warrants an application alert. This may be used by custom logic to provide an alert to a user that entities exist in this state. This field is hidden for a child business object.

Use **Display Sequence** to define the relative order of this status for display purposes. For example when displayed on the status accordion and on the summary tab page. This field is hidden for a child business object.

Algorithms

The **Algorithms** grid contains algorithms that control important functions for a given status. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
- If the algorithm is implemented as a script, a link to the **Script** is provided. Refer to *Plug-In Scripts* for more information.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

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

System Event	Optional / Required	Description
Enter	Optional	Algorithms of this type apply business rules when a business object instance enters a given state. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to *Business Object Inheritance* for more information. Click *here* to see the algorithm types available for this system event.
Exit	Optional	Algorithms of this type apply business rules when a business object instance exits a given state. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to *Business Object Inheritance* for more information. Click *here* to see the algorithm types available for this system event.
Monitor	Optional	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. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to *Business Object Inheritance* for more information. Click *here* to see the algorithm types available for this system event.

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** > **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 **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.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and XAI Inbound 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.

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.

Configuring 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** > **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

The **Business Objects with Status Reason List** zone displays the business objects that have one or more statuses that can have status reasons 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.

The following actions are available:

- Click **Add** to create a new status reason code for this business object.
- Click **Edit** to update the status reason.
- Click **Delete** to delete the status reason.
- Click **Duplicate** to duplicate status reason.

Business Services

In the same way that a *business object* is used to simplify a maintenance object, a business service can be used simplify a back-end service. The rating engine is a good example of a complex back-end service because it must satisfy a vast array of differing requirements. However, it is also true that the actual data required for a specific rating task can be quite simple. For example, if your *script* must calculate a simple cost per ton: you can pass the rating engine just a single service quantity and receive a calculated amount back.

Business services define alternative services to our internal services that are easier to work with. A Business Service provides a "simpler" data interface thus making interaction with the actual service easier. For example, it may flatten out complex collections and set up default values for other pieces of information (like a rate schedule).

```
<input type="group">
    <serviceQuantity dataType="number"/>
  </input>
<output type="group">
    <amount dataType="money"/>
  </output>
```

Figure 4: Weight Calculation Business Service

```
</pageBody>
    <field type="date" name="USER_START_DT">xpath</field>
    <field type="string" name="BSEG_ID">xpath</field>
    <field type="string" name="BSEG_INFO">xpath</field>
    <field type="string" name="SA_INFO">xpath</field>
    <field type="string" name="SA_ID">xpath</field>
    <field type="date" name="START_DT">xpath</field>
    <field type="date" name="END_DT">xpath</field>
    <field type="boolean" name="THIRD_PARTY_SW">xpath</field>
    <field type="date" name="ACCOUNTING_DT">xpath</field>
    <field type="string" name="SP_ID">xpath</field>
    <field type="string" name="PREM_ID">xpath</field>
    <field type="string" name="LANGUAGE_CD">xpath</field>
    <field type="string" name="REV_CL_CD">xpath</field>
    <field type="string" name="RC_DESCR">xpath</field>
    <field type="string" name="RS_CD">xpath</field>
    <field type="string" name="RS_DESCR">xpath</field>
    <list>
        <listHeader>
        </listHeader>
        <listBody>
            <field type="rowActionFlag">xpath</field>
            <field type="string" name="CHAR_TYPE_CD">xpath</field>
            <field type="string" name="CHAR_TYPE_DESCR">xpath</field>
            <field type="string" name="CHAR_VAL">xpath</field>
            <field type="string" name="CHAR_VAL_DESCR">xpath</field>
            <field type="string" name="ADHOC_CHAR_VAL">xpath</field>
            <field type="string" name="CHAR_TYPE_FLG">xpath</field>
        </listBody>
    </list>
    <list>
        <listHeader>
        </listHeader>
        <listBody>
            <field type="rowActionFlag">xpath</field>
            <field type="string" name="RR_SP_ID">xpath</field>
            <field type="bigInteger" name="SEQNO">xpath</field>
            <field type="bigDecimal" name="REG_CONST">xpath</field>
            <field type="string" name="USAGE_FLG">xpath</field>
            <field type="bigInteger" name="USE_PCT">xpath</field>
            <field type="string" name="HOW_TO_USE_FLG">xpath</field>
            <field type="boolean" name="MSR_PEAK_QTY_SW">xpath</field>
            <field type="string" name="UOM_CD">xpath</field>
            <field type="string" name="TOU_CD">xpath</field>
            <field type="string" name="SQI_CD">xpath</field>
```

Figure 5: Rate Application Service (about 30% of actual 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. Keep in mind that many back-end services have child table collections (e.g., a collection of input service quantities, or a collection of output bill lines) and therefore the definition of where a business service property resides can be sophisticated. For example, defining a business service property like "Weight" requires the implementer to define:

- The collection on the business service that holds service quantities (e.g., SQ)
- The unique identifier of the instance of the collection in which the value resides (e.g., where SQI Code = **TONS**)
- The name of the field in which the property resides (e.g., SQI_CD)

```
<input type="group">
   <serviceQuantity mapField="INIT_SQ">
      <row mapList="SQ">
         <SQI_CD is="TONS"/>
      </row>
   </serviceQuantity>
</input>
<output type="group">
   <amount mapField="CALC_AMT">
      <row mapList="CALC_HDR">
         <HEADER_SEQ is="1"/>
      </row>
   </amount>
</output>
```

Figure 6: Weight Charge Schema

NOTE: Flatten. We use the terms "flattening" and "flatten out" to describe a business service property that resides in a collection but is defined as a singular element.

NOTE: Schema Definition Tips. A context sensitive "Schema Tips" zone appears when you open the *Business Service* page to assist you with the schema definition syntax. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

Inbound web services and *scripts* support interaction with business services. You can also invoke a business service from a Java class.

Service Program

This transaction defines services available in the system. These include user interface services as well as stand-alone services that perform a specific function. A service may be referenced by a business service. Use this transaction to view existing service and introduce a new stand-alone service to be made available to a Business Service.

Select **Admin** > **Service Program** > **Search** to maintain service programs.

Description of Page

Service Name is the unique identifier of the service.

CAUTION: Important! When adding new service programs, carefully consider its naming convention. Refer to *System Data Naming Convention* for more information.

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

Description describes the service.

Service Type indicates whether the service is a **Java Based Service** or a **Java (Converted) Service**.

This **Program Component** grid shows the list of program user interface components associated with the service. For a stand-alone service, this list is typically not applicable.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_MD_SVC*.

Defining Business Services

The topics in this section describe how to maintain business services.

Business Service - Main

Use this page to define basic information about a Business Service. Open this page using **Admin** > **Business Service** > **Search**

Description of Page

Enter a unique **Business Service** name and **Description**. Use the **Detailed Description** to describe the purpose of this business service in detail. **Owner** indicates if the business service is owned by the base package or by your implementation (**Customer Modification**).

> **CAUTION:** Important! If you introduce a new business service, carefully consider its naming convention. Refer to *System Data Naming Convention* for more information.

Enter the internal **Service Name** being called when this business service is invoked.

Enter the **Application Service** that is used to provide security for the business service. The application service must have an Access Mode of Execute.

Click the **View Schema** to view the business service's expanded schema definition. Doing this opens the *schema viewer* window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

Click the **View XML** hyperlink to view the XML document used to pass data to and from the service in the *Service XML Viewer*. You may find it useful to leave the application viewer window open while defining your business service schema.

> **NOTE: XML document may not be viewable.** If you create a new service program and do not regenerate the application viewer, you will not be able to view its XML document.

Where Used

Follow this link to open the data dictionary to view the tables that reference *F1_BUS_SVC*.

Business Service - Schema

Use this page to maintain a Business Service's schema and to see where the Business Service is used in the system. Open this page using **Admin** > **Business Service** > **Search** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the business service.

The *Schema Designer* zone allows you to edit the business service's schema. The purpose of the schema is to map the business service's elements to the corresponding fields of the back-end service program it rides on.

> **NOTE: Generating a Schema** A context sensitive "Generate Schema" zone is associated with this page. The zone provides a button that allows the user to generate a basic schema that includes all the elements that are found in the XML of the BS's service.

NOTE: Schema Definition Tips. A context sensitive "Schema Tips" zone is associated with this page. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and XAI Inbound Services. For each type of referencing entity, the *tree* displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

User Interface (UI) Maps

The User Interface (UI) map holds HTML to be rendered within *portal zones* and *Business Process Assistant (BPA) scripts*. UI maps allow your implementation to create input forms and output maps that closely match your customer's business practices. In other words, the UI Map is designed to facilitate the capture and display of your *business objects* and *business services*.

The UI map is a repository for a single HTML document paired with an XML schema where the schema defines the data that the HTML document displays and/or modifies. The UI Map HTML gives you the ability to craft the display by any method that an html document can support, including JavaScript and full CSS functionality.

Configuration tool support for UI Maps hinges around the ability to inject and extract an XML document from the HTML. For more information on the specialized support for HTML and JavaScript functionality - please refer to the HTML tips document (more on this below).

```
<html>
<head>
<title>Output Personal Information</title>
<link rel="stylesheet" type="text/css" href="cm_templates/cmStyles.css"/>
</head>
<body>

<table width="100%" border="0" cellpadding="2" style="margin-top:15px;" >

    <tr>
        <td/>
         <td/> <!-- locate the edit button on the bottom of the third column -->
        <td rowspan="99" style="vertical-align:bottom; margin-left:5px">
            <input type="button" value="edit" onClick="oraRunScript('HumanInfoU','personId');"/>
        </td>
    </tr>

    <tr>
        <td class="outputLabel">Name:</td>
        <td><span oraField="name" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Home Phone:</td>
        <td><span oraField="homePhone" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Business Phone:</td>
        <td><span oraField="businessPhone" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Cell Phone:</td>
        <td><span oraField="cellPhone" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">FAX:</td>
        <td><span oraField="fax" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Social Security:</td>
        <td><span oraField="socialSecurity" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Drivers License:</td>
        <td><span oraField="driversLicense" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Email:</td>
        <td><span oraField="email" class="outputData"></span></td>
    </tr>

</table>

</body>

<xml>
<root>
    <name>Greer,Johan</name>
    <email>jurgen.greer@media.com</email>
    <socialSecurity>939-30-3939</socialSecurity>
    <driversLicense>C8392020</driversLicense>
    <homePhone>(838) 030-0303</homePhone>
    <cellPhone>(444) 444-4040</cellPhone>
    <businessPhone>(737) 393-3838</businessPhone>
    <fax>(373) 939-3939</fax>
    <personId>1239997654</personId>
</root>
</xml>
</html>
```

Figure 7: 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**

edit

Figure 8: 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.

Figure 9: HTML Input Form Rendered (for BPA Script)

Portal zones can reference a UI map for the zone header. They may also utilize a UI map to define their filter area. This type of UI map is not a complete HTML document, but is instead configured as a UI Map "fragment".

NOTE: UI Map Tips. A context sensitive "UI Map Tips" zone appears when you open the *UI Map* page to assist you with the HTML document and schema definition syntax. The "tips" describe methods to take advantage of the configuration tools, however, they are not an HTML reference.

NOTE: Editing HTML. You can use a variety of HTML editors to compose your HTML, which you can then cut, and paste into your UI map. In addition, the zone provides a complete list of the XML schema nodes and attributes available to you when you construct the map's data schema.

Defining UI Maps

The topics in this section describe how to maintain UI Maps.

UI Map - Main

Use this page to define basic information about a user interface (UI) Map. Open this page using **Admin** > **UI Map** > **Search**

Description of Page

Enter a unique **Map** name. **Owner** indicates if the UI map is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new UI map, carefully consider its naming convention. Refer to *System Data Naming Convention* for more information.

Use **UI Map Type** to indicate whether the map is a **Complete HTML Document** or an **HTML Fragment**. Portal zones can reference a UI map to describe a fragment of their HTML, for example the zone header or filter area. In this case the UI map is not a complete HTML document, but is instead configured as a UI Map "fragment".

NOTE: A context sensitive "UI Map Tips" zone is associated with this page to assist you with the HTML document definition syntax. Refer to the tips zone for more information on how to use fragment UI maps to construct portal zone HTML components.

Enter a **Description**. Use the **Detailed Description** to describe how this map is used in detail.

Click on the **View Schema** to view the UI map's expanded schema definition. Doing this opens the *schema viewer* window.

Use the **Test UI Map** hyperlink to render your html in a test window.

NOTE: The **Test UI Map** hyperlink also exercises the proprietary functionality that binds an xml element with an html element so you can get immediate feedback on your html syntax.

Where Used

Follow this link to open the data dictionary to view the tables that reference *F1_MAP*.

UI Map - Schema

Use this page to maintain a UI Map's HTML and schema and to see where the UI Map is used in the system. Open this page using **Admin** > **UI Map** > **Search** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the UI Map.

The **HTML Editor** zone allows you to edit the HTML document of the map.

NOTE: HTML Definition Tips. A context sensitive "UI Map Tips" zone is associated with this page to assist you with the HTML document definition syntax. The "tips" describe good ways to produce simple HTML, however, they are not an HTML reference. Note that you can use a variety of HTML editors to compose your HTML, which you can then cut and paste into your UI map.

NOTE: Providing Help. A *tool tip* can be used to display additional help information to the user. This applies to section elements as well as individual elements on a map. Refer to the tips context sensitive zone for more information on how to enable and provide UI map help.

The ***Schema Designer*** zone allows you to edit the data schema part of the map. The purpose of the schema is to describe the data elements being displayed by the map.

NOTE: Schema Definition Tips. The same "UI Map Tips" zone above also provides a complete list of the XML nodes and attributes available to you when you construct the map's data schema.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and XAI Inbound 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 Hints

UI Hints are tags and elements that allow dynamic generation of formatted UI Maps. For example, UI Hints can be used as an alternative to manually creating maintenance and display UI Maps for business objects, thereby reducing the amount of time and technical expertise needed for creating those objects. UI Hints can also be added to the schemas on business services, service scripts, data areas, and UI maps. If you develop a BO using UI Hints, you should use them for the other configuration tool schemas for the BO, such as the data areas, UI map fragments, and display service scripts.

Additional information, including a list of working business objects that use UI Hints, is available in the context sensitive “Tips” zone on the schema pages for the above configuration tools.

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 is 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.
```

Maintaining Managed Content

The Managed Content maintenance object is used to store content such as XSL files used to create vector charts, JavaScript include files, and CSS files. These files may then be maintained in the same manner as the HTML in UI Maps.

The topics in this section describe the Managed Content portal.

Managed Content - Main

This page is used to define basic information about the content. Open this page using **Admin** > **Managed Content** > **Search** .

Description of Page

Enter a unique name for the content in the **Managed Content** field.

Owner indicates if the content is owned by the base package or by your implementation.

Use **Managed Content Type** to indicate the type of content, for example, XSLT or JavaScript.

Enter a **Description.**

Use the **Detailed Description** to describe in detail how this map is used.

Managed Content - Schema

This page is used to create and maintain the managed content. Open this page using **Admin** > **Managed Content** > **Search** and then navigate to the **Schema** tab.

Description of Page

The General Information zone displays the main attributes of the content. The Editor zone allows you to edit the content.

Data Areas

The data area has no business purpose other than to provide a common schema location for re-used schema structures. It exists solely to help eliminate redundant element declaration. For example, if you have multiple schemas that share a common structure, you can set up a stand-alone data area schema for the common elements and then include it in each of the other schemas.

Be aware that a stand-alone data area can hold elements that are mapped to true fields. For example, you might have 50 different types of field activities and all might share a common set of elements to identify where and when the activity should take place. It would be wise to declare the elements that are common for all in a stand-alone data area and then include it in the 50 field activity business objects.

It's strongly recommended that you take advantage of stand-alone data areas to avoid redundant data definition!

CAUTION: Dynamic inclusion! When the system renders a schema, all schemas included within it are expanded real-time. This means that any change you make to a data area will take effect immediately within all schemas it is referenced within.

NOTE: Schema Definition Tips. A context sensitive "Schema Tips" zone appears when you open the *Data Area* page to assist you with the schema definition syntax. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

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** > **Data Area** > **Search** .

Description of Page

Enter a unique **Data Area** name and **Description**. Use the **Detailed Description** to describe what this data area defines in detail. **Owner** indicates if the data area is owned by the base package or by your implementation (**Customer Modification**).

> **CAUTION:** Important! If you introduce a new data area, carefully consider its naming convention. Refer to *System Data Naming Convention* for more information.

Click the **View Schema** to view the data area's expanded schema definition. Doing this opens the *schema viewer* window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

To extend another data area, reference that data area in the **Extended Data Area** field. By extending a data area you can add additional elements to a base product data area.

Here's an example of an extended data area:

- The product releases with data area A, which contains elements a, b, and c.
- Your implementation creates data area CM-A, which contains element z, and references data area A as the extended data area.
- At run time, everywhere data area A is included it will contain elements a, b, c, and z.

Where Used

Follow this link to open the data dictionary to view the tables that reference *F1_DATA_AREA*.

Data Area - Schema

Use this page to maintain a Data Area's schema and to see where the data area is used in the system. Open this page using **Admin** > **Data Area** > **Search** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays the main attributes of the data area.

The ***Schema Designer*** zone allows you to edit the data area's schema. The purpose of the schema is to describe the structure and elements of the data area.

> **NOTE: Schema Definition Tips.** A context sensitive "Schema Tips" zone is associated with this page. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and XAI Inbound Services. For each type of referencing entity, the *tree* displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

Schema Designer

The Schema Designer is a Framework component that offers 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.
- Defining "flattened" fields.
- Adding custom nodes.
- Adding custom attribute values.

The designer provides the ability to toggle between **Design** and **Source** modes by clicking the icons on the **Schema Designer** title bar:

In the graphical **Design** mode, the zone is split into two areas: the left pane displays elements of the schema in a tree-like presentation, while the right pane displays all valid attributes and values for selected schema elements.

Context-sensitive embedded help is provided for fields and controls in the right pane by clicking the Help icon

You can also use links in the **Schema Tips** zone in the dashboard area to open a navigable Help window containing schema-related topics and tips. To search topic content, press `Ctrl+F` when the Schema Tips Help window has focus.

The **Schema Designer** is available by choosing the **Schema** tab on the *Business Object*, *Data Area*, *UI Map*, *Business Service*, and *Script* pages.

Schema Viewer

The schema viewer shows a tree-view presentation of a schema in its expanded form.

The schema illustrates the structure to be used when communicating with the schema's associated object. The following takes place when a schema is expanded:

- If the schema definition includes references to other schemas, these references are replaced with the corresponding schema definitions.
- Also, if the schema definition contains **private** elements, they are omitted from this view.

Clicking on any node on the tree populates the text box on the top with the node's absolute XPath expression. You will find this feature very useful when writing scripts interacting with schema-based objects. *Scripting* often involves referencing elements in a schema-based XML document using their absolute XPath expression. You can use this feature on the schema viewer to obtain the XPath expression for an element and copy it over to your script.

Business Event Log

Business Event Log may be viewed as a tool designed to capture any type of business event worth noting. You configure business objects to represent the various types of events your application calls for. The following type of details may be captured for each event:

- The business object representing the type of event.
- The date and time the event took place and who initiated it.
- The business entity for which this event is logged.

- Standard application message to describe the event.
- Additional context information that is available at the time of the event and varies for each type of event. The Business Event Log maintenance object supports a standard characteristics collection as well as an XML storage (CLOB) field. The event's business object determines where each piece of information resides. Refer to *Business Objects* for more information.

One common type of event may be the audit of changes made to sensitive data, for example, tracking an address change. Whenever an entity associated with a business object is added, changed, or deleted the system summarizes the list of changes that took place in that transaction and hands them over to **Audit** business object algorithms to process. You may design such an algorithm to audit the changes as business event logs. Refer to *a business object may define business rules* for more information.

You can also allow users to initiate business event logs to capture important notes about a business entity by exposing a *BPA Script* to invoke the event's corresponding business object.

Bottom line is that any process can create a business event log by invoking the business object representing the appropriate type of event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *F1_BUS_EVT_LOG*.

Chapter 7

To Do Lists

Certain events that occur within the system will trigger messages describing work that requires attention. For example, if a bill segment has an error, the system generates a To Do message to alert the person responsible for correcting such errors.

Each type of message represents a To Do list. For example, there are To Do lists for bill segment errors, payment errors, customer contact reminder, etc.

We refer to each message as a **To Do Entry**. Each To Do entry is assigned a specific **To Do Role**. The role defines the users who may work on the entry. A To Do entry has a **To Do log** that maintains record of the progress on the To Do entry. For example, the To Do log indicates when the To Do entry was created, when it was assigned to a user and to whom it was assigned, and when and by whom it was completed.

FASTPATH: Refer to *To Do Processing* for a description of end-user queries and tools assisting in reviewing, assigning and processing To Do entries.

The Big Picture of To Do Lists

The topics below provide more information about To Do configuration.

To Do Entries Reference A To Do Type

Every *To Do entry* references a To Do type. The To Do type controls the following functions:

- The To Do list on which the entry appears.
- The page into which a user is taken when they drill down on an entry.
- The message that appears in the user's To Do list. Note this message can be overridden for specific To Do messages by specifying a different message number in the process that creates the specific To Do entry. For example, the process that creates To Do entries associated with bill segments that are in error displays the error message rather than a generic "bill segment is in error" message.

- The To Do list's sort options. Sort options may be used on the To Do list page to sort the entries in a different order. For example, when a user looks at the bill segment error To Do list, they have the option of sorting it in error number order, account name order, or in customer class order. Note the default sort order is also defined on To Do type.
- Whether (and how) the To Do entry is downloaded to an external system (e.g., an email system).
- The roles to which an entry may be reassigned.
- The default priority of the To Do list in respect of other To Do lists.
- The To Do list's usage, which indicates whether a To Do of that type may be created manually by a user.
- The algorithms used to perform To Do list specific business rules.
- The characteristics applicable to the To Do list.

To Do Entries Reference A Role

Every *To Do entry* references a role. The role defines the users who may be assigned to **Open** entries.

The permissible roles that may be assigned to a To Do entry are defined on the entry's To Do type. After an entry is created, its role may be changed to any role defined as valid for the entry's To Do type.

An entry's initial role is assigned by the background process or algorithm that creates the entry. Because you can create your own processes and algorithms, there are an infinite number of ways to default an entry's role. However, the base package processes and algorithms use the following mechanisms to default an entry's role:

- The system checks if an entry's message category / number is suppressed (i.e., not created). If so, the entry is not created. Refer to *To Do Entries Can Be Rerouted Or Suppressed Based On Message Number* for more information.
- The system checks if an entry's message category / number is rerouted to a specific role. If so, it defaults this role. Refer to *To Do Entries Can Be Rerouted Or Suppressed Based On Message Number* for more information.
- Your specific product may introduce additional criteria for assigning a role, for example perhaps important accounts are assigned to a kind of account management group and the account management group includes configuration for special role for certain To Do types. Refer to *Set Additional Information Before a To Do is Created* for more information.
- If a Role wasn't determined in one of the previous steps and a Role is provided by the initiating process, the entry is created with that Role.
- If the entry does not have a role after the above takes place, the entry's To Do type's default role is assigned to the entry.

NOTE:
At installation time, the system provides a default role assigned to the system To Do types when first installed called **F1_DFLT**. This is done to allow testing of the system prior to implementing of appropriate To Do roles for your organization. The recommendation is to configure all the To Do Types with appropriate business oriented To Do roles once they are defined.

CAUTION: Important! Most organizations have the notion of a supervisor who is responsible for all entries assigned to a given role. It's important for this user (or users) to be part of all such roles. Refer to *To Do Supervisor Functions* for information about transactions that can be used by supervisors to review and assign work to users.

To Do Entries Can Be Rerouted (Or Suppressed) Based On Message Number

Consider the To Do type used to highlight bill segments that are in error. To Do entries of this type reference the specific bill segment error number so that the error message can be shown when the Bill Segments in Error To Do list is displayed.

NOTE: Message Category / Message Number. Every error in the system has a unique message category / number combination. Refer to *The Big Picture of System Messages* for more information about message categories and numbers.

If you want specific types of errors to be routed to specific users, you can indicate such on the To Do type. For example, if certain bill segment errors are always resolved by a given rate specialist, you can indicate such on the To Do type. You do this by updating the *To Do type's message overrides*. On this page you specify the message category / number of the error and indicate the To Do role of the user(s) who should work on such errors. Once the To Do type is updated, all new To Do entries of this type that reference the message number are routed to the desired role.

NOTE: Reroute versus suppression. Rather than reroute an entry to a specific role, you can indicate that an entry with a given message number should be suppressed (i.e., not created). You might want to do this if you have a large volume of certain types of errors and you don't want these to clutter your users' To Do lists.

Obviously, you would only reroute those To Do types that handle many different types of messages. In other words, if the To Do type already references a specific message category / number rerouting is not applicable.

We do not supply documentation of every possible message that can be handled by a given To Do type. The best way to build each To Do type's reroute list is during the pre-production period when you're testing the system. During this period, compile a list of the messages that should be routed to specific roles and add them to the To Do type.

Keep in mind that if a message number / category is not referenced on a To Do type's reroute information, the entry is routed as described under *To Do Entries Reference A Role*.

NOTE: Manually created To Do entries cannot be rerouted or suppressed. The rerouting occurs as part of the batch process or algorithm processing when the To Do is created. The role or user to whom a manual To Do should be assigned is specified when the To Do is created online. A manually created To Do may also be forwarded to another user or role.

The Priority Of A To Do Entry

Some To Do entries may be more urgent to resolve than others. A To Do entry is associated with a priority level representing its relative processing order compared to other entries.

Priority level is initially assigned as follows:

- If a **Calculate Priority** plug-in is defined on the To Do entry's type, the system calls it to determine the entry's priority. You may want to use this method if an entry's priority is based on context or time-based factors. For example, when priority takes into consideration account specific attributes. When applicable, you may design a process that triggers priority recalculation of To Do entries "at will". For example, when priority is reassessed periodically based on factors external to the To Do entry's information. Refer to *To Do Type* for more information on priority calculation algorithms.
- If a priority value has not been determined by a plug-in, the system defaults a To Do entry's initial priority to the value specified on its type.

A user may manually override a To Do entry's priority at any time. Notice that once a To Do entry's priority is overridden, **Calculate Priority** plug-ins are no longer called so as to not override the value explicitly set by the user.

NOTE: The system does not use priority values to control order of assignment nor processing of To Do entries. Priority is available to assist your organization with supporting a business practice that ensures higher priority issues are worked on first.

Working On A To Do Entry

A user can drill down on a To Do entry. When a user drills down on an entry, the user is transferred to the transaction associated with the entry. For example, if a user drills down on a bill segment error entry, the user is taken to the Bill Segment - Main page. Obviously, the page to which the user is taken differs depending on the type of entry.

It is also possible to configure the To Do type to launch a *script* when a user drills down on an entry rather than taking the user to a transaction. The script would walk the user through the steps required to resolve the To Do entry. Refer to *Launching Scripts When A To Do Is Selected* for more information.

After finishing work on an entry, the user can mark it as **Complete**. Completed entries do not appear on the To Do list queries (but they are retained on the database for audit purposes). If the user cannot resolve the problem, the user can forward the To Do to another user.

Launching Scripts When A To Do Is Selected

Users can complete many To Do entries without assistance. However, you can set up the system to launch a *script* when a user selects a To Do entry. For example, consider a To Do entry that highlights a bill that's in error due to an invalid mailing address. You can set up the system to execute a script when this To Do entry is selected by a user. This script might prompt the user to first correct the customer's default mailing address and then re-complete the bill.

A script is linked to a To Do type based on its message number using the *To Do type's message overrides*. Refer to *Executing A Script When A To Do Is Selected* for more information.

To Do Entries Have Logs

Each *To Do entry* has a To Do log that maintains a record of the To Do's progress in the system. For example, the To Do log indicates when the To Do entry was created, when it was assigned to a user and to whom it was assigned, and when and by whom it was completed. Users can view the log to see who assigned them a particular To Do and whether any work has already been done on the To Do.

A log entry is created for all actions that can be taken on a To Do entry. Log entries are created for the following events:

- A To Do entry is created (either by the system or by a user)
- A To Do entry is completed (either by the system or by a user)
- A user takes an open To Do entry
- A supervisor assigns a To Do entry
- A user forwards an entry to another user or role
- A user sends back a To Do to the user who forwarded it
- A user manually adds a log entry to record details about the To Do's progress
- A user manually overrides the To Do entry's priority

FASTPATH: For information about the contents of log entries for each of the events, refer to *Log Entry Events*.

How Are To Do Entries Created?

A To Do Entry may be created in the following ways:

- A *background process* can create To Do Entries.

- An *algorithm* can create entries of a given type. Because the use of algorithms is entirely dependent on how you configure the control tables, the number of types of such entries is indeterminate.
- A user can create entries of To Do types that have a **Manual** usage. Refer to *To Do Entries Created Manually* for information about setting up manual To Do types.

For any base product process that includes logic to create a To Do entry, the system supplies a sample To Do type that may be used. Although the To Do types provided by the product are system data, the following information related to each To Do type may be customized for an implementation and is not overwritten during an upgrade:

- The creation process. If the To Do is created by a background process where the background process is referenced on a To Do type. Refer to *To Do Entries Created By Background Processes* for more information.
- The routing process. Refer to *To Do Entries May Be Routed Out of the System* for more information.
- The priority. Refer to *To Do Type - Main* for more information.
- The *roles* that may be associated with the To Do type. Refer to *To Do Entries Reference a Role* for more information.
- The *message override* information. Refer to *To Do Entries Can Be Rerouted (Or Suppressed)* and *Launching Scripts When a To Do Is Selected* for more information.

To Do Entries Created By Background Processes

There are different types of To Do entries created by background processes:

- To Do entries created by dedicated To Do background processes
- To Do entries created for object-specific errors detected in certain background processes
- To Do entries created based on a specific condition

Dedicated To Do Background Processes

There are To Do entries that are created by system background processes whose main purpose is to create To Do entries based on a given condition. For these background processes, the To Do Type indicates the creation background process.

NOTE: If you don't schedule the background process, the entries will NOT be created! The To Do entries of this type will only be created if you have scheduled the associated background process. Therefore, if you want the system to produce a given entry, schedule the background process.

To Dos Created for Object-Specific Error Conditions

A system background process may create a To Do entry when an error is detected during object-specific processing and it is not possible to create an exception record. (I.e., either no exception record exists for the process or the error is not related to the entity reported in the exception record.)

For these background processes, the To Do Type must reference the creation background process. To have the system create To Do entries for some or all of the errors generated by one of these processes, you must do the following:

- If you want the system to generate To Do entries for errors detected by one of the background processes below, go to the appropriate To Do type and populate the creation background process.
- If you want the system to generate To Do entries for some errors for the process, but not for all errors, populate the creation background process and then proceed to the *message overrides* tab to *suppress* certain messages. To this by indicating the message category and message number you want to suppress. Any error that is suppressed is written to the *batch run tree*.

If you do not populate the creation background process, the errors are written to the *batch run tree*.

NOTE: Errors received while creating a To Do entry. If the background process cannot successfully create a To Do entry to report an object-specific error, the error is written to the batch run tree along with the error encountered while attempting to create the To Do entry.

NOTE: System errors are not included. To Do entries are not created for a system error, for example an error related to validation of input parameter. These errors are written to the *batch run tree*. Refer to *Processing Errors* for more information.

To Dos Created by Background Processes for Specific Conditions

There are some system background processes that create a To Do entry when the process detects a specific condition that a user should investigate. For each background process, the To Do type is an input parameter to the process. The system provides To Do types for each base package background process that may create a To Do entry.

NOTE: No Creation Process. These To Do types do not need (and should not have) a **Creation Process** specified.

To Do Entries Created By Algorithms

There are To Do entries that are created by algorithm types supplied with the base package. The system supplies a To Do Type for each of these To Do entries that you may use.

If you want to take advantage of these types of entries for system algorithm types, you must do the following:

- Create an *algorithm*:
  - This algorithm must reference the appropriate Algorithm Type.
  - These algorithms have a parameter of the To Do Type to be created. You should specify the To Do Type indicated in the table.
- Plug the algorithm into the respective control table.

To Do Entries Created Manually

You must set up manual To Do entry types if you want your users to be able to create To Do entries online. Users may create a manual To Do entry as a reminder to themselves to complete a task. Online To Do entries may also be used like electronic help tickets in the system. For example, if a user is having a problem starting service, the user can create a To Do that describes the problem. The To Do can be assigned to a help resolution group that could either resolve the problem or send the To Do back to the initiating user with information describing how to resolve the problem.

If you want to take advantage of manual To Do entries, create a To Do type and specify the following information.

On the Main tab:

- Set the **To Do Type Usage** flag to **Manual**.
- Set the **Navigation Option** to **toDoEntryMaint** (To Do entry maintenance).
- Set the **Message Category** and **Message Number** to the message you want to be used for To Do entries of this type. A generic base message is provided (category 15, message 1000) that can be used for manual To Dos. If you use this message, the To Do's subject appears as the message for the To Do.

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.

On the Drill Keys tab:

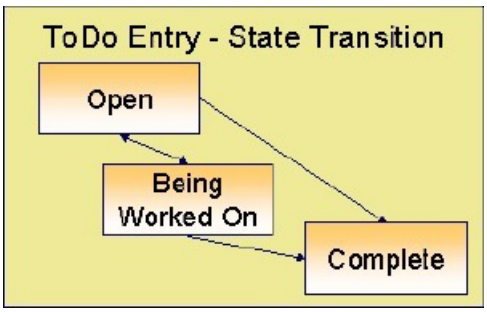
When a user adds a manual To Do entry, it is created with a drill key value equal to the To Do entry's ID. When the user clicks the Go To button next to the message in the To Do list, the system uses the drill down application service (defined on the main tab) and the drill key to display the associated To Do entry.

The To Do type should be set up with a drill key that reference the To Do entry table and the To Do entry ID:

Sequence	Table	Field
1	CI_TD_ENTRY	TD_ENTRY_ID

The Lifecycle Of A To Do Entry

The following state transition diagram will be useful in understanding the lifecycle of a To Do entry.



A To Do entry is typically created in the **Open** state. Entries of this type can be viewed by all users belonging to the entry's role. Refer to *How Are To Do Entries Created?* for information about how entries are created.

An **Open** entry becomes **Being Worked On** when it is assigned to a specific user or when a user proactively assumes responsibility for the entry. While an entry is **Being Worked On**, it is visible on the To Do Summary page only by the user who is assigned to it.

> **NOTE: To Do entries may be created in the Being Worked On state.** Some To Do background processes may create To Do entries in the **Being Worked On** state. When a user adds a To Do entry online and assigns the To Do to a user (as opposed to a role), the To Do entry is also created in the **Being Worked On** state.

A **Being Worked On** entry may be forwarded to a different user or role. If the entry is forwarded to a role, it becomes **Open** again.

When an entry becomes **Complete**, it is no longer visible in the To Do list queries (but it remains on the database for audit purposes). There are two ways an entry can become **Complete**:

- A user can manually indicate it is **Complete** (there are several ways to do this).

- For To Do entries that are logically associated with the state of some object, the system automatically marks the entry **Complete** when the object is no longer in the respective state. For example, an entry that's created when an account doesn't have a bill cycle is completed when the account has a bill cycle.

CAUTION: Important! The automatic completion of To Do entries occurs when the background process responsible for creating entries of a given type is executed. Therefore, if you only run these processes once per day, these entries remain **Being Worked On** even if the object is no longer in the respective state.

Linking Additional Information To A To Do Entry

Additional information may be linked to a To Do entry using characteristics. For example, when creating a manual To Do entry, a user may define the account related to the To Do.

When creating an automatic To Do entry, the program that generates the To Do may link related data to the To Do using characteristics. Use system *algorithm* to link related entities. For manually created To Dos, the valid characteristic types that may be linked to the To Do entry must be defined on the *To Do type* for that To Do entry.

If your To Do entries reference characteristics that are related to your global context data, you may want to configure an *Alerts* to display an alert if a related entry is **Open** or **Being Worked On**.

Implementing Additional To Do Entry Business Rules

If your business practice calls for additional validation rules or processing steps to take place after a To Do Entry is created or updated, you may want to take advantage of the **To Do Post Processing** plug-ins defined on *To Do type*.

For example, you may want to validate that To Do entries are only assigned to users with the proper skill levels needed to resolve them. Refer to *F1-VAL-SKILL* for a sample algorithm handling such validation.

To Do Entries May Be Routed Out Of The System

A To Do type can be configured so that its entries are interfaced to another system.

For example, a given To Do type can be configured to create an email message whenever a new To Do entry is created. The following points describe how to do this:

- Define the name of the background process responsible for interfacing the new To Do entries to another system on the respective To Do type. The base package contains a batch process called *F1-TDEER* that can be used for most situations. This batch process invokes the **External Routing** *algorithms* defined on each entry's To Do type.
- Plug in an appropriate **External Routing** algorithm on the respective To Do type. The logic in this type of algorithm performs the interface efforts for a specific To Do entry. For example, if an email message should be created for a To Do entry, the logic in the algorithm would compose and send the email message(s) for a specific To Do entry.

Click *here* to see the algorithm types available for this system event.

Periodically Purging To Do Entries

Completed To Do entries should be periodically purged from the system by executing the *F1–TDPG* background process. This background process offers you the following choices:

- You can purge all To Do entries older than a given number of days.
- You can purge To Do entries for a specific list of To Do types that are older than a given number of days.
- You can purge all To Do entries except for a specific list of To Do types that are older than a given number of days.

We want to stress that there is no system constraint as to the number of **Completed** To Do entries that may exist. You can retain these entries for as long as you desire. However, you will eventually end up with a very large number of **Completed** entries and these entries will cause the various To Do background processes to degrade over time. Therefore, you should periodically purge **Completed** To Do entries as they exist only to satisfy auditing and reporting needs.

NOTE: Different retention periods for different types of To Do entries. Keep in mind that the purge program allows you to retain different types of entries for different periods of time.

Setting Up To Do Options

The topics in this section describe how to set up To Do management options.

Installation Options

The following section describes configuration setup on the installation options.

To Do Information May Be Formatted By An Algorithm

A **To Do Information** algorithm may be plugged in on the *installation record* to format the standard To Do information that appears throughout the system. This algorithm may be further overridden by a corresponding plug-in on the *To Do Type*.

Set Additional Information Before A To Do Is Created

A **To Do Pre-creation** algorithm may be plugged in on the *installation record* to set additional information for a To Do entry before it is created. Algorithms of this type are used for two common purposes:

- Linking context specific data to the To Do entry using characteristics. For example, Oracle Utilities Customer Care and Billing provides an algorithm that attempts to link a related person, account, premise, service agreement or service point to a To Do entry based on its drill key value. Note, before you can set up this algorithm, you must define the characteristic types that you'll use to hold each of these entities. Also note that it is not necessary to define these characteristics as valid characteristic types on the To Do type.
- Overriding the Role of a To Do entry based on specific configuration related to the To Do's context data. For example, Oracle Utilities Customer Care and Billing provides an algorithm to determine a To Do role based on overrides related to the account's account management group or division.

Alerts

If your To Do entries reference characteristics related to your global context data and your product supports dashboard alerts generated by algorithms, you may want configure an algorithm to display an alert if the entry is **Open** or **Being Worked On** for the data currently in context.

Refer to your product's documentation to determine if these types of alerts are supported.

Next Assignment Algorithm

If your organization opts to use the next assignment feature supported by the Current To Do dashboard zone, you need to plug-in a **Next To Do Assignment** algorithm into the *installation options* to determine the next To Do entry the user should work on. Make sure you provide users with security access rights to the zone's next assignment action.

FASTPATH: Refer to the *Current To Do* zone for more information.

Messages

You need only set up new messages if you use algorithms to create To Do entries or prefer different messages than those associated with the base package's To Do types.

Feature Configuration

The base package is provided with a generic **Activity Queue Management** *Feature Configuration* type. You may want to set up a feature configuration of this type to define any To Do management related options supporting business rules specific to your organization.

For example, the base package provides the following plug-ins to demonstrate a business practice where To Do entries are only assigned to users with the proper skill levels to work on them.

- The base **To Do Post Processing** To Do Type algorithm *F1-VAL-SKILL* validates that a user has the necessary skill levels required to work on a given To Do entry.
- The base **Next To Do Assignment** installation options algorithm *F1-NEXT-ASSG* only assigns To Do entries to users that have the proper skills to work on them. This plug-in is only applicable if your organization practices *work distribution* "on demand."

You must set up such an **Activity Queue Management** feature configuration if you want to use any of the above base package plug-ins.

The following points describe the various **Option Types** provided with the base package:

- **Skill.** This option provides a reference to a skill category. For example, if you were using characteristics to represent skill categories then you should reference each characteristic type using this option.
- **Override Skill.** This option provides an override skill information reference for a specific message. For example, if you were using a To Do Type characteristic to specify an override skill category and level for a specific message category / number then you would reference this characteristic type using this option.

NOTE: Skill Setup. Refer to the description of the above base package algorithms for further details on how to setup skill level information.

NOTE: More Options. Your implementation may define additional options types. You do this by add new lookup values to the lookup field **F1QM_OPT_TYP_FLG**.

NOTE: Only one. The system expects only one **Activity Queue Management** feature configuration to be defined.

Defining To Do Roles

This section describes the control table used to maintain To Do roles.

To Do Role - Main

The **Main** page is used to define basic information about a To Do role.

To maintain this information, select **Admin** > **To Do Role** > **Search** .

Description of Page

Enter a unique **To Do Role** and **Description** for the To Do role.

The grid contains the ID of each **User** that may view and work on entries assigned to this role. The First Name and Last Name associated with the user is displayed adjacent.

> **NOTE: System Default Role**. The system supplies a default role **F1_DFLT** linked to each system To Do type. This is done so that To Do functionality may be tested prior to the creation of appropriate business oriented To Do roles.

Where Used

Follow this link to view the tables that reference *CI_ROLE* in the data dictionary schema viewer.

In addition, various “type” objects or algorithms may reference a To Do role to use when creating a To Do for a given business scenario. This is dependent on your specific product.

To Do Role - To Do Types

The **To Do Types** page defines the To Do types that may be viewed and worked on by users belonging to a given To Do role.

To maintain this information, select **Admin** > **To Do Role** > **Search** and navigate to the **To Do Types** page.

Description of Page

Enter the ID of each **To Do Type** whose entries may be viewed and worked on by the role.

Use As Default is a display-only field that indicates if the role is assigned to newly created entries of this type. You may define the default role for a given To Do type on the To Do Type maintenance page.

> **CAUTION:** If you remove a To Do type where this role is the default, you must define a new role as the default for the To Do type. You do this on the To Do Type maintenance page.

Defining To Do Types

This section describes the control table used to maintain To Do types.

To Do Type - Main

The **Main** page is used to define basic information about a To Do type.

FASTPATH: Refer to *The Big Picture Of To Do Lists* for more information about To Do types and To Do lists in general.

To maintain this information, select **Admin** > **To Do Type** > **Search** .

CAUTION: Important! If you introduce a To Do type, carefully consider its naming convention. Refer to *System Data Naming Convention* for more information.

Description of Page

Enter a unique **To Do Type** and **Description** for the To Do type.

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

Use the **Detailed Description** to provide further details related to the To Do Type.

Enter the default **Priority** of To Do entries of this type in respect of other To Do types. Refer to *The Priority Of A To Do Entry* for more information.

For **To Do Type Usage**, select **Automatic** if To Dos of this type are created by the system (i.e., a background process or algorithm). Select **Manual** if a user can create a To Do of this type online.

Define the **Navigation Option** for the page into which the user is transferred when drilling down on a To Do entry of this type.

Use **Creation Process** to define the background process, if any, that is used to manage (i.e., create and perhaps complete) entries of this type. A **Creation Process** need only be specified for those To Do types whose entries are created by a background process. Refer to *To Do Entries Created By Background Processes* for more information.

Use **Routing Process** to define the background process that is used to download entries of a given type to an external system, if any. A **Routing Process** need only be specified for those To Do types whose entries are routed to an external system (e.g., an Email system or an auto-dialer). Refer to *To Do Entries May Be Routed Out Of The System* for more information.

Use **Message Category** and **Message Number** to define the message associated with this To Do type's entries. Note: this message will only be used if the process that creates the To Do entry does not supply a specific message number. For example, the process that creates To Do entries that highlight bill segments that are in error would not use this message; rather, the entries are marked with the message associated with the bill segment's error.

Use the characteristics collection to define a **Characteristic Type** and **Characteristic Value** common to all To Do entries of this type. You may enter more than one characteristic row for the same characteristic type, each associated with a unique **Sequence** number. If not specified, the system defaults it to the next sequence number for the characteristic type.

Where Used

Follow this link to view the tables that reference *CI_TD_TYPE* in the data dictionary schema viewer.

To Do Type - Roles

The **Roles** page defines the roles who may view and work on entries of a given To Do type.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Roles** page.

Description of Page

Enter each **To Do Role** that may view and work on entries of a given type. Turn on **Use as Default** if the role should be assigned to newly created entries of this type. Only one role may be defined as the default per To Do type.

FASTPATH: Refer to *To Do Entries Reference A Role* for more information about roles and To Do entries.

To Do Type - Sort Keys

The **Sort Keys** page defines the various ways a To Do list's entries may be sorted. For example, when you look at the bill segment error To Do List, you have the option of sorting the entries in error number order, account name order, or in customer class order.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Sort Keys** page.

CAUTION: Do not change this information unless you are positive that the process / algorithm that creates entries of a given type stores this information on the entries.

Description of Page

The following fields display for each sort key.

Sequence is the unique ID of the sort key.

Description is the description of the sort key that appears on the To Do list.

Use as Default indicates the default sort key (the one that is initially used when a user opens a To Do list). Only one sort key may be defined as the default per To Do type.

Sort Order indicates whether the To Do entries should be sorted in **Ascending** or **Descending** order when this sort key is used.

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

To Do Type - Drill Keys

The **Drill Keys** page defines the keys passed to the application service (defined on the Main page) when you drill down on an entry of a given type.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Drill Keys** page.

CAUTION: Do not change this information unless you are positive that the process / algorithm that creates entries of a given type stores this information on the entries.

Description of Page

Navigation Option shows the page into which the user is transferred when drilling down on a To Do entry of this type.

The following fields display for each drill key.

Sequence is the unique ID of the drill key.

Table and **Field** are passed to the application service when you drill down on an entry of a given type.

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

To Do Type - Message Overrides

The **Message Overrides** page is used if you want To Do entries that reference a given message category / number to be routed to a specific To Do role (or suppressed altogether) or if you want to associate a script to a given message category / number.

FASTPATH: Refer to *To Do Entries Reference A Role* and *To Do Entries Can Be Rerouted Or Suppressed* for more information.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Message Overrides** page.

Description of Page

The following fields display for each override.

Message Category and **Number** allow the message to be overridden.

Exclude To Do Entry indicates if a To Do entry of this type that references the adjacent **Message Category** and **Number** should NOT be created.

Override Role indicates the to do role to which a To Do entry of this type that references the adjacent **Message Category** and **Number** should be addressed. This field is protected if **Exclude To Do Entry** is on.

Script indicates the script that should execute when a user drills down on a To Do entry of this type that references the adjacent **Message Category** and **Number**. This field is protected if **Exclude To Do Entry** is on. Refer to *Working On A To Do Entry* for more information.

To Do Type - To Do Characteristics

The **To Do Characteristics** page defines characteristics that can be defined for To Do entries of this type. The characteristic types for characteristics that are linked to the To Do entry as a result of a pre-creation algorithm do not need to be defined here.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **To Do Characteristics** page.

Turn on the **Required** switch if the **Characteristic Type** must be defined on To Do entries of this type.

Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

To Do Type - Algorithms

The **To Do Algorithms** page defines the algorithms that should be executed for a given To Do type.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Algorithms** page.

Description of Page

The grid contains **Algorithms** that control important To Do functions. If you haven't already done so, you must *set up the appropriate algorithms* in your system. You must define the following for each algorithm:

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

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

System Event	Optional / Required	Description
Calculate Priority	Optional	Algorithms of this type may be used to calculate a To Do entry's priority. They are called initially when a To Do entry is created and each time it gets updated so long as the To Do entry's priority has not been manually overridden. Once overridden, these algorithms are not called anymore. 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.

System Event	Optional / Required	Description
		If more than one algorithm is plugged-in the system calls them one by one until the first to return a calculated priority. Click *here* to see the algorithm types available for this system event.
External Routing	Optional	Algorithms of this type may be used to route a To Do entry to an external system. 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. 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. Click *here* to see the algorithm types available for this system event.
To Do Information	Optional	We use the term "To Do information" to describe the basic information that appears throughout the system to describe a To Do entry. The data that appears in "To Do information" is constructed using this algorithm. 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. Click *here* to see the algorithm types available for this system event.
To Do Post-Processing	Optional	Algorithms of this type may be used to validate and/or further process a To Do entry that has been added or updated. Click *here* to see the algorithm types available for this system event.

List of System To Do Types

The To Do types available to use with the product are found in the To Do Type page. In addition, they may be viewed in the *application viewer*'s *To Do type* viewer. If your implementation adds To Do types, you may *regenerate* the application viewer to see your additions reflected there.

Implementing The To Do Entries

To enable the To Do entries visible in the To Do Type page and application viewer, you must configure the system as follows:

- Define the To Do roles associated with each To Do type and link the appropriate users to them. Once you have defined the roles appropriate for your organization's To Do types, remove the reference to this system default role **F1_DFLT**. Refer to *To Do Entries Reference A Role* for more information.
- For any To Do Type that is provided for a specific background process, the To Do simply needs to reference the appropriate Creation Background Process. When the background process is scheduled, To Dos are created based on the logic of the related background process. This applies to *To Dos Created for Object-Specific Error Conditions* and *Dedicated To Do Background Processes*.

- For any To Do Type that is provided for creation by an algorithm or other process, there may be configuration required to populate that To Do type as an algorithm parameter or as an attribute on a control table.

  NOTE: Refer to the description of the To Do type for more information.

Chapter 8

Defining Background Processes

This section describes how to set up the background processes that perform many important functions throughout your product such as:

- Processing To Do Entries
- Processes that purge data
- Processes that archive data
- And many more...

There are also batch processes that will apply to your specific source application. Please refer to the documentation section that applies to your source application for more information.

The Big Picture of Background Processes

The topics in this section provide background information about a variety of background process issues.

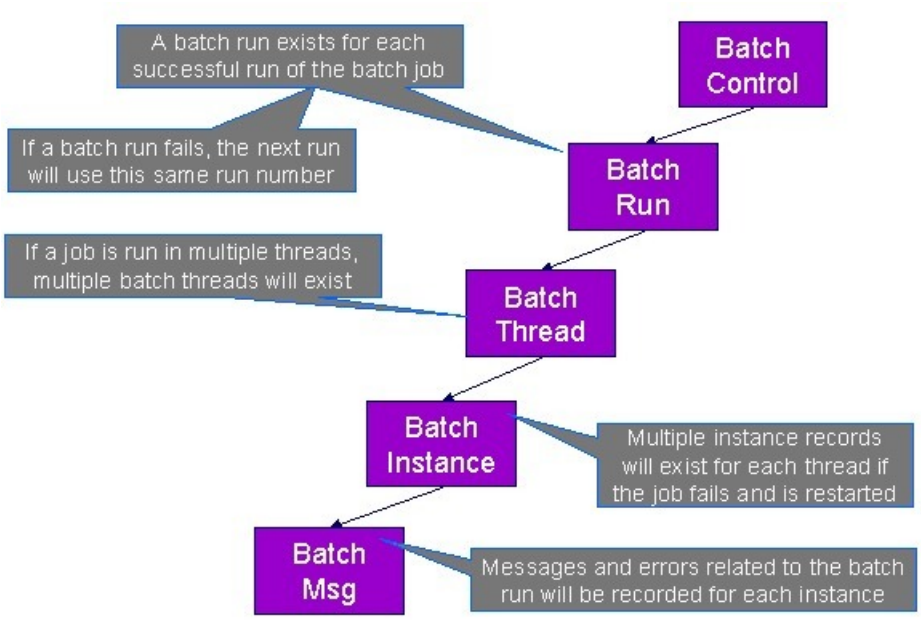
Background Processing Concepts

While the system uses a third-party party 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:
  - 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 extracts bill print information defines the next batch number to be assigned to recently completed bill routings. When this bill print extract process next runs, it extracts all bill routings 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 produces bills in Oracle Utilities Customer Care and Billing can be executed so that bills are produced in multiple "threads" (and multiple threads can execute at the same time). 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 and will be used when the *background process is submitted on-line*.

The following diagram illustrates the relationships that exist for batch control records.



Results of each batch run can be viewed using the *Batch Run Tree* page.

Parameters Supplied To Background Processes

All background processes receive the following parameters.

- **Batch code.** Batch code is the unique identifier of the background process.
- **Batch thread number.** Thread number is only used for background processes that can be run in multiple parallel threads. It contains the relative thread number of the process. For example, if the billing process has been set up to run in 20 parallel threads, each of the 20 instances receives its relative thread number (1 through 20). Refer to *Optimal Thread Count for Parallel Background Processes* for more information.
- **Batch thread count.** Thread count is only used for background processes that can be run in multiple parallel threads. It contains the total number of parallel threads that have been scheduled. For example, if the billing process has been set up to run in 20 parallel threads, each of the 20 instances receives a thread count of 20. Refer to *Optimal Thread Count for Parallel Background Processes* for more information.
- **Batch rerun number.** Rerun number is only used for background processes that download information that belongs to given run number. It should only be supplied if you need to download an historical run (rather than the latest run).
- **Batch business date.** Business date is only used for background processes that use the current date in their processing. For example, billing using the business date to determine which bill cycles should be downloaded. If this parameter is left blank, the system date is used. If supplied, this date must be in the format YYYY-MM-DD. Note: this parameter is only used during QA to test how processes behave over time.

- **Override maximum records between commits.** This parameter is optional and overrides each background process's Standard Commit. You would reduce this value, for example, if you were submitting a job during the day and you wanted more frequent commits to release held resources. You might want to increase this value when a background process is executed at night (or weekends) and you have a lot of memory on your servers.
- **Override maximum minutes between cursor re-initiation**. This parameter is optional and overrides each background process's Standard Cursor Re-Initiation Minutes. You would reduce these values, for example, if you were submitting a job during the day and you wanted more frequent commits to release held resources (or more frequent cursor initiations). You might want to increase these values when a background process is executed at night (or weekends) and you have a lot of memory on your servers.

  NOTE: The maximum minutes between cursor re-initiation is relevant for Oracle implementations only. Most of the system background processes contain an outermost loop / cursor. The cursor is opened at the beginning of the process and closed at the end. If Oracle feels that the cursor is open for too long, it may incorrectly interpret this as a problem and my issue an error that the snapshot is too old. The commit processing for the background processes is designed to refresh the cursor based on the minutes between cursor re-initiation in order to prevent this error.

- **User ID.** Please be aware of the following in respect of user ID:
  - The user ID is a user who should have access to all application services in the system. This is because some batch processes update records via services that may be check security.
  - Any batch process that stamps a user ID on a record it creates or updates uses this user ID.
  - This user ID's *display profile* controls how dates and currency values are formatted in messages.
- **Password.** Password is not currently used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace program at start** (Y/N), **trace program exit** (Y/N), **trace SQL** (Y/N) and **output trace** (Y/N). These switches are only used during QA and benchmarking. If trace program start is set to Y, a message is displayed whenever a program is started. If trace program at exist is set to Y, a message is displayed whenever a program is exited. If trace SQL is set to Y, a message is displayed whenever an SQL statement is executed. If output trace is set to Y, special messages formatted by the background process are written.

  NOTE: The information displayed when the output trace switch is turned on depends on each background process. It is possible that a background process displays no special information for this switch.

Override Maximum Errors in Batch Process Parameter

Each of the batch processes has, as part of its run parameters, a preset constant that determines how many errors that batch process may encounter before it is required to abort the run. A user can override this constant with an optional additional parameter (MAX-ERRORS). If a user chooses not to enter a value for the parameter, the process uses its own preset constant.

The input value must be an integer that is greater than or equal to zero. The maximum valid value for this parameter is 999,999,999,999,999.

The syntax for entering this additional parameter when submitting the batch process is "MAX-ERRORS=PARM VALUE", where the PARM VALUE is the desired value (e.g., **MAX-ERRORS=50**).

Extra Parameters

Some background processes receive additional parameters that are specific to their functionality. When a process receives additional parameters, they are defined and documented in the batch control entry in the application. They are also visible in the *batch control* viewer (part of the *application viewer*).

The syntax for entering these parameters when submitting the batch process is "PARM-NAME=PARM VALUE", where PARM-NAME is the name of the parameter as indicated in the batch control record. For example, if the batch control includes the **ADD-WORK-DAYS**, with possible values of **Y** and **N**, and you want to pass a value of **N**, enter the following when prompted: **ADD-WORK-DAYS=N**.

Indicating a File Path

Some of the system background processes use extra parameters to indicate a File Path and/or File Name for an input file or an output file. For example, most extract processes use File Path and File Name parameter to indicate where to place the output file.

When supplying a FILE-PATH variable, the directory specified in the FILE-PATH must already exist and must grant write access to the administrator account for the product. You may need to verify a proper location with your systems administrator.

The syntax of the FILE-PATH depends on the platform used for the product application server. Contact your system administrator for verification. For example, if the platform is UNIX, use forward slashes and be sure to put a trailing slash, for example **/spltemp/filepath/**.

Processing Errors

When a background process detects an error, the error may or may not be related to a specific object that is being processed. For example, if the program finds an error during batch parameter validation, this error is not object-specific. However, if the program finds an error while processing a specific bill, this error is object-specific. The system reports errors in one of the following ways:

- Errors that are not object-specific are written to the error message log in the *Batch Run Tree*.
- Some batch processes create entries in an "exception table" for certain object-specific errors. For example, an error detected in the creation of a bill in Oracle Utilities Customer Care and Billing may be written to the bill exception table. If an error is written to an exception table, it does not appear in the batch run tree. For each exception table, there is an associated to do entry process that creates a To Do Entry for each error to allow a user to correct the problem on-line.
- For some background processes, errors that do not result in the creation of an exception record may instead generate a To Do entry directly. For these processes, if you wish the system to directly create a To Do entry, you must configure the To Do type appropriately. Refer to *To Do entry for object-specific errors* for information about configuring the To Do type. If the background process detects an object specific error AND you have configured the system to create a To Do entry, the error is not written to the batch run tree. If you have configured your To Do type to not create To Do entries for certain errors, these errors are written to the *batch run tree*.

> **NOTE: Some processes create exceptions and To Do entries**. It is possible for a background process to create entries in an exception table AND create To Do entries directly, depending on the error. Consider batch billing in Oracle Utilities Customer Care and Billing; any conditions that cause a bill or bill segment to be created in **error** status result in a record added to the bill exception table or the bill segment exception table. However, any object-specific error that is not related to a specific bill or bill segment or any error that prevents a bill or bill segment from being created may result in a To Do entry for the object-specific error.

System Background Processes

NOTE: List of system background processes. The list of background processes provided in the base product may be viewed in the *application viewer*'s *batch control* viewer. In addition if your implementation adds batch control records, you may *regenerate* the application viewer to see your additions reflected there.

Parallel Background Processes

Many processes have been designed to run in parallel in order to speed execution. For example, the billing process in Oracle Utilities Customer Care and Billing can be executed so that bills are produced in multiple "threads" (and multiple threads can execute at the same time).

FASTPATH: The documentation for each background process provided with the system indicates if it may be run in parallel. Open the help index and navigate to the index entry labeled **background processes / system processes** to find the list of system processes for an indication of which processes can be run in parallel.

By default the system distributes the data for the multiple threads using the primary key of the main table of the process. For example, if you are running BILLING in Oracle Utilities Customer Care and Billing with multiple threads, the batch process distributes the accounts across the multiple threads. The Account ID is 10 digits long so if you run it with 4 threads, the records are processed as follows:

- Thread 1: Account IDs 0000000001 - 2500000000
- Thread 2: Account IDs 2500000001 - 5000000000
- Thread 3: Account IDs 5000000001 - 7500000000
- Thread 4: Account Ids 7500000001 - 9999999999

Note that the multi-threading logic relies on the fact that primary keys for master and transaction data are typically system generated random keys.

NOTE: Overriding the thread ranges. Your implementation has the ability to override the thread ranges if certain data in your system takes longer to process. For example, imagine you have a single account in Oracle Utilities Customer Care and Billing that has thousands of service agreements (maybe the account for a large corporation or a major city). You may want to set up the thread ranges to put this large account into its own thread and distribute the other accounts to the other threads. To do this, you should create the appropriate batch thread records ahead of time in a status of **Thread Ready** (**50**) with the key ranges pre-populated. Note that the base product does not provide the ability to add batch thread records online. If you are interested in more information about this technique, contact Customer Support.

Optimal Thread Count

Running a background process in multiple threads is almost always faster than running it in a single thread. The trick is determining the number of threads that is optimal for each process.

NOTE: A good rule of thumb is to have one thread for every 100 MHz of application server CPU available. For example if you have four 450 MHz processors available on your application server, you can start with 18 threads to begin your testing: (450 * 4) / 100 = 18.

This is a rule of thumb because each process is different and is dependent on the data in your database. Also, your hardware configuration (i.e., number of processors, speed of your disk drives, speed of the network between the database server and

the application server) has an impact on the optimal number of threads. Please follow these guidelines to determine the optimal number of threads for each background process:

- Execute the background process using the number of threads dictated by the rule of thumb (described above). During this execution, monitor the utilization percentage of your application server, database server and network traffic.
- If you find that your database server has hit 100% utilization, but your application server hasn't one of the following is probably occurring:
  - There may be a problematic SQL statement executing during the process. You must capture a database trace to identify the problem SQL.
  - It is also possible that your commit frequency may be too large. Commit frequency is a parameter supplied to every background process. If it is too large, the database's hold queues can start swapping. Refer to *Parameters Supplied to Background Processes* for more information about this parameter.
- It is normal if you find that your application server has hit 100% utilization but your database server has not. This is normal because, in general, all processes are CPU bound and not IO bound. At this point, you should decrease the number of threads until just under 100% of the application server utilization is achieved. And this will be the optimal number of threads required for this background process.
- If you find that your application server has NOT hit 100% utilization, you should increase the number of threads until you achieve just under 100% utilization on the application server. And remember, the application server should achieve 100% utilization before the database server reaches 100% utilization. If this proves not to be true, something is probably wrong with an SQL statement and you must capture an SQL trace to determine the culprit.

Another way to achieve similar results is to start out with a small number of threads and increase the number of threads until you have maximized throughput. The definition of "throughput" may differ for each process but can be generalized as a simple count of the records processed in the batch run tree. For example, in the Billing background process in Oracle Utilities Customer Care and Billing, throughput is the number of bills processed per minute. If you opt to use this method, we recommend you graph a curve of throughput vs. number of threads. The graph should display a curve that is steep at first but then flattens as more threads are added. Eventually adding more threads will cause the throughput to decline. Through this type of analysis you can determine the optimum number of threads to execute for any given process.

How to Re-extract Information

If you need to recreate the records associated with an historical execution of an extract process, you can - simply supply the desired batch number when you request the batch process.

How to Submit Batch Jobs

Most batch jobs are submitted via a batch scheduler.

In addition, you can manually submit your adhoc background processes or submit a special run for one of your scheduled background processes using the online *batch job submission* page.

How to Track Batch Jobs

You can track batch jobs using the batch process pages, which show the execution status of batch processes. For a specified batch control id and run id, the tree shows each thread, the run-instances of each thread, and any messages (informational, warnings, and errors) that might have occurred during the run.

FASTPATH: For more information, refer to *Tracking Batch Processes*.

How to Restart Failed Jobs and Processes

Every process in the system can be easily restarted if it fails (after fixing the cause of the failure). All you have to do is resubmit the failed job; the system handles the restart.

CAUTION: If a batch process terminated abnormally, you must run the batch process **UPDERR**- Change Batch Process's Status to Error before you resubmit the job. Refer to *Dealing with Abnormally Terminated Batch Processes* for more information.

Defining Batch Controls

The system is delivered with all necessary batch controls. If you introduce a new background process, open **Admin** > **Batch Control** > **Add** to define the related batch control 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.

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.

If the batch process is Timed, then the following fields are available:

- **Timer Interval** is the number of seconds between batch process submissions.
- **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 for development or testing purposes.
- **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 indicates the operating condition of the batch control.

- **Disabled** indicates that the batch level of service algorithm is not enabled for the batch control.

- **Error** indicates that errors were detected in the last batch process, or that the amount of time between batch processes exceeded a specified period.
- **Normal** indicates that no errors were detected in the last batch process.

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 in *System Background Processes*.

Description. A description of the parameter.

Detailed Description. A more detailed description of the parameter.

Required. Indicate whether or not this is a required parameter.

Parameter Value. Enter a default value, if applicable.

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** > **Batch Control** > **Search** and then navigate to the **Algorithms** tab.

Description of Page

The **Algorithms** grid contains algorithms that control important functions for instances of this batch control. You must define the following for each algorithm:

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

The following table describes the **System Events**.

System Event	Optional / Required	Description
Batch Control — Level of Service	Optional	Algorithms of this type are called to determine the Level of Service provided by a batch control by analyzing the batch run tree. Click *here* to see the algorithm types available for this system event.

The Big Picture of Requests

Requests enable an implementer to design an ad-hoc batch process using the configuration tools.

An example of such a process might be to send an email to a group of users that summarizes the To Do entries that are assigned to them. This is just one example. The request enables many types of diverse processing.

Request Type Defines Parameters

For each type of process that your implementation wants, you must configure a request type to capture the appropriate parameters needed by the process.

Previewing and Submitting a Request

To submit a new request, go to **Menu** > **Batch** > **Request** > **Add** . You must select the appropriate request type and then enter the desired parameter values, if applicable.

After entering the parameters, the following actions are possible:

- Click **Save** to submit the request.
- Click **Cancel** to cancel the request.
- Click **Preview** to see a sample of records that satisfy the selection criteria for this request. This information is displayed in a separate map. In addition, the map displays the total number of records that will be processed when the request is submitted. From this map you can click **Save** to submit the request, **Back** to adjust the parameters, or **Cancel** to cancel the request.

When a request is saved, the job is not immediately submitted for real time processing. The record is saved with the status **Pending** and a monitor process for this record's business object is responsible for transitioning the record to **Complete**.

As long as the record is still **Pending**, it may be edited to adjust the parameters. The preview logic described above may be repeated when editing a record.

The actual work of the request, such as generating an email, is performed when transitioning to **Complete** (using an enter processing algorithm for the business object).

To Do Summary Email

The base product includes a sample request process that sends an email to users that have incomplete To Dos older than a specified number of days.

The following configuration tasks are required to use this process:

- Define an Outbound Message Type. The business object usually defined for the Outbound Message Type is **F1-EmailMessage.**

- Define an External System that contains the Outbound Message Type in one of its steps. In the configuration determine if the communication is through SOA, batch, or real-time processing method when sending the email notification. Refer to *Outbound Messages* for more information about configuration needed for the different processing methods.
- Create a Request Type that includes the Outbound Message Type and the corresponding External System.
- Create a Request for the created Request Type.

Exploring Request Data Relationships

Use the following links to open the application viewer where you can explore the physical tables and data relationships behind the request functionality:

- Click *F1-REQ-TYPE* to view the request type maintenance object's tables.
- Click *F1-REQ* to view the request maintenance object's tables.

Defining a New Request

To design a new ad-hoc batch job that users can submit via Request, first create a new Request Type business object. The base product BO for request type, **F1-TodoSumEmailTyp**, may be used as a sample.

The business object for the request includes the functionality for selecting the records to process, displaying a preview map for the user to review, and for performing the actual processing. The base product BO for request, **F1-TodoSumEmailReq**, may be used as a sample. The following points highlight the important configuration details for this business object:

- Special BO options are available for request BOs to support the Preview Request functionality.
  - **Request Preview Service Script**. This script retrieves the information that is displayed when a user asks for a preview of a request.
  - **Request Preview Map.** This map displays the preview of a request.
- The enter algorithm plugged into the Complete state is responsible for selecting the records that satisfy the criteria and processing the records accordingly.

Setting Up Request Types

Use the Request Type portal to define the parameters to capture when submitting a request. Open this page using **Admin** > **Request Type** > **Add** .

This topic describes the base-package zones that appear on the Request Type portal.

Request Type List. The Request Type List zone lists every request type. The following functions are available:

- Click a **broadcast** icon to open other zones that contain more information about the request type.
- Click **Add** in the zone's title bar to add a new request type.

Request Type. The Request Type zone contains display-only information about a request type. This zone appears when a request type has been broadcast from the Request Type List zone or if this portal is opened via a drill down from another page. The following functions are available:

- Click **Edit** to start a business process that updates the request type.
- Click **Delete** to start a business process that deletes the request type.
- Click **Deactivate** start a business process that deactivates the request type.
- Click **Duplicate** to start a business process that duplicates the request type.

- State transition buttons are available to transition the request type to an appropriate next state.

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

Maintaining Requests

Use the Request transaction to view and maintain pending or historic requests.

Open this page using **Menu** > **Batch** > **Request** > **Search** . This topic describes the base-package portals and zones on this page.

Request Query. Use the query portal to search for an existing request. Once a request is selected, you are brought to the maintenance portal to view and maintain the selected record.

Request Portal. This portal appears when a request has been selected from the Request Query portal. The following base-package zones appear on this portal:

- **Actions.** This is a standard actions zone.
- **Request.** The Request zone contains display-only information about a request. Please see the zone's help text for information about this zone's fields.
- **Request Log.** This is a standard log zone.

Chapter 9

Defining Algorithms

In this section, we describe how to set up the user-defined algorithms that perform many important functions including:

- Validating the format of a phone number entered by a user.
- Validating the format of a latitude/longitude geographic code entered by a user.
- In products that support payment and billing:
  - Calculating late payment charges.
  - Calculating the recommended deposit amount.
  - Constructing your GL account during the interface of financial transactions to your GL
- And many other functions...

The Big Picture Of Algorithms

Many functions in the system are performed using a user-defined algorithm. For example, when a CSR requests a customer's recommended deposit amount, the system calls the deposit recommendation algorithm. This algorithm calculates the recommended deposit amount and returns it to the caller.

NOTE: Algorithm = Plug-in. We use the terms plug-in and algorithm interchangeably throughout this documentation.

So how does the system know which algorithm to call? When you set up the system's control tables, you define which algorithm to use for each component-driven function. You do this on the control table that governs each respective function. For example:

- You define the algorithm used to validate a phone number on your phone types.
- You define the algorithm in Oracle Utilities Customer Care and Billing used to calculate late payment charges on each Service Agreement Type that has late payment charges.
- The list goes on...

The topics in this section provide background information about a variety of algorithm issues.

Algorithm Type Versus Algorithm

You have to differentiate between the type of algorithm and the algorithm itself.

- An **Algorithm Type** defines the program that is called when an algorithm of this type is executed. It also defines the types of parameters that must be supplied to algorithms of this type.
- An **Algorithm** references an **Algorithm Type**. It also defines the value of each parameter. It is the algorithm that is referenced on the various control tables.

FASTPATH: Refer to *How to Add A New Algorithm* for an example that will further clarify the difference between an algorithm and an algorithm type.

How To Add A New Algorithm

Before you can add a new algorithm, you must determine if you can use one of the sample algorithm types supplied with the system. Refer to *List of Algorithm Types* for a complete list of algorithm types.

If you can use one of the sample algorithm types, simply add the algorithm and then reference it on the respective control table. Refer to *Setting Up Algorithms* for how to do this.

If you have to add a new algorithm type, you may have to involve a programmer. Let's use an example to help clarify what you can do versus what a programmer must do. Assume that you require an additional geographic type validation algorithm. To create your own algorithm type you must:

- Write a new program to validate geographic type in the appropriate manner. Alternatively, you may configure a plug-in script to implement the validation rules. The advantage of the latter is that it does not require programming. Refer to *plug-in script* for more information.
- Create an Algorithm Type called **Our Geographic Format** (or something appropriate). On this algorithm type, you'd define the name of the program (or the plug-in script) that performs the function. You'd also define the various parameters required by this type of algorithm.
- After creating the new Algorithm Type, you can reference it on an Algorithm.
- And finally, you'd reference the new Algorithm on the Geographic Type that requires this validation.

Minimizing The Impact Of Future Upgrades

The system has been designed to use algorithms so an implementation can introduce their own logic in a way that's 100% upgradeable (without the need to retrofit logic). The following points describe strong recommendations about how to construct new algorithm type programs so that you won't have to make program changes during future upgrades:

- Do not alter an algorithm type's hard parameters. For example, you might be tempted to redefine or initialize parameters defined in an algorithm type's linkage section. Do not do this.
- Follow the naming conventions for the new algorithm type code and your source code, i.e., both the source code and the algorithm type should be prefixed with "CM". The reason for this naming convention is to make it impossible for a new, base-package algorithm type from overwriting your source code or algorithm type meta-data (we will never develop a program or introduce meta-data beginning with CM).
- Avoid using inline SQL to perform insert/update/delete. Rather, invoke the base-package's object routines or common routines.
- Avoid using base messages (outside of common messages, i.e., those with a message number < 1000) as we may deprecate or change these messages in future releases. The most common problem is caused when an implementation

clones a base package algorithm type program because they need to change a few lines of logic. Technically, to be 100% upgradeable, you should add new messages in the "90000" or greater category (i.e., the category reserved for implementation-specific messages) for every message in your new program even though these messages may be duplicates of those in the base package.

Setting Up Algorithm Types

The system provides many algorithm types to support base product functionality. If you need to introduce a new type of algorithm, open **Admin** > **Algorithm Type** > **Add** .

FASTPATH: Refer to *The Big Picture Of Algorithms* for more information.

CAUTION: Important! If you introduce a new algorithm type, carefully consider its naming convention. Refer to *System Data Naming Convention* for more information.

Description of Page

Enter an easily recognizable **Algorithm Type** and **Description**.

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

Enter a **Detailed Description** that describes, in detail, what algorithms of this type do.

Use **Algorithm Entity** to define where algorithms of this type can be "plugged in". If a detailed description about an algorithm entity is available, a small help icon is visible adjacent to the dropdown. Click the icon to view the information.

NOTE: The values for this field are customizable using the *lookup* table. This field name is **ALG_ENTITY_FLG**.

Use **Program Type** to define if the algorithm's program is written using **Java**, a **Plug-In Script**, or **Java (Converted)**, meaning the program has been converted to Java.

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

Use **Program Name** to define the program to be invoked when algorithms of this type are executed:

- If the Program Type is **Java (Converted)**, enter the name of the converted program.
- If the Program Type is **Java**, enter the Java class name.
- If the Program Type is **Plug-In Script**, enter the plug-in *script* name. Only plug-in scripts defined for the algorithm entity may be used.

NOTE: **View the source.** If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the *Java docs viewer*. For plug-in scripts, drill into the plug-in script to view the details.

Use the **Parameter Types** grid to define the types of parameters that algorithms of this type use. The following fields should be defined for each parameter:

- Use **Sequence** to define the relative position of the **Parameter**.
- Use **Parameter** to describe the verbiage that appears adjacent to the parameter on the Algorithm page.
- Indicate whether the parameter is **Required**. This indicator is used when parameters are defined on algorithms that reference this algorithm type.

- **Owner** indicates if the parameter for this algorithm type is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an algorithm type with parameters. This information is display-only.

  NOTE: When adding a new algorithm type that is for a Java program, the parameters are automatically generated based on the Java code. Once an algorithm type exists, any additional parameters defined in the Java code should be manually added to the algorithm type. For other program types, algorithm type parameters must be manually defined.

  NOTE: When a new algorithm type parameter is added for any program type, existing algorithms for the algorithm type do not automatically get updated with the new parameter. The algorithms must be manually updated.

Where Used

An Algorithm references an Algorithm Type. Refer to *Setting Up Algorithms* for more information.

List of Algorithm Types

The algorithm types available to use with the product may be viewed in the algorithm type page and in the *application viewer*'s *algorithm* viewer. If your implementation adds algorithm types or adds algorithms to reference existing algorithm types, you may *regenerate* the application viewer to see your additions reflected there.

Setting Up Algorithms

If you need to introduce a new algorithm, open **Admin** > **Algorithm** > **Add** . Refer to *The Big Picture Of Algorithms* for more information.

Description of Page

Enter an easily recognizable **Algorithm Code** and **Description** of the algorithm. **Owner** indicates if this algorithm is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new algorithm, carefully consider its naming convention. Refer to *System Data Naming Convention* for more information.

Reference the **Algorithm Type** associated with this algorithm.

FASTPATH: Refer to *Algorithm Type Versus Algorithm* for more information about how an algorithm type controls the type of parameters associated with an algorithm.

The parameters available for an algorithm are defined on the algorithm type. The system allows a set of parameter values to change over time. Use the parameter scroll to view parameter values for a given **Effective Date**. The **Owner** of the collection of parameters is displayed. The collection shows the **Parameter** description, the **Sequence** and the **Value** for each parameter.

NOTE: If the product delivers an algorithm with parameter values defined, an implementation may override the base provided parameter values by adding an additional effective dated collection of parameters.

NOTE: If an algorithm is defined and subsequently a new parameter is added to the algorithm type, existing algorithms for the algorithm type should be updated as follows: Click the “+” to add a new effective dated entry to the parameter collection. At this point the latest list of parameters for the algorithm type are visible. Configure the parameters accordingly.

Where Used

Algorithms are plugged in on control tables throughout the system. Each algorithm type's Algorithm Entity indicates the name of the control table where it is plugged in. The *algorithm* viewer in the application viewer may also be used to see a list of plug-in spots along with their algorithm types and algorithms.

Chapter 10

Defining Script Options

We use the term "script" to define processing rules that your implementation sets up to control both front-end and back-end processing:

- Rules that control front-end processing are defined using *Business Process Assistant* (BPA) scripts. For example, your implementation could set up a BPA script to guide a user through your organization's payment cancellation process.
- Rules that control back-end processing are defined using *Server-based* scripts. For example, your implementation could set up a server-based script to control the processing that executes whenever a given type of adjustment is canceled.

The topics in this section describe how to configure your scripts.

The Big Picture Of Scripts

This section describes features and functions that are shared by both BPA scripts and server-based scripts.

Scripts Are Business Process-Oriented

To create a script, you must analyze the steps used to implement a given business process. For example, you could create a "stop auto pay" BPA script that:

- Asks the user to select the customer / taxpayer using an appropriate search page
- Asks the user to define the date on which the person would like to stop making automatic payments
- Invokes a server-based script that populates the end-date on the account's latest automatic payment instructions.

After you understand the business process, you can set up a script to mimic these steps. If the business process is straightforward (e.g., users always perform the same steps), the script configuration will be straightforward. If the business process has several logic branches, the composition of the script may be more challenging.

A Script Is Composed Of Steps

A script contains one or more steps. For example, a "stop auto pay" BPA script might have three steps:

- Ask the user to select the customer / taxpayer using an appropriate search page
- Ask the customer the date on which they'd like to stop making automatic payments (and default the current date)
- Invoke a server-based script that, in turn, updates the account's auto pay options.

Each step references a step type. The step type controls what happens when a step executes. It might be helpful to think of a script as a macro and each step as a "line of code" in the macro. Each step's step type controls the function that is executed when the step is performed.

> **FASTPATH:** Refer to *How To Set Up Each Step Type* for a detailed description of all available step types and how to set them up.

Designing And Developing Scripts

Constructing a script is similar to writing a computer program. We recommend that you follow the approach outlined below when you construct scripts:

- Thoroughly understand the business process to be scripted
- Thoroughly understand how the transactions and services that your script uses work
- Design the steps for the script "on paper"
- Determine the most maintainable way to set up your scripts. Rather than creating complex, monolithic scripts, we recommend dividing scripts into smaller sections. For example:
  - For BPA scripts,
    - Determine if several scripts have similar steps. If so, set up a script that contains these common steps and invoke it from the main scripts. For example, if the main script were a BPA script, this would be invoked via a **Perform script** step.
    - Determine if a large script can be divided into logical sections. If so, set up a small script for each section and create a "master script" to invoke each. For example, if the main script were a BPA script, this would be invoked via a **Transfer control** step.
  - For server-based script, you can segregate reusable steps into "service scripts" and then use **Invoke service script** steps to execute the common logic.
- Add the script using *Script Maintenance*
- Thoroughly test the script

A Script May Declare Data Areas

Both BPA and server-based scripts may have one or more *data areas*:

- If the script contains steps that exchange XML documents, you must declare a data area for each type of XML document. For example, if a BPA script has a step that invokes a service script, the BPA script must declare a data area that holds the XML document that is used to pass information to and from the service script.
- You can use a data area as a more definitive way to declare your temporary storage. For example, you can describe your script's temporary storage variables using a *stand-alone data area* schema and associate it with your script.

Various step types involve referencing the script's data areas as well as support the ability to compare and move data to and from field elements residing in the data areas.

An *Edit Data* step supports the syntax to dynamically declare data areas as part of the step itself. This technique eliminates the need to statically declare a data area. Refer to *Designing Generic Scripts* for an example of when this technique may be useful.

Designing Generic Scripts

Scripts may be designed to encapsulate an overall procedure common across different business objects.

For example, BPA scripts may implement a standard procedure to maintain business entities in which the first step obtains the entity's data, the second step presents its associated UI map to the user for update, and the last step updates the entity. Notice that in this case the only difference from one object to another is the data to capture. Rather than designing a dedicated BPA script with static data areas and invocation steps for each business object, you can design a single generic script that dynamically invokes a business object and its associated UI map.

This functionality is available only within an *Edit Data* step. With this technique, the name of the schema-based object is held in a variable or an XPath schema location and is used to both declare and invoke the object.

NOTE: Tips. Refer to the tips context zone associated with the script maintenance page for more information on edit data commands and examples.

Securing Script Execution

The system supports the ability to secure the execution of scripts by associating the script with an Application Service. Refer to *The Big Picture of Application Security* for more information. Application security is optional and applies to service scripts and user-invocable BPA scripts only. If a script is not associated with an application service, all users may execute the script. Otherwise, only users that have **Execute** access to the application service may execute the script.

The Big Picture Of BPA Scripts

FASTPATH: Refer to *The Big Picture Of Scripts* to better understand the basic concept of scripts.

Users may require instructions in order to perform certain tasks. The business process assistant allows you to set up scripts that step a user through your business processes. For example, you might want to create scripts to help users do the following:

- Add a new person to an existing account
- Set up a customer to pay automatically
- Modify a customer who no longer wants to receive marketing information
- Modify a customer's web password
- Record a trouble order
- Merge two accounts into one account
- Fix a bill with an invalid rate
- ... (the list is only limited by your time and imagination)

Users execute these scripts via the *business process assistant* (BPA). Users can also define their favorite BPA scripts in their *user preferences*. By doing this, a user can execute a script by pressing an accelerator key (`Ctrl + Shift` + a number).

Don't think of these scripts as merely a training tool. BPA scripts can also reduce the time it takes to perform common tasks. For example, you can write a script that reduces the "number of clicks" required to add a new person to an existing account.

CAUTION: Future upgrade issues. Although we make every effort not to remove fields or tab pages between releases, there may be times when changes made by the base-package will necessitate changes to your scripts. Please refer to the release notes for a list of any removed fields or tab pages.

CAUTION: Scripts are not a substitute for end-user training. Scripts minimize the steps required to perform common tasks. Unusual problems (e.g., a missing meter exchange) may be difficult to script as there are many different ways to resolve such a problem. However, scripts can point a user in the right direction and reduce the need to memorize obscure business processes.

The topics in this section describe background topics relevant to BPA scripts.

How To Invoke Scripts

Refer to *Initiating Scripts* for a description of how end-users initiate scripts.

Developing and Debugging Your BPA Scripts

You may find it helpful to categorize the step types into two groups: those that involve some type of activity in the *script area*, and those that don't. The following step types cause activity in the script area: **Height**, **Display text**, **Prompt user**, **Input data**, **Input Map**, **Set focus to a field**. The rest of the step types are procedural and involve no user interaction. For debugging purposes, you can instruct the system to display text in the script area for the latter step types. Also note, for debugging purposes, you can display an entire data area (or a portion thereof) in the script area by entering **%+...+%** where **...** is the name of the node whose element(s) should be displayed.

NOTE: Time saver. When you develop a new BPA script, change your *user preferences* to include the script as your first "favorite". This way, you can press `Ctrl+Shift+1` to invoke the script (eliminating the need to select it from the *script menu*).

Launching A Script From A Menu

You can create menu items that launch BPA scripts rather than open a page. To do this, create a *navigation option* that references your script and then add a menu item that references the navigation option.

If the navigation option is referenced on a *context menu* and the navigation option has a "context field", a temporary storage variable will be created and populated with the unique identifier of the object in context. For example, if you add a "script" navigation option to the bill context menu and this navigation option has a context field of BILL_ID, the system will create a temporary storage variable called BILL_ID and populate it with the respective bill id when the menu item is selected.

Launching A Script When Starting The System

You can set the system to launch a script upon startup.

For example, imagine that through an interactive voice response system, a customer has keyed in their account ID and has indicated that they would like to stop an automatic payment. At the point when the IVR system determines that the customer must speak to a user, the interface can be configured to launch the application. When launched it passes the script name and account ID. It can also pass a *navigation option* to automatically load the appropriate page (if this information is not part of the script).

To do this, parameters are appended to the standard system URL. The following parameters may be supplied:

- script=<scriptname>
- ACCT_ID=<account id>
- location=<navigation option>

Parameters are added to the standard system URL by appending a question mark (?) to the end and then adding the "key=value" pair. If you require more than one parameter, use an ampersand (&) to separate each key=value pair.

For example, the following URLs are possible ways to launch the **StopAutoPay** script at startup, assuming your standard URL for launching the system is http://system-server:1234/cis.jsp:

- http://system-server:1234/cis.jsp?script=StopAutoPay
- http://system-server:1234/cis.jsp?script=StopAutoPay&ACCT_ID=1234512345
- http://system-server:1234/cis.jsp?script=StopAutoPay&ACCT_ID=1234512345&location=accountMaint

It doesn't matter in which order the parameters are provided. The system processes them in the correct order. For example, the following examples are processed by the system in the same way:

- http://system-server:1234/cis.jsp?ACCT_ID=1234512345&script=StopAutoPay&location=accountMaint
- http://system-server:1234/cis.jsp?ACCT_ID=1234512345&location=accountMaint&script=StopAutoPay

These parameters are kept in a common area accessible by any script for the duration of the session. To use these parameters on a script you may reference the corresponding **%PARM-<name>** *global variables*. In this example, after the system is launched any script may have access to the above account ID parameter value by means of the **%PARM-ACCT_ID** global variable. Also note, these parameters are also loaded into temporary storage (to continue the example, there'd also be a temporary storage variable called **ACCT_ID** that holds the passed value).

Executing A Script When A To Do Entry Is Selected

The system creates *To Do entries* to highlight tasks that require attention (e.g., records in error). Users can complete many of these tasks without assistance. However, you can set up the system to automatically launch a script when a user selects a To Do entry. For example, consider a To Do entry that highlights a bill that's in error due to an invalid mailing address. You can set up the system to execute a script when this To Do entry is selected by a user. This script might prompt the user to first correct the customer's default mailing address and then re-complete the bill.

The following points provide background information to help you understand how to implement this functionality:

- Every To Do entry references a *To Do type* and a *message category and number*. The To Do type defines the category of the task (e.g., bill errors). The message number defines the specific issue (e.g., a valid address can't be found.). Refer to *The Big Picture of System Messages* for more information about message categories and numbers.
- When a user drills down on a To Do entry, either a script launches OR the user is transferred to the transaction associated with the entry's To Do type. You control what happens by configuring the To Do type accordingly:
  - If you want to launch a script when a user drills down on an entry, you link the script to the *To Do type and message number*. Keep in mind that you can define a different script for every message (and some To Do types have many different messages).
  - If the system doesn't find a script for an entry's To Do type and message number, it transfers the user to the To Do type's default transaction.

NOTE: How do you find the message numbers? We do not supply documentation of every To Do type's message numbers (this list would be impossible to maintain and therefore untrustworthy). The best way to determine which message numbers warrant a script is during pre-production when you're testing the system. During this period, To Do entries will be generated. For those entries that warrant a script, simply display the entry on *To Do maintenance*. On this page, you will find the entry's message number adjacent to the description.

- These types of scripts invariably need to access data that resides on the selected To Do entry. Refer to *How To Use To Do Fields* for the details.

The Big Picture Of Script Eligibility Rules

You can configure *eligibility criteria* on the scripts to limit the scripts that a user sees in the *script search*. For example, you could indicate a script should only appear on the script menu if the user belongs to the level 1 customer service representative group. You may also indicate that a script should only appear if the data a user is viewing has certain criteria. For example, if you are using Oracle Utilities Customer Care and Billing, you can indicate that a script should only appear if the current account's customer class is residential. By doing this, you avoid presenting the user with scripts that aren't applicable to the current data in context or the user's role.

The topics in this section describe eligibility rules.

Script Eligibility Rules Are Not Strictly Enforced

The *script search* gives a user a choice of seeing all scripts or only scripts that are eligible (given the current data in context and their user profile). This means that it's possible for a script that isn't eligible for the given context data / user to be executed via this search. In other words, the system does not strictly enforce a script's eligibility rules.

It might be more helpful to think of eligibility rules as "highlight conditions". These "highlight conditions" simply control whether the script appears in the *script search* when a user indicates they only want to see eligible scripts.

You Can Mark A Script As Always Eligible

If you don't want to configure eligibility rules, you don't have to. Simply indicate that the script is always eligible.

You Can Mark A Script As Never Eligible

If you have scripts that you do not want a user to select from the script menu, indicate that it is never eligible. An example of a script that you wouldn't want a user to select from the menu is one that is *launched when a To Do entry is selected*. These types of scripts most likely rely on data linked to the selected To Do entry. As a result, a user should only launch scripts of this type from the To Do entry and not from the script menu.

Criteria Groups versus Eligibility Criteria

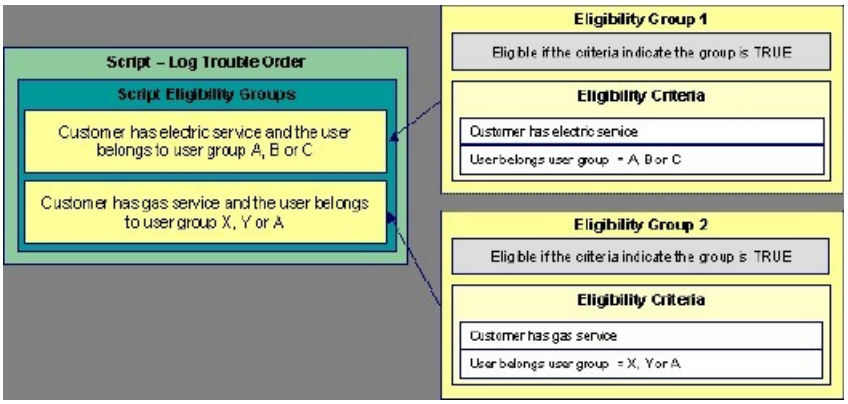
Before we provide concrete examples of eligibility criteria, we need to explain two concepts: Criteria Groups and Eligibility Criteria. A script's criteria groups control whether a user is eligible to choose a script. At a high level, it works like this:

- A criteria group has one or more eligibility criteria. A group's criteria control whether the group is considered true or false.
- When you create a group, you define what should happen if the group is true or false. You have the following choices:
  - The user is eligible to choose the script

- The user is not eligible to choose the script
- The next group should be checked

We'll use the following example from Oracle Utilities Customer Care and Billing to help illustrate these points. Assume a script is only eligible if:

- The customer has electric service and the user belongs to user group A, B or C
- OR, the customer has gas service and the user belongs to user group X, Y or A



This script requires two eligibility groups because it has two distinct conditions:

- IF (Customer has electric service AND (User belongs to user group A, B or C))
- IF (Customer has gas service AND (User belongs to user group X, Y or A))

If either condition is true, the script is eligible.

You would need to set the following criteria groups in order to support this requirement:

Group No.	Group Description	If Group is True	If Group is False
1	Customer has electric service and the user belongs to user group A, B or C	**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). Refer to SECF-ACCTFLD in the product documentation for an example of an algorithm type that retrieves a field value from an account.

The last criterion contains a time span comparison. Time span comparisons are used to compare a date to something. In our example, we have to determine the age of the customer based on their birth date. If the resultant age is > 65, they are

considered a senior citizen. To pull this off, you can take advantage of a comparison algorithm supplied with the base script as described below.

- Field to Compare. The person characteristic in which the customer's birth date is held is selected.
- Comparison Method. We chose a comparison algorithm that returns a value of **True** if the related field value (the customer's date of birth) is greater than 65 years (refer to *SECC-TIMESPN* for an example of this type of algorithm).

You'll notice that if a value of **True** is returned by the **True if senior** algorithm, the group is true (and we've set up the group to indicate a true group means the script is eligible).

> **NOTE: The time span algorithm can be used to compare days, weeks, months, etc.** Refer to *SECC-TIMESPN* for more information about this algorithm.

A Script With Service Type Comparison

Imagine a script that is only eligible if the current customer has gas service and the user belongs to user groups A, B or C. This script would need the following eligibility rules:

- Customer has gas service
- User belongs to user group A, B, or C

These rules require only one eligibility group on the script. It would looks 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 *SECF-USRNGRP* for an example of this type of algorithm.
- Comparison Method. We simply compare the value returned by the algorithm to True and indicate the appropriate response.

NOTE: Bottom line. The "field to compare" algorithm isn't actually returning a specific field's value. Rather, it's returning a value of **True** or **False**. This value is in turn, compared by the "comparison method" and the group is set to true, false or check next accordingly.

The Big Picture Of Server-Based Scripts

FASTPATH: Refer to *The Big Picture Of Scripts* to better understand the basic concept of scripts.

Server-based scripts allow an implementation to configure backend business processes. The system supports two types of server-based scripts, **Plug-In** scripts and **Service** scripts.

- Plug-in scripts allow an implementation to develop routines that are executed from the system's various plug-in spots without coding. For example, an implementation could configure a plug-in script that is executed every time an adjustment of a given type is frozen.
- Service scripts allow an implementation to develop common routines that are invoked from both front-end and back-end services. For example, an implementation could create a service script that terminates an account's automatic payment preferences. This service script could be invoked from a BPA script initiated by an end-user when a customer asks to stop paying automatically, and it could also be executed from a plug-in script if a customer fails to pay their debt on time.

The topics in this section describe background topics relevant to server-based scripts.

Plug-In Scripts

NOTE: This section assumes you are familiar with the notion of plug-in spots (algorithm entities) and plug-ins. See *The Big Picture Of Algorithms* for more information.

Rather than write a java program for a plug-in spot, you can create a plug-in using the scripting "language". In essence, this is the only difference between a program-based plug-in and a script-based one. Obviously, this is a significant difference as it allows you to implement plug-ins without programming (and compilers).

The following topics describe basic concepts related to plug-in scripts.

A Plug-In Script's API

Like program-based plug-ins, plug-in scripts:

- Run on the application server
- Have their API (input / output interface) defined by the plug-in spot (i.e., plug-in scripts don't get to declare their own API)
- Can only be invoked by the "plug-in spot driver"

The best way to understand a plug-in script's API is to use the **View Script Schema** hyperlink to view its parameters data area schema.

Notice the two groups: soft and hard. If you are familiar with plug-in spots, you'll recognize these as the classic soft and hard parameters:

- The **soft** parameters are the values of the parameters defined on the algorithm. Notice they are not named - if you want to reference them in your plug-in script, you must do it by position (this is equivalent to what a Java programmer does for plug-ins written in Java).
- The **hard** parameters are controlled by the plug-in spot (i.e., the algorithm entity). Notice that this plug-in spot has a single input parameter called " **businessObject/id"**. Also notice the **use=** attribute - this shows that this parameter is input-only (i.e., you can't change it in the plug-in script).

NOTE: XPath. You can click on an element name to see the XPath used to reference the element in your script.

Setting Up Plug-In Scripts

You can write plug-in scripts for all plug-in spots that have been Java-enabled. The following points describe how to implement a plug-in script:

- Compose your plug-in *script*, associating it with the appropriate algorithm entity (plug-in spot).
- Create a new algorithm type for the respective algorithm entity, referencing your plug-in script as the program to carry out the algorithm type's function. Only plug-in scripts associated with the algorithm entity may be referenced on the algorithm type.
- Set up an algorithm for the respective algorithm type and plug it in where applicable. Refer to *Setting Up Algorithm Types* for more information.

NOTE: No plug-in scripts are shipped with the base-package.

Service Scripts

BPA scripts run on the client's browser and guide the end-users through business processes. Service scripts run on the application server and perform server-based processing for *BPA scripts*, *zones* and more. You may want to think of a service script as a common routine that is set up via the scripting (rather than programming).

The following topics describe basic concepts related to service scripts.

A Service Script's API

As with any common routine, a service script must declare its input / output parameters (i.e., its API). A service script's API is defined on its *schema*.

NOTE: Schema Definition Tips. A context sensitive "Script Tips" zone appears when you open the *Script* page to assist you with the schema definition syntax. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema.

Invoking Service Scripts

Any type of script may *invoke a service script*:

- A BPA script may invoke a service script to perform server-based processing.
- Plug-in scripts may invoke a service script (like a "common routine").
- A service script may call another service script (like a "common routine").

Map zones may be configured to invoke service scripts to obtain the data to be displayed. Refer to *Map Zones* for more information.

Inbound web services support interaction with service scripts allowing the outside world to interact directly with a service script.

You can also invoke a service script from a Java class.

Debugging Server-Based Scripts

The server can create log entries to help you debug your server scripts. These logs are only created if you do the following:

- Start the system in **?debug=true** mode
- Turn on the **Global debug** checkbox in the upper corner of the browser. When you click this check box, a pop-up appears asking you what you want to trace - make sure to check box that causes script steps to be traced.

The logs contain a great deal of information including the contents of the referenced data area for **Move data**, **Invoke business object**, **Invoke business service** and **Invoke service script** steps.

You can view the contents of the logs by pressing the **Show User Log** button appears at the top of the browser.

Please note that all log entries for your user ID are shown (so don't share user IDs).

Maintaining Scripts

The script maintenance transaction is used to maintain your scripts. The topics in this section describe how to use this transaction.

FASTPATH: Refer to *The Big Picture Of Scripts* for more information about scripts.

Script - Main

Use this page to define basic information about a script. Open this page using **Admin** > **Script** > **Search** .

NOTE: Script Tips. A context sensitive "Script Tips" zone is associated with this page. The zone provides useful tips on various topics related to setting up scripts.

Description of Page

Enter a unique **Script** code and **Description** for the script. Use the **Detailed Description** to describe the purpose of this script in detail. **Owner** indicates if the script is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new script, carefully consider its naming convention. Refer to *System Data Naming Convention* for more information.

Script Type indicates if this is a **BPA Script**, **Plug-In Script** or **Service Script**. Refer to *The Big Picture Of BPA Scripts* and *The Big Picture Of Server Based Scripts* for more information.

Accessibility Option appears only for BPA scripts. Set this value to **Accessible from Script Menu** for any script that may be launched as a stand-alone script. Scripts with this configuration may be linked to a navigation option so that they may be invoked from a menu and may be configured by a user as a favorite script. Set this value to **Not Accessible from Script Menu** for any script that cannot be launched on its own. For example, any script that is a invoked as a sub-script from another script should have this setting. In addition, any script that is designed to be launched from within a specific portal where certain data is provided to the script should include this setting.

Enter an **Application Service** if the execution of the script should be secured. The application service should include **Execute** as one of its access modes. Refer to *Securing Script Execution* for more information.

Algorithm Entity appears only for *plug-in scripts*. Use this field to define the *algorithm entity* into which this script can be plugged in.

Business Object appears only for business object related plug-in scripts. Enter the *Business Object* whose elements are to be referenced by the plug-in script.

Script Engine Version defines the version of the XML Path Language (XPath) to be used for the script. Script engine versions 2 and 3 use the XPath 2 engine supplied by the XQuery team. This is the same engine used inside the Oracle database. The current script engine version 3 is a modified version that offers performance improvements without impacting existing version 3 scripts. **Notes regarding Script engine version 1**: The XPath library used is Jaxen; for BPA scripts, use IE's built-in MSXML parser; Xpath 1 (and even JavaScript) uses floating point arithmetic, which means that adding a collection of numbers with two decimal places might end up with a value of 10779.079999999998 instead of 10779.08.

Click the **View Script Schema** to view the *script's data areas* on the *schema viewer* window.

Click the **View XSD** hyperlink to view a service script’s schema definition in XSD format.

The **View Script As Text** hyperlink appears for server-based scripts only. Click this link to view the internal scripting commands in a separate window. The presented script syntax is valid within *edit data* steps.

The *tree* summarizes the script's steps. You can use the hyperlink to transfer you to the **Step** tab with the corresponding step displayed.

Script - Step

Use this page to add or update a script's steps. Open this page using **Admin** > **Script** > **Search** 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.

Common Step Types To All Script Types

The contents of this section describe common step types applicable to all script types.

How To Set Up Conditional Branch Steps

Conditional branch steps allow you to conditionally jump to a different step based on logical criteria. For example, you could jump to a different step in a script if the customer is residential as opposed to commercial. In addition, several fields are required for **Conditional Branch** steps:

Compare Field Type and **Compare Field Name** define the first operand in the comparison. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Current To Do Information**. Use this field type when the field being compared resides on the current To Do entry. Refer to *How To Use To Do Fields* for instructions on how to define the appropriate **Field Name**.
- **Data Area**. Use this field type when the field being compared is one that you put into one of the scripts data areas in an earlier step. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to *How To Reference Fields In Data Areas* for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model**. Use this field type when the field being compared resides on one of the tab pages in the *object display area*. Refer to *How To Find The Name Of Page Data Model Fields* for instructions on how to find the appropriate **Field Name**.
- **Predefined Value**. Use this field type when the field being compared is a *global variable*.
- **Temporary Storage**. Use this field type when the field being compared is one that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field**. Use this field type when the field being compared resides on the currently displayed tab page. Refer to *How To Find The Name Of User Interface Fields* for instructions on how to find the appropriate **Field Name**.

Condition defines the comparison criteria:

- Use >, <, =, >=, <=, <> (not equal) to compare the field using standard logical operators. Enter the comparison value using the following fields.
- Use **IN** to compare the first field to a list of values. Each value is separated by a comma. For example, if a field value must equal **1**, **3** or **9**, you would enter a comparison value of **1,3,9**.
- Use **BETWEEN** to compare the field to a range of values. For example, if a field value must be between **1** and **9**, you would enter a comparison value of **1,9**. Note, the comparison is inclusive of the low and high values.

Comparison Field Type, **Comparison Field Name** and **Comparison Value** define what you're comparing the first operand to. The following points describe each field type:

- **Current To Do Information**. Use this field type when the comparison value resides on the current To Do entry. Refer to *How To Use To Do Fields* for instructions on how to define the appropriate **Field Name**.
- **Data Area**. Use this field type when the comparison value resides in one of the scripts data areas. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to *How To Reference Fields In Data Areas* for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model**. Use this field type when the comparison value resides on one of the tab pages in the *object display area*. Refer to *How To Find The Name Of Page Data Model Fields* for instructions on how to find the appropriate **Field Name**.
- **Predefined Value**. Use this field type when the field being compared is a constant value defined in the script. When this field type is used, use **Comparison Value** to define the constant value. Refer to *How To Use Constants In Scripts* for instructions on how to use constants.
- **Temporary Storage**. Use this field type when the comparison value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field**. Use this field type when the comparison value resides on the currently displayed tab page. Refer to *How To Find The Name Of User Interface Fields* for instructions on how to find the appropriate **Field Name**.

NOTE: Conditional field types. The field types **Current To Do Information**, **Page Data Model** and **User Interface Field** are only applicable to BPA scripts.

The above fields allow you to perform a comparison that results in a value of **TRUE** or **FALSE**. The remaining fields control the step to which control is passed given the value:

- **If TRUE, Go to** defines the step that is executed if the comparison results in a **TRUE** value.
- **If FALSE, Go to** defines the step that is executed if the comparison results in a **FALSE** value.

NOTE: Numeric Comparison. Comparison of two values may be numeric or textual (left-to-right). Numeric comparison takes place only when values on both side of the comparison are recognized as numeric by the system. Otherwise, textual comparison is used. Fields for **Current To Do Information**, **Data Area**, **Page Data Model**, and **User Interface Field** types are explicitly associated with a data type and therefore can be recognized as numeric or not. This is not the case for fields residing in **Temporary Storage** or those set as **Predefined Values** . A **Temporary Storage** field is considered numeric if it either holds a numeric value moved to it from an explicitly defined numeric value (see above) or it is a resultant field of mathematical operation. A **Predefined Value** field is considered numeric if the other field it is compared to is numeric. For example, if a numeric field is compared to a **Predefined Value** the latter is considered numeric as well resulting in numeric value comparison. However, if the two fields are defined as **Predefined Values** the system assumes their values are text strings and therefore applies textual comparison.

How To Set Up Edit Data Steps

Edit data steps provide a free format region where you can specify commands to control your script processing.

In general, the syntax available within edit data mimics the commands available within the explicit step types. However, there are a few commands that are available only within edit data. For example, the two structured commands: **For**, and **If**.

For server-based scripts, you may find it useful to create a few explicit step types and then use the **View Script as Text** hyperlink on the *Script - Step* page to better understand the edit data syntax.

NOTE: Not all BPA step types are supported using the edit data syntax. **Tips.** Refer to the tips context zone associated with the script maintenance page for more information on edit data commands and examples.

Additional field required for **Edit data** steps:

Enter your scripting commands in the **Edit Data Text** field.

How To Set Up Go To Steps

Go to steps allow you to jump to a step other than the next step. Additional fields required for **Go To** steps:

Next Step defines the step to which the script should jump.

How To Set Up Invoke Business Object Steps

Invoke business object steps allow you to interact with a *business object* in order to obtain or maintain its information.

The following additional fields are required for **Invoke business object** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

Group Name references the *data area* to be passed to and from the server when communicating with the **Business Object**. Indicate the **Action** to be performed on the object when invoked. Valid values are **Add, Delete, Fast Add (No Read), Fast Update (No Read), Read, Replace, Update**.

> **NOTE: Performance note**. The actions **Fast Add** and **Fast Update** should be used when the business object's data area does not need to be re-read subsequent to the **Add** or **Update** action. In other words, the **Add** and **Update** actions are equivalent to **Fast Add + Read** and **Fast Update + Read** .

The business object call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script, to determine the step to which control is passed given the outcome of the call.

If Success, Go To defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

If Error, Go To defines the step that is executed if the call returns on error. Please note that the error information is held in *global variables*. This field is only applicable to BPA scripts.

> **NOTE: Error technique.** Let's assume a scenario where a business object is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

How To Set Up Invoke Business Service Steps

Invoke business service steps allow you to interact with a *business service*.

The following additional fields are required for **Invoke business service** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

Group Name references the *data area* to be passed to and from the server when the **Business Service** is invoked.

The business service call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script, to determine the step to which control is passed given the outcome of the call.

If Success, Go To defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

If Error, Go To defines the step that is executed if the call returns on error. Please note that the error information is held in *global variables*. This field is only applicable to BPA scripts.

> **NOTE: Error technique.** Let's assume a scenario where a business service is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

How To Set Up Invoke Service Script Steps

Invoke service script steps allow you to execute a *service script*.

The following additional fields are required for **Invoke service script** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

Group Name references the *data area* to be passed to and from the server when the **Service Script** is invoked.

The service script call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script to determine the step to which control is passed given the outcome of the call.

If Success, Go To defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

If Error, Go To defines the step that is executed if the call returns on error. Please note that the error information is held in *global variables*. This field is only applicable to BPA scripts.

> **NOTE: Error technique.** Let's assume a scenario where a service script is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

How To Set Up Label Steps

Label steps allow you to describe what the next step(s) are doing. Steps of this type are helpful to the script administrators when reviewing or modifying the steps in a script, especially when a script has many steps. When designing a script, the label steps enable you to provide a heading for common steps that belong together. The script tree displays steps of this type in a different color (green) so that they stand out from other steps.

There are no additional fields for **Label** steps.

How To Set Up Move Data Steps

Move data steps allow you to move data (from a source to a destination). The following additional fields are required for **Move data** steps:

Source Field Type, **Source Field Name** and **Source Field Value** define what you're moving. The following points describe each field type:

- **Context Variable**. Use this field type in a plug-in or service script if the source value is a variable initiated in a higher level script.
- **Current To Do Information**. Use this field type when the source value resides on the current To Do entry. Refer to *How To Use To Do Fields* for instructions on how to define the appropriate **Field Name**.
- **Data Area**. Use this field type when the field being compared is one that you put into one of the script's data areas in an earlier step. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to *How To Reference Fields In Data Areas* for instructions on how to construct the appropriate **Field Name**.
- **Global Context**. Use this field type in a BPA script when the source value is a global variable.
- **Page Data Model**. Use this field type in a BPA script when the source value resides on any of the tab pages in the *object display area* (i.e., the source field doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to *How To Find The Name Of Page Data Model Fields* for instructions on how to find the appropriate **Field Name**.
- **Portal Context**. Use this field type when the source value is a variable in the portal context.
- **Predefined Value**. Use this field type when the source value is a constant value defined in the script. When this field type is used, use **Source Field Value** to define the constant value. Refer to *How To Use Constants In Scripts* for instructions on how to use constants.

> **NOTE: Concatenating fields together.** You can also use **Predefined Value** if you want to concatenate two fields together. For example, let's say you have a script that merges two persons into a single person. You might want this script to change the name of the person being merged out of existence to include the ID of the person remaining. In this example, you could enter a **Source Field Value** of **%ONAME merged into person %PERID** (where **ONAME** is a field in temporary storage that contains the name of the person being merged out of existence and **PERID** contains the

ID of the person being kept). Refer to *How To Substitute Variables In Text* for a description of how you can substitute field values to compose the field value.

- **Temporary Storage**. Use this field type when the source value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field**. Use this field type when the source value resides on the currently displayed tab page. Refer to *How To Find The Name Of User Interface Fields* for instructions on how to find the appropriate **Field Name**.

Destination Field Type and **Destination Field Name** define where the source field will be moved. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Context Variable**. Use this field type in your plug-in or service script if you use a variable to communicate information to a lower level service script or schema.
- **Data Area**. Use this field type when the destination field resides on one of the scripts data areas. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to *How To Reference Fields In Data Areas* for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model**. Use this field type when the destination field resides on any of the tab pages in the *object display area* (i.e., the field populated doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to *How To Find The Name Of Page Data Model Fields* for instructions on how to find the appropriate **Field Name**.
- **Portal Context**. Use this field type in a BPA script when the destination to be updated is in the current portal context.
- **Temporary Storage**. Use this field type when the destination field resides in temporary storage. Use **Field Name** to name the field in temporary storage. Use **Field Name** to name the field in temporary storage. Refer to *How To Name Temporary Storage Fields* for more information.
- **User Interface Field**. Use this field type when the destination field resides on the currently displayed tab page. Refer to *How To Find The Name Of User Interface Fields* for instructions on how to find the appropriate **Field Name**.

NOTE: Conditional field types. The field types **Current To Do Information**, **Page Data Model** and **User Interface Field** are only applicable to BPA scripts.

How To Set Up Terminate Steps

Terminate steps cause a server-based script to end processing successfully or issue an error.

The following additional fields are required for **Terminate** steps:

Error indicates whether an error should be thrown or not. If error, **Error Data Text** must be specified, indicating the error message and any message substitution parameters. Refer to the tips zone associated with the Script page for the actual syntax of initiating an error message.

NOTE: The ability to terminate a step in error is only supported for server-based scripts.

Step Types Applicable to BPA Scripts only

The contents of this section describe step types that are only applicable to BPA scripts.

How To Set Up Display Text Steps

Display text steps cause a text string to be displayed in the script area. Steps of this type can be used to provide the user with guidance when manual actions are necessary. In addition, they can be used to provide confirmation of the completion of tasks.

The information you enter in the **Text** field is displayed in the *script area* when the step is executed.

The text string can contain *substitution variables* and *HTML formatting commands*. Also note that for debugging purposes, you can display an entire data area (or a portion thereof) by entering **%+...+%** where **...** is the name of the node whose element(s) should be displayed.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Height Steps

Height steps are used to change the height of the script area to be larger or smaller than the standard size.

The following additional fields are required for **Height** steps:

Script Window Height defines the number of **Pixels** or the **Percentage** (according to the **Height Unit**) that the script window height should be adjusted. The percentage indicates the percentage of the visible screen area that the script area uses. For example, a percentage value of **100** means that the script area will use the entire area.

NOTE: Standard Number of Pixels. The default number of pixels used by the script area is **75**.

NOTE: Adjust script height in the first step. If you want to adjust the height of the script area, it is recommendation to define the **height** step type as your first step. Otherwise, the script area will open using the standard height and then readjust, causing the screen to redisplay.

NOTE: Hide script area. You could use this type of step to set the height to **0** to hide the script area altogether. This is useful if the script does not require any prompting to the user. For example, perhaps you define a script to take a user to a page and with certain data pre-populated and that is all.

NOTE: Automatically close script area. If you want the script area to close when a script is completed, you could define the final step type with a height of 0.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Input Data Steps

Input data steps cause the user to be prompted to populate an input field in the script area. The input value can be saved in a field on a page or in temporary storage. A **Continue** button always appears adjacent to the input field. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Input Data** steps:

Destination Field Type and **Destination Field Name** define where the input field will be saved. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Page Data Model**. Use this field type to put the input field into a field that resides on any of the tab pages in the *object display area* (i.e., the field populated doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to *How To Find The Name Of Page Data Model Fields* for instructions on how to find the appropriate **Field Name**.
- **Temporary Storage**. Use this field type to put the input field into temporary storage. Use **Field Name** to name the field in temporary storage. Refer to *How To Name Temporary Storage Fields* for more information.

- **User Interface Field**. Use this field type to put the input field into a field that resides on the currently displayed tab page. Note, if you want to execute underlying default logic, you must populate a **User Interface Field**. Refer to *How To Find The Name Of User Interface Fields* for instructions on how to find the appropriate **Field Name**.

The **Prompt Values** grid may be used to define additional buttons. A separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button. Refer to *How To Substitute Variables In Text* for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons.
- **Next Script Step** defines the step to execute if the user clicks the button.

> **NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

How To Set Up Invoke Function Steps

> **NOTE:** Functions were implemented prior to the introduction of business services (BS), service scripts (SS) and business objects (BO). The functionality is still supported, but the recommendation for implementations going forward is to use a step that invokes one of the above configuration tool objects in a script rather than defining a function.

Invoke function steps may be used to retrieve or update data independent of the page currently being displayed. For example, if you design a script that takes different paths based on the customer's customer class, you could invoke a function to retrieve the customer's customer class.

> **FASTPATH:** You must set up a function before it can be referenced in a script. Refer to *Maintaining Functions* for the details.

The following additional fields are required for **Invoke Function** steps:

Function defines the name of the function. The function's **Long Description** is displayed below.

When a function is invoked, it will either be successful or return an error. The next two fields control the step to which control is passed given the outcome of the function call:

- **If Success, Go to** defines the step that is executed if the function is successful.
- **If Error, Go to** defines the step that is executed if the function returns on error. Refer to *How To Use Constants In Scripts* for a list of the global variables that are populated when a function returns an error.

> **NOTE: Error technique.** If a function returns an error, we recommend that you invoke a step that transfers control to a script that displays the error message information and stops (note, the error information is held in *global variables*). You would invoke this script via a **Transfer Control**.

The **Send Fields** grid defines the fields whose values are sent to the function and whose field value source is not **Defined On The Function**. For example, if the function receives an account ID, you must define the name of the field in the script that holds the account ID.

- **Field** contains a brief description of the field sent to the function.
- **Source Field Type** and **Mapped Field / Value** define the field sent to the function. Refer to the description of Source Field under *How To Set Up Move Data Steps* for a description of each field type.
- **Comments** contain information about the field (this is defined on the function).

The **Receive Fields** grid defines the fields that hold the values returned from the function. For example, if the function returns an account's customer class and credit rating, you must set up two fields in this grid.

- **Field** contains a brief description of the field returned from the function.

- **Destination Field Type** and **Mapped Field** define the field returned from the function. Refer to the description of Destination Field under *How To Set Up Move Data Steps* for a description of each field type.
- **Comments** contain information about how the field (this is defined on the function).

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Invoke Map Steps

Invoke map steps are used to invoke a *UI Map* to display, capture and update data using an HTML form. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Invoke map** steps:

Group Name references the *data area* to be passed to and from the server when rendering the HTML form associated with the **Map**.

Use **Target Area** to designate where the map will be presented.

- Select **BPA Zone** if the map should be presented within the *script area*.
- Select **Page Area** if the map should be presented in the *object display area*, i.e. the frame typically used to house a maintenance page.
- Select **Pop-up Window** if the map should be launched in a separate window.

The **Returned Values** grid contains a row for every button defined on the map.

- **Returned Value** is the value returned when the user clicks the button.
- **Use as Default** can only be turned on for one entry in the grid. If this is turned on, this value's Next Script Step will be executed if the returned value does not match any other entry in the grid. For example, if the user closes a pop-up (rather than clicking a button), the default value will be used.
- **Next Script Step** defines the step to execute if the user clicks the button.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Mathematical Operation Steps

Mathematical operation steps allow you to perform arithmetic on fields. You can also use this type of step to add and subtract days from dates. For example, you could calculate a date 7 days in the future and then use this value as the customer's next credit review date. The following additional fields are required for **Mathematical Operation** steps:

Base Field Type and **Base Field Name** define the field on which the mathematical operation will be performed. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Page Data Model**. Use this field type when the field resides on any of the tab pages in the *object display area*. Refer to *How To Find The Name Of Page Data Model Fields* for instructions on how to find the appropriate **Field Name**.
- **Temporary Storage**. Use this field type when the field resides in temporary storage. You must initialize the temporary storage field with a Move Data step before performing mathematical operations on the field. Refer to *How To Set Up Move Data Steps* for more information.
- **User Interface Field**. Use this field type when the field resides on the currently displayed tab page. Refer to *How To Find The Name Of User Interface Fields* for instructions on how to find the appropriate **Field Name**.

Math Operation controls the math function to be applied to the **Base Field**. You can specify **+**, **-**, /, and *****. Note, if the base field is a date, you can only use **+** or **-**.

Math Field Type, **Math Field Name** and **Math Field Value** define the field that contains the value to be added, subtracted, divided, or multiplied. The following points describe each field type:

- **Current To Do Information**. Use this field type when the value resides on the current To Do entry. Refer to *How To Use To Do Fields* for instructions on how to define the appropriate **Field Name**.
- **Page Data Model**. Use this field type when the value resides on any of the tab pages in the *object display area*. Refer to *How To Find The Name Of Page Data Model Fields* for instructions on how to find the appropriate **Field Name**.
- **Predefined Value**. Use this field type when the value is a constant. When this field type is used, use **Source Field Value** to define the constant value. Refer to *How To Use Constants In Scripts* for more information. Note, if you are performing arithmetic on a date, the field value must contain the number and type of **days**/ **months**/ **years**. For example, if you want to add 2 years to a date, the source field value would be **2 years**.
- **Temporary Storage**. Use this field type when the value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field**. Use this field type when the value resides in a field on the current tab page. Refer to *How To Find The Name Of User Interface Fields* for instructions on how to find the appropriate **Field Name**.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Navigate To A Page Steps

Navigate to a page steps cause a new page (or tab within the existing page) to be displayed in the object display area. Steps of this type are a precursor to doing anything on the page. The following additional field is required for **Navigate to a page** steps:

Navigation Option defines the transaction, tab, access mode (add or change) and any context fields that are passed to the transaction in change mode. For example, if you want a script to navigate to Person - Characteristics for the current person being displayed in the dashboard, you must set up an appropriate navigation option. Refer to *Defining Navigation Options* for more information.

NOTE: Navigating to a page in update mode. Before you can navigate to a page in change mode, the page data model must contain the values to use for the navigation option's context fields. If necessary, you can move values into the page data model using a *Move Data step* first. For example, before you can navigate to a page in change mode with an account ID in context, you may need to move the desired account ID into the ACCT_ID field in the page data model. The actual field name(s) to use are listed as context fields on the *navigation option*.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Perform Script Steps

Perform script steps cause another BPA script to be performed. After the performed script completes, control is returned to the next step in the original script. You might want to think of the scripts referred to on steps of this type as "subroutines". This functionality allows you to encapsulate common logic in reusable BPA scripts that can be called from other BPA scripts. This simplifies maintenance over the long term.

The following additional field is required for **Perform script** steps:

Subscript is the name of the script that is performed.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Press A Button Steps

Press a button steps cause a button to be pressed in the *object display area* or in 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 page actions toolbar at the top of the page). Refer to *How To Find The Name Of A Button* for more information.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Prompt User Steps

Prompt user steps cause the user to be presented with a menu of options. The options can be presented using either buttons or in the contents of a drop down. You can also use steps of this type to pause a script while the user checks something out (and when the user is ready to continue with the script, they are instructed to click a prompt button). The following additional fields are required for **Prompt User** steps:

Prompt Type controls if the prompt shown in the script area is in the form of **Button(s)** or a **Dropdown**. Note, if you use a **Dropdown**, a Continue button appears adjacent to the dropdown in the script area when the step executes. The user clicks the Continue button when they are ready for the script to continue.

The **Prompt Values** grid contains a row for every value that can be selected by a user. Note, if you use a **Prompt Type** of **Button(s)**, a separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button or in the dropdown entry. Refer to *How To Substitute Variables In Text* for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons or dropdown entries.
- **Use As Default** can only be turned on for one entry in the grid. If this is turned on for a dropdown entry, this value is defaulted in the grid. If this is turned on for a button, this button becomes the default (and the user should just have to press `Enter` (or `space`) rather than click on it).
- **Next Script Step** defines the step to execute if the user clicks the button or selects the dropdown value.

NOTE: Conditional step type. This step type is only applicable to BPA scripts.

How To Set Up Set Focus To A Field Steps

Set focus to a field steps cause the cursor to be placed in a specific field on a page. A **Continue** button always appears in the script area when this type of step executes. The user may click the **Continue** button when they are ready for the script to continue. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Set focus to a field** steps:

Destination Field Name defines the field on which focus should be placed. This field must reside on the currently displayed tab page. Refer to *How To Find The Name Of User Interface Fields* for instructions on how to find the appropriate **Field Name**.

The **Prompt Values** grid may be used to define additional buttons. A separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button. Refer to *How To Substitute Variables In Text* for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons.
- **Next Script Step** defines the step to execute if the user clicks the button.

> **NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

How To Set Up Transfer Control Steps

Transfer control steps cause the current BPA script to terminate and the control to pass to another BPA script. You might want to construct a BPA script with steps of this type when the script has several potential logic paths and you want to segregate each logic path into a separate BPA script (for ease of maintenance).

The following additional fields are required for **Transfer control** steps:

Subscript is the name of the script to which control is transferred.

> **NOTE: Conditional step type.** This step type is only applicable to BPA scripts.

Additional Topics

The contents of this section provide additional information about steps.

How To Find The Name Of User Interface Fields

Follow these steps to find the name of a field that resides on a page:

- Navigate to the page in question.
- Right click in the body of the page (but not while the pointer is in an input field). Note, if the field in question resides in a grid, you must right click while the pointer is in the section that contains the grid (but not while the pointer is in an input field in the grid) - this is because there's a separate HTML document for each grid on a page.
- Select **View Source** from the pop-up menu to display the source HTML.
- Scroll to the Widget Info section (towards the top of the HTML document). It contains a list of all of the objects on a page. For example, the following is an example from the Account - Main page:

```
Widget Info:
    Widget_ID , Element Type - label info - label
    ENTITY_NAME, IL - $ENTITY_NAME -  Name
    ACCT_ID, IT - $ACCT_ID -  Account ID
    ACCT_CHECK_DIGIT, IL - $ACCT_CHECK_DIGIT -  Account Check Digit
    IM_ACCT_ID, IM - $SEARCH_FOR_ACC_LBL -  Search for Account
    COLL_CL_CD, HD - $COLL_CL_CD -  Collection Class
    SETUP_DT, IT - $SETUP_DT -  Set Up Date
    CURRENCY_CD, IS - $CURRENCY_CD -  Currency Code
    CIS_DIVISION, IS - $CIS_DIVISION -  CIS Division
    PROTECT_DIV_SW, CB - CI_ACCT$PROTECT_DIV_SW -  Protect CIS Division
    CUST_CL_CD, IS - $CUST_CL_CD -  Customer Class
    ACCESS_GRP_CD, IT - $ACCESS_GRP_CD -  Access Group
    IM_ACCESS_GRP_CD, IM - $SRCH_ACC_GRP_CD -  Search for Access Group
    ACCESS_DESCR, IL - $DESCR -  Description
    ACCT_MGMT_GRP_CD, IT - $ACCT_MGMT_GRP_CD -  Account Management Group
    IM_ACCT_MGMT_GRP_CD, IM - $SEARCH_FOR_TDR_LBL -  Search for To Do Role
    MGMT_DESCR, IL
    ALERT_INFO, IA - $ALERT_INFO -  Alert Information
    BILL_CYC_CD, IS - $BILL_CYC_CD -  Bill Cycle
    BILL_AFTER_DT, IT - $BILL_AFTER_DT -  Bill After
    PROTECT_CYC_SW, CB - $PROTECT_CYC_SW -  Protect Bill Cycle
    BILL_PRT_INTERCEPT, IT - $BILL_PRT_INTERCEPT -  Bill Print Intercept
    IM_BILL_PRT_INTERCEPT, IM - $FOR_BILL_PRINT_LBL -  Search for User
    MAILING_PREM_ID, IT - $MAILING_PREM_ID -  Mailing Premise
    IM_MAILING_PREM_ID, IM - $SEARCH_FOR_MAI_LBL -  Search for Mailing Premise
    PREM_INFO, IL
    PROTECT_PREM_SW, CB - $PROTECT_PREM_SW -  Protect Mailing Premise
    dataframe, GD
```

The field names that you'll reference in your scripts are defined on the left side of the HTML (e.g., ENTITY_NAME, ACCT_ID, CUST_CL_CD, etc.).

The names of fields that reside in scrolls are in a slightly different format. The following is an example of the HTML for the persons scroll that appears on Account - Person. Notice that the fields in the scroll are prefixed with the name of the scroll plus a **$** sign. For example, the person's ID is called **ACCT_PER$PER_ID**.

```
Widget Info:
    Widget_ID , Element Type - label info - label
    ENTITY_NAME, IL - $ENTITY_NAME -  Name
    PREM_INFO, HD
    ACCT_ID, IT - $ACCT_ID -  Account ID
    ACCT_CHECK_DIGIT, IL - $ACCT_CHECK_DIGIT -  Account Check Digit
    IM_ACCT_ID, IM - $SEARCH_FOR_ACC_LBL -  Search for Account
    ACCT_PER$recordCount, SN - $OF_LBL -  Of
    ACCT_PER$PER_ID, IT - $PER_ID -  Person ID
    IM_ACCT_PER$PER_ID, IM - $FOR_PERSON_LBL -  Search for Person
    ACCT_PER$ENTITY_NAME, IL - $ENTITY_NAME -  Name
    ACCT_PER$MAIN_CUST_SW, CB - $MAIN_CUST_SW -  Main Customer
    ACCT_PER$FIN_RESP_SW, CB - $FIN_RESP_SW -  Financially Responsible
    ACCT_PER$THRD_PTY_SW, CB - $THRD_PTY_SW -  Third Party Guarantor
    ACCT_PER$ACCT_REL_TYPE_CD, IS - CI_ACCT_PER$ACCT_REL_TYPE_CD -  Relationship Type
    ACCT_PER$WEB_ACCESS_FLG, IS - $WEB_ACCESS_FLG -  Web Self Service Access Flag
    ACCT_PER$PFX_SFX_FLG, IS - $PFX_SFX_FLG -  Prefix/Suffix
    ACCT_PER$NAME_PFX_SFX, IT - $NAME_PFX_SFX -  Pfx/Sfx Name
    ACCT_PER$RECEIVE_COPY_SW, CB - $RECEIVE_COPY_SW -  Receives Copy of Bill
    ACCT_PER$BILL_RTE_TYPE_CD, IS - $BILL_RTE_TYPE_CD -  Bill Route Type
    ACCT_PER$BILL_RTE_TYPE_INFO, IL
    ACCT_PER$BILL_RTG_METH_FLG, HD
    ACCT_PER$BILL_FORMAT_FLG, IS - $BILL_FORMAT_FLG -  Bill Format
```

The names of fields that reside in grids are in a slightly different format. The following is an example of the HTML for the names grid that appears on Person - Main. Notice that the fields in the grid are prefixed with the name of the grid plus a **:x $**. For example, the person's name is called **PER_NAME:x$ENTITY_NAME**. When you reference such a field in your script, you have the following choices:

- Substitute **x** with the row in the grid (and keep in mind, the first row in a grid is row **0** (zero); this means the second row is row **1**).
- If you want to reference the "current row" (e.g., the row in which the cursor will be placed), you can keep the **x** notation (**x** means the "current row").

```
Widget Info:
    Widget_ID , Element Type - label info - label
    PER_NAME:x$NAME_TYPE_FLG, IS - $NAME_TYPE_FLG -  Name Type
    PER_NAME:x$ENTITY_NAME, IT - CI_PER_NAME$ENTITY_NAME -  Person Name
```

How To Find The Name Of Page Data Model Fields

You find the name of a **Page Data Model** field in the same way described under *How To Find The Name Of User Interface Fields*. The only restriction is that you cannot refer to hidden / derived fields. However, you can refer to ANY of the object's fields regardless of the tab page on which they appear. For example, if you position the object display area to the Main tab of the Account transaction, you can reference fields that reside on all of the tab pages.

CAUTION: If you populate a **Page Data Model** field, none of the underlying default logic takes place. For example, if you populate a customer contact's contact type, none of the characteristics associated with the customer contact type are defaulted onto the customer contact. If you want the underlying defaulting to take place, you must populate a **User Interface Field**.

How To Find The Name Of A Button

If you want a **Press a button** step to press a button in the page actions toolbar, use one of the following names:

- IM_GOBACK
- IM_HISTORY
- IM_GOFORWARD
- IM_SAVE
- IM_REFRESH
- IM_CLEAR
- IM_COPY
- IM_DELETE
- IM_ScrollBack
- IM_ScrollForward
- IM_menuButton
- IM_CTRL_CENT
- IM_CTRL_CENTAI
- IM_USER_HOME
- IM_TO_DO
- IM_PrevTo Do
- IM_NextTo Do
- IM_CurrentTo Do
- IM_MY_PREF
- IM_helpButton
- IM_aboutButton
- IM_CTI
- IM_MINIMIZE_DASHBOARD. Pressing this will collapse the dashboard.
- IM_MAXIMIZE_DASHBOARD. Pressing this will expand the dashboard.

NOTE: Can't push the BPA button. You'll notice that the BPA Scripting button is missing. This is deliberate! Don't run a script from within a script!

Follow these steps to find the name of other buttons that reside in the object display area:

- Navigate to the page in question.
- Right click in the body of the page (but not while the pointer is in an input field). Note, if the field in question resides in a grid, you must right click while the pointer is in the section that contains the grid (but not while the pointer is in an input field in the grid) - this is because there's a separate HTML document for each grid on a page.
- Select **View Source** from the pop-up menu to display the source HTML.
- Scroll to the Widget Info section (towards the top of the HTML document). It contains a list of all of the objects on a page, including buttons.
- Iconized buttons (e.g., search buttons) are represented as HTML images and their field names are prefixed with **IM**. The following is an example of the HTML on the Bill - Main page (notice the **IM** fields for the iconized buttons).

```
Widget Info:
    widget_ID , Element Type - label info - label
    BILL_INFO, IL
    BILL_ID, IT - CI_BILL$BILL_ID -  Bill ID
    IM_BILL_ID, IM - $SEARCH_FOR_A_B_LBL -  Search for a Bill
    ACCT_ID, IT - $ACCT_ID -  Account ID
    IM_ACCT_ID, IM - $FOR_AN_ACCOUNT_LBL -  Search for Account
    ACCT_NAME, IL
    IM_BILLCTXT, IM
    CREDIT_NOTE_MSG, IL
    IM_GO_TO_CR_NOTE_FR_BILL, IM - $GO_TO_BILL_LBL -  Go To Bill
```

- Transaction-specific actions buttons (e.g., the buttons use to generate and cancel a bill) are represented as switches. The following is an example of the HTML on the Bill - 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.

```
TOT_AFT_COMPLETION, IL - $DERIVED_AMT -  Payoff Balance
TOT_GEN_CHG, IL - $DERIVED_AMT -  Payoff Balance
ACTION_GENERATE_SW, BU - $GENERATE_LBL -  Generate
ACTION_FREEZE_SW, BU - $FREEZE_LBL -  Freeze
ACTION_CAN_FRZN_SW, BU - $CANCEL_FROZEN_LBL -  Cancel Frozen
ACTION_COMPLETE_SW, BU - $COMPLETE_LBL -  Complete
ACTION_FRZ_CMPL_SW, BU - $SA_ID_SRH -  SA ID
ACTION_DELETE_SW, BU - $DELETE_LBL -  Delete
ACTION_REOPEN_SW, BU - $REOPEN_LBL -  Reopen
ACTION_CR_NOTE_SW, BU - $CREDIT_NOTE_LBL -  Credit Note
```

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 "PressContinue after you've selected the customer" (the **** and **** 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.

You can also use "spans" to customize the look of the contents of a text string. For example, your text string could be "Press Continue after you've selected the customer". This would make the word "Continue" appear as large, bold, Courier text. Please refer to a Cascading Style Sheets (CSS) reference manual or website for more examples.

How To Use Constants In Scripts

Some steps can reference fields called **Predefined Values**. For example, if you want to compare an input value to the letter "Y", the letter **Y** would be defined as a Predefined Value's field value.

Special constants are used for fields defined as switches. When you move **TRUE** to a switch, it turns it on. When you move **FALSE** to a switch, it turns it off.

You can use a *global variable* as a Predefined Value. For example, if you wanted to move the current date to a field, you'd indicate you wanted to move a Predefined Value named **%CURRENT_DATE**.

How To Use Global Variables

As described above, some steps can reference fields called **Predefined Values**. In addition to referencing an ad hoc constant value (e.g., the letter **Y**), you can also reference a global variable in such a field value. A global variable is used when you want to reference system data. The following global variables exist:

- **%PARM-<name>** is the value of a parameter of that name passed in to the application when launched via the standard system URL. Refer to *Launching A Script When Starting the System* for more information on these parameters.
- **%PARM-NOT-SET** is to be used to compare against **%PARM-<name>** parameters to check if the parameter has been set or not when the application was launched. A parameter that has not been set would test as equal to this global variable. It is recommended to test parameters against this global variable before using them for the first time.
- **%CONTEXT-PERSONID** is a constant that contains the ID of the current person
- **%CONTEXT-ACCOUNTID** is a constant that contains the ID of the current account
- **%CONTEXT-PREMISEID** is a constant that contains the ID of the current premise
- **%BLANK** is a constant that contains a blank value, i.e. no value
- **%SPACE** is a constant that contains a single space value
- **%CURRENT-DATE** is the current date (as known by the browser, not the server)
- **%SAVE-REQUIRED** is a flag that contains an indication of whether the data on a page has been changed (and this requires saving). You may want to interrogate this flag to force a user to save their work before executing subsequent steps. This flag will have a value of **TRUE** or **FALSE**.

- **%NEWLINE** is a constant that contains a new line character (carriage return). Upon substitution, a line break is inserted in the resultant text.

  NOTE: The constant **%NEWLINE** does not have the desired effect when the resultant text is HTML. For example, a step's text and prompt strings. This is because HTML ignores special characters such as new lines. Refer to *How To Use HTML Tags And Spans In Text* to learn how to cause a line break in an HTML text.

In addition, if an **Invoke Function** step returns an error, the following global variables contain information about the error:

- **%ERRMSG-CATEGORY** and **%ERRMSG-NUMBER** contain the unique identifier of the error message number.
- **%ERRMSG-TEXT** contains the brief description of the error.
- **%ERRMSG-LONG** contains the complete description of the error.

How To Name Temporary Storage Fields

Input Data and **Move Data** steps can create fields in temporary storage. You specify the name of the temporary storage field in the step's **Field Name**. The name of the field must NOT begin with % and must not be named the same as the *global variables*. Besides this restriction, you can use any **Field Name** that's acceptable to JavaScript (i.e., you can name a field in temporary storage almost anything). Keep in mind that field names are case-sensitive.

How To Work With Dates

Before we discuss how to work with dates in your scripts, we need to point out that there are two types of date fields: date-only and date-time. Date-only fields only contain a date. Date-time fields contain both a date and a time. The following topics describe how to work with dates on the various step types.

NOTE: If you're working with a field that resides on the database (as opposed to a temporary storage field), the database field name will tell you what type of date it is: date-only fields are suffixed with **DT**, and date-time fields are suffixed with **DTTM**.

Move Data

If you intend to use a **Move data** step to populate a *date-time* field, please be aware of the following:

- If the destination field resides in the *page data model*, the source field value must be in the format YYYY-MM-DD-HH.MM.SS or YYYY-MM-DD. If the field is in the format YYYY-MM-DD, the time of 12:00 am will be defaulted.
- If the destination field resides in the *user interface*, you must use two steps if you want to populate both date and time. To explain this, we'll assume the field you want to populate is called EXPIRE_DTTM:
  - First, you populate the date portion of the field. To do this, you'd move a date (this value can be in any valid date format that a user is allowed to enter) to a field called EXPIRE_DTTM_FWDDTM_P1. In other words, you suffix **_FWDDTM_P1** to the field name.
  - If you want to populate the time, you'd move the time (again, the field value can be in any format that a user could use to enter a time) to a field called EXPIRE_DTTM_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 **MSGPARM[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 **MSGPARM[index]**.

A To Do entry's unique ID is accessed by using a **Field Type** of **Current To Do Information** and a **Field Value** of **TD_ENTRY_ID**.

In addition, any of the above fields can be *substituted into a text string or prompt*. Simply prefix the To Do field name with a % as you would fields in temporary storage. For example, assume you want your script to display the following text in the script area: "ABC Supply does not have a bill cycle" (where ABC Supply is the account's name). If the first sort key linked to the To Do entry contains the account's name, you'd enter a text string of **%SORTKEY[1] does not have a bill cycle**.

How To Reference Fields In Data Areas

Various step types involve referencing field elements residing in the *script's data areas*. To reference an element in a data area you need to provide its absolute XPath notation starting from the data area name. For example, use "CaseLogAdd/caseID" to reference a top-level "caseID" element in a script data area called "CaseLogAdd".

You don't have to type in long XPath notions. Use the **View Script Schema** hyperlink provided on the *Script - Step* tab page to launch the script's data areas schema.

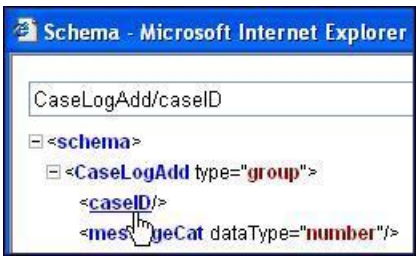


Figure 10: Schema Viewer

Doing this opens the *schema viewer* window where you can:

- Click on the field element you want to reference in your script step. The system automatically populates the text box on the top with the element's absolute XPath notation.
- Copy the element's XPath notation from the text box to your script.

You can also use the **View Data Area**, **View Service Script Data Area**, or **View Plug-In Script Data Area** links on *Script - Data Area* to the same effect. These open up the schema viewer for a specific data area respectively.

Script - Data Area

Use this page to define the data areas used to pass information to and from the server or any other data area describing your temporary storage. Open this page using **Admin** > **Script** > **Search** and then navigate to the **Data Area** tab.

Description of Page

The grid contains the script's data areas declaration. For steps that invoke an object that is associated with a schema, you must declare the associated schema as a data area for your script. In addition, if you have defined one or more data areas to describe the script's temporary storage, you need to declare them too. The following bullets provide a brief description of each field on a script data area:

- **Schema Type** defines the type of schema describing the data area's element structure.
- The data area's schema is the one associated with the referenced **Object**. Only objects of the specified Schema Type may be selected.
- **Data Area Name** uniquely identifies the data area for referencing purposes. By default, the system assigns a data area with the associated object name.
- Click on the **View Data Area** link to view the data area's schema in the *schema viewer* window.

The **View Service Script Data Area** link appears for service scripts only. Use this link to view the script's parameters data area schema in the *schema viewer* window.

The **View Plug-In Script Data Area** link appears for plug-in scripts only. Use this link to view the script's parameters data area schema in the *schema viewer* window.

FASTPATH: Refer to *A Script May Declare Data Areas* for more information on data areas.

Script - Schema

Use this page to define the data elements passed to and from a service script. Open this page using **Admin** > **Script** > **Search** and then navigate to the **Schema** tab.

NOTE: Conditional tab page. This tab page only appears for *service scripts*.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays the script name and description.

The *Schema Designer* zone allows you to edit the service script's parameters schema. The purpose of the schema is to describe the input and output parameters used when invoking the script.

> **NOTE: Script Definition Tips.** A context sensitive "Script Tips" zone is associated with this page. The zone provides a complete list of the XML nodes and attributes available to you when you construct a schema as well as other useful information assisting with setting up scripts.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. For each type of referencing entity, the *tree* displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

Script - Eligibility

Use this page to define a script's eligibility rules. Open this page using **Admin** > **Script** > **Search** and then navigate to the **Eligibility** tab.

> **NOTE: Conditional tab page.** This tab page only appears for *BPA scripts*.

Description of Page

Use the **Eligibility Option** to indicate whether the script is **Always Eligible**, **Never Eligible** or to **Apply Eligibility Criteria**. The remaining fields on the page are only visible if the option is **Apply Eligibility Criteria**.

> **CAUTION:** The following information is not intuitive; we strongly recommend that you follow the guidelines under *The Big Picture Of Script Eligibility* before attempting to define this information.

The **Eligibility Criteria Group** scroll contains one entry for each group of eligibility criteria. The following fields may be defined for each group:

- Use **Sort Sequence** to control the relative order in which the group is executed when the system determines if the script should appear in the *script search*.
- Use **Description** and **Long Description** to describe the criteria group.
- Use **If Group is True** to define what should happen if the eligibility criteria (defined in the following grid) return a value of **True**.
  - Choose **Eligible** if this script should appear.
  - Choose **Ineligible** if this script should not appear.
  - Choose **Check Next Group** if the next criteria group should be checked.
- Use **If Group is False** to define what should happen if the eligibility criteria (defined in the following grid) return a value of **False**.
  - Choose **Eligible** if this script should appear.
  - Choose **Ineligible** if this script should not appear.
  - Choose **Check Next Group** if the next criteria group should be checked.

The grid that follows contains the script's eligibility criteria. Think of each row as an "if statement" that can result in the related eligibility group being true or false. For example, you might have a row that indicates the script is eligible if the current account in context belongs to the residential customer class. The following bullets provide a brief description of each field on an eligibility criterion. Please refer to *Defining Logical Criteria* for several examples of how this information can be used.

- Use **Sort Sequence** to control the order in which the criteria are checked.
- Use **Criteria Field** to define the field to compare:
  - Choose **Algorithm** if you want to compare anything other than a characteristic. Push the adjacent search button to select the algorithm that is responsible for retrieving the comparison value. Click *here* to see the algorithm types available for this plug-in spot.
  - Some products may also include an option to choose **Characteristic**. Choosing this option displays adjacent fields to define the object on which the characteristic resides and the characteristic type. The objects whose characteristic values may be available to choose from depend on your product.
- Use **Criteria Comparison** to define the method of comparison:
  - Choose **Algorithm** if you want an algorithm to perform the comparison and return a value of True, False or Insufficient Data. Push the adjacent search button to select the algorithm that is responsible for performing the comparison. Click *here* to see the algorithm types available for this plug-in spot.
  - Choose any other option if you want to compare the **Criteria Field** using a logical operator. The following options are available:
- Use >, <, =, >=, <=, <> (not equal) to compare the **Criteria Field** using standard logical operators. Enter the comparison value in the adjacent field.
- Use **IN** to compare the **Criteria Field** to a list of values. Each value is separated by a comma. For example, if a field value must equal **1**, **3** or **9**, you would enter a comparison value of **1,3,9**.
- Use **BETWEEN** to compare the **Criteria Field** to a range of values. For example, if a field value must be between **1** and **9**, you would enter a comparison value of **1,9**. Note, the comparison is inclusive of the low and high values.
- The next three fields control whether the related logical criteria cause the eligibility group to be considered true or false:
  - Use **If True** to control what happens if the related logical criterion returns a value of True. You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.
  - Use **If False** to control what happens if the related logical criterion returns a value of False. You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.
  - Use **If Insufficient Data** to control what happens if the related logical criterion returns a value of "Insufficient Data". You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.

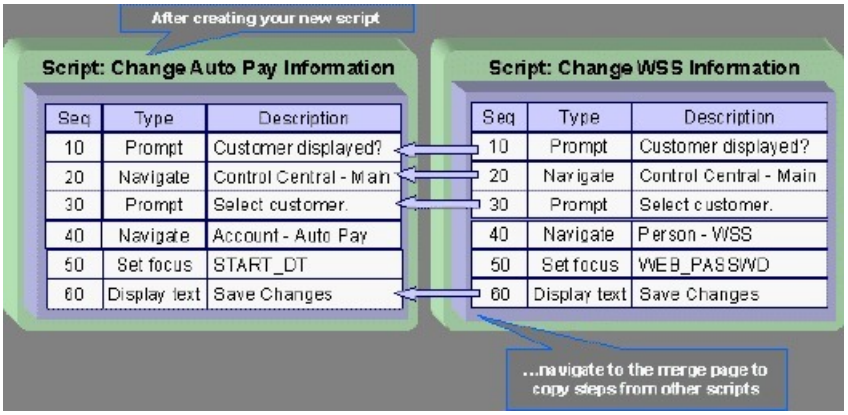
Merging Scripts

Use the Script Merge page to modify an existing script by copying steps from other scripts. The following points summarize the many diverse functions available on the Script Merge transaction:

- You can use this transaction to renumber steps (assign them new sequence numbers).
- You can use this transaction to move a step to a different position within a script. When a step is moved, all references to the step are changed to reflect the new sequence number.
- You can use this transaction to delete a step.
- You can use this transaction to copy steps from other scripts. For example,
  - You may want to create a script that is similar to an existing script. Rather than copying all the information from the existing script and then removing the inapplicable steps, this page may be used to selectively copy steps from the existing script to the new script.

- You may have scripts that are very similar, but still unique. You can use this transaction to build large scripts from smaller scripts. In this scenario, you may choose to create special 'mini' scripts, one for each of the various options that may make a script unique. Then, you could use the script merge page to select and merge the mini scripts that are applicable for a main script.

NOTE: The target script must exist prior to using this page. If you are creating a new script, you must first create the *Script* and then navigate to the merge page to copy step information.



NOTE: Duplicate versus Merge. The *Script* page itself has *duplication* capability. You would duplicate a script if you want to a) create a new script AND b) populate it with *all* the steps from an existing script.

Script Merge

Open **Admin** > **Script Merge** to open this page.

Description of Page

For **Original Script**, select the target script for merging steps.

For **Merge From Script**, select the template script from which to copy the steps.

NOTE: You may only copy steps from one Merge From script at a time. If you want to copy steps from more than one script, select the first Merge From script, copy the desired steps, save the original script, and then select the next Merge From script.

The left portion of the page displays any existing steps for the **Original Script**. The right portion of the page displays the existing steps for the **Merge From Script**.

You can use the **Copy All** button to copy all the steps from the **Merge From** script to the **Original** script. If you use **Copy All**, the steps are added to the end of the original script.

Each time you save the changes, the system renumbers the steps in the original script using the **Start From Sequence Number** and **Increment By**.

Merge Type indicates **Original** for steps that have already been saved in the original script or **Merge** for steps that have been merged, but not yet saved. The **Sequence**, **Step Type** and **Description** for each step are displayed.

The topics that follow describe how to perform common maintenance tasks:

Resequencing Steps

If you need to resequence the steps:

- Use the up and down arrows in the Original Script grid to reorder the steps.
- Make any desired changes to the **Start From Sequence Number** or **Increment By**.
- Click Save.

The steps are given new sequence numbers according to their order in the grid.

FASTPATH: Refer to *Editable Grid* in the system wide standards documentation for more information about adding records to a collection by selecting from a list and repositioning rows within a grid.

Removing a Step from Script

If you want to remove a record linked to the Original script, click the delete button, , to the left of the record.

For example, to remove the **Reset existing bundle XML** step, click the icon.

		Merge Type	Sequence	Step Type	Description
		Original	10	Edit data	Edit data - Check that the BO is an Export Bundle
		Original	20	Edit data	Edit data - Read the Bundle
		Original	30	Edit data	Edit data - Reset existing bundle XML
		Original	40	Edit data	Edit data - Create new bundle XML

After removal, the grid displays:

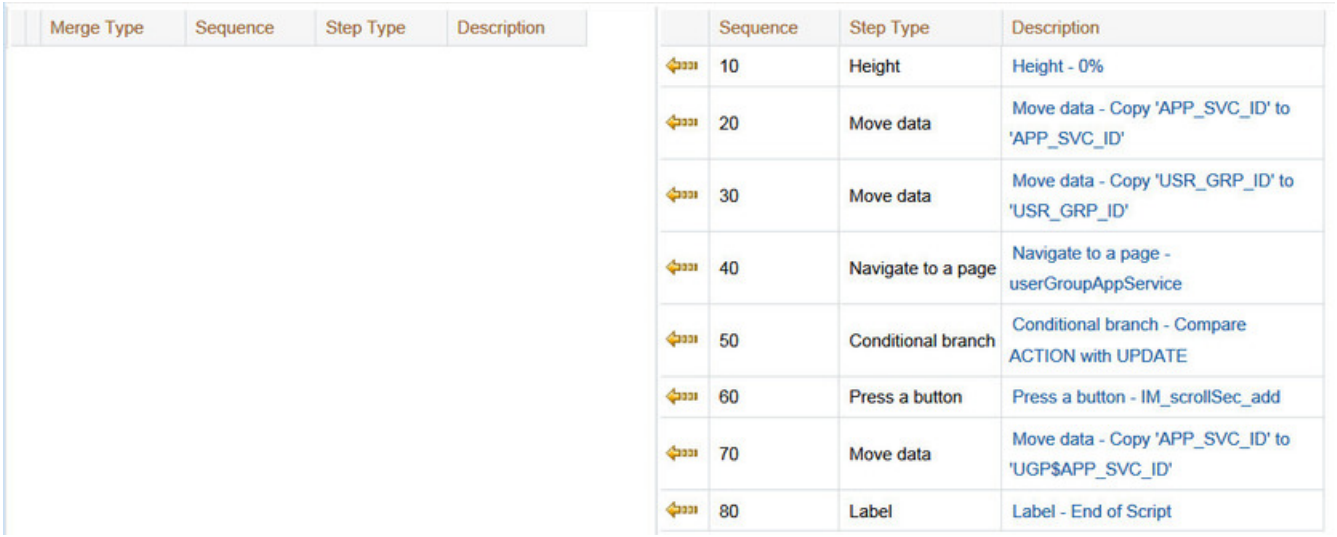
		Merge Type	Sequence	Step Type	Description
		Original	10	Edit data	Edit data - Check that the BO is an Export Bundle
		Original	20	Edit data	Edit data - Read the Bundle
		Original	40	Edit data	Edit data - Create new bundle XML

NOTE: You cannot delete a step that is referenced by other steps unless you also delete the referencing steps, such as **Go to step** or **Prompt** type steps. The system informs you of any missing referenced steps when you attempt to save the original script.

Adding a Step to a Script

You can move any of the steps from the Merge From script to the Original Script by clicking the left arrow adjacent to the desired step. Once a record is moved it disappears from the Merge From information and appears in the Original information with the word **Merge** in the Merge Type column.

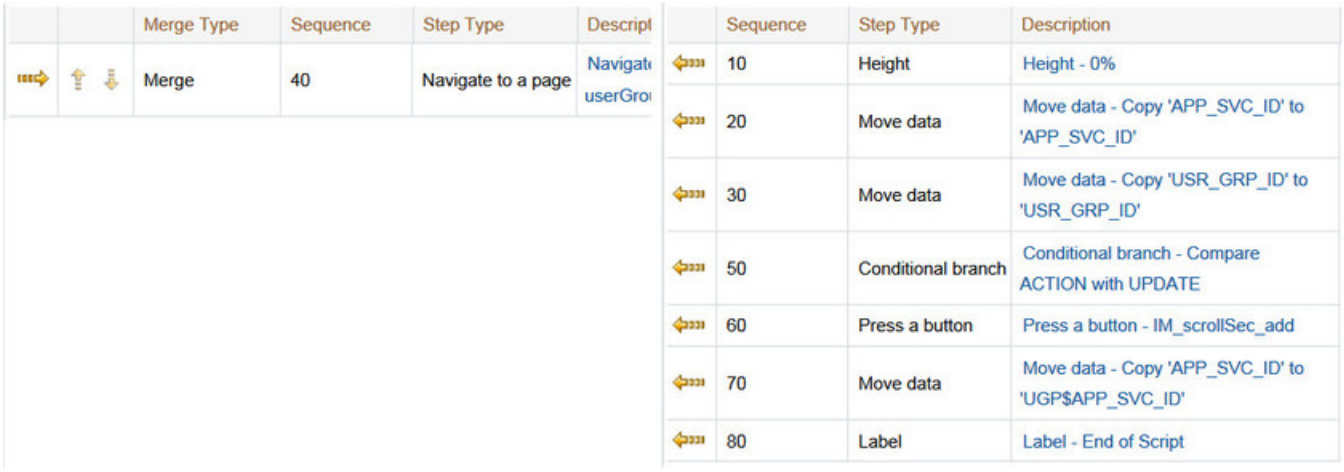
For example, to copy the **Navigate to a page** step, click the left arrow.



Merge Type	Sequence	Step Type	Description

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.



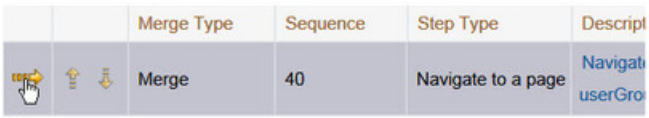
		Merge Type	Sequence	Step Type	Descript
		Merge	40	Navigate to a page	Navigat userGro

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

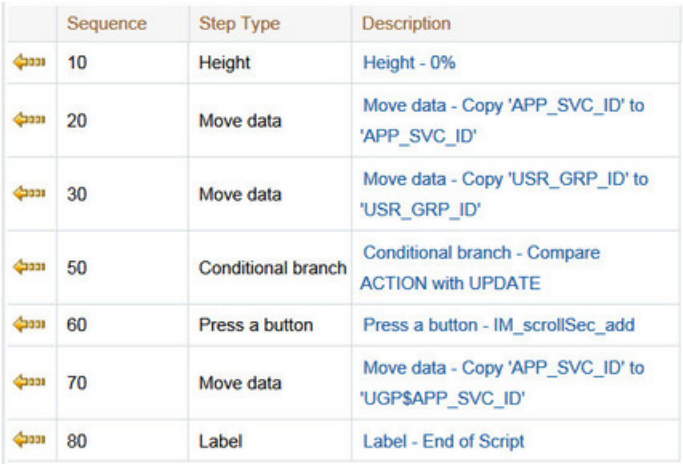
NOTE: If you add a step, such as **Go to step** or **Prompt** type steps, that references other steps, you must also add the referenced steps. The step references are updated to use the new sequence numbers when you save the original script. The system informs you of any referenced steps that haven't been added when you attempt to save the original script.

Removing an Uncommitted Step from a Script

If you have moved a row to the original script by mistake, you can remove it by clicking the right arrow adjacent to the appropriate record.



		Merge Type	Sequence	Step Type	Descript
		Merge	40	Navigate to a page	Navigat userGro



	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** > **Function** > **Search** .

Description of Page

Enter a unique **Function** code and **Description** for the function.

Use the **Long Description** to describe, in detail, what the function does.

Define the **Internal Service** that the function invokes.

NOTE: In this release, only page services can be invoked.

Click the **View XML** hyperlink to view the XML document used to pass data to and from the service. Doing this causes the XML document to be displayed in the Application Viewer.

NOTE: XML document may not be viewable. If you create a new page service and do not regenerate the application viewer, you will not be able to view its XML document.

The tree summarizes the following:

- The fields sent to the service. You can use the hyperlink to transfer to the **Send Fields** tab with the corresponding field displayed.
- The fields received from the service. You can use the hyperlink to transfer to the **Receive Fields** tab with the corresponding field displayed.
- Scripts that reference the function. You can use the hyperlink to transfer to the script page.

Function - Send Fields

Use this page to add or update the fields sent to the service. Open this page using **Admin** > **Function** > **Search** and then navigate to the **Send Fields** tab.

NOTE: Displaying a specific field. Rather than scrolling through each field, you can navigate to a field by clicking on the respective node in the tree on the Main tab. Also note, you can use the Alt+right arrow and Alt+left arrow accelerator keys to quickly display the next and previous entry in the scroll.

NOTE: You're defining the service's input fields. On this tab, you define which fields are populated in the XML document that is sent to the service. Essentially, these are the service's input fields.

Description of Page

Use **Sequence** to define the order of the **Send Fields**.

Enter a unique **Function Field Name** and **Description** for each field sent to the application service. Feel free to enter **Comments** to describe how the field is used by the service.

Use **Field Value Source** to define the source of the field value in the XML document sent to the service:

- If the field's value is the same every time the function is invoked, select **Defined On The Function**. Fields of this type typically are used to support "hard-coded" input values (so that the scriptwriter doesn't have to populate the field every time they invoke the function). Enter the "hard-coded" **Field Value** in the adjacent field.
- If the field's value is supplied by the script, select **Supplied By The Invoker**. For example, if the function retrieves an account's customer class, the script would need to supply the value of the account ID (because a different account ID is passed each time the function is invoked). Turn on **Required** if the invoker must supply the field's value (it's possible to have optional input fields).

Regardless of the Field Value Source, use **XML Population Logic** to define the XPath expression used to populate the field's value in the XML document sent to the service.

NOTE: Usability suggestion. You populate a field's value in an XML document by specifying the appropriate XPath expression for each field. Rather than referring to an XPath manual, the system can create the XPath expression for you. To do this, click the adjacent **View XML** hyperlink. This will display the XML document used to communicate with the **Service** defined on the Main page. After the XML document is displayed, click the **XPath** hyperlink adjacent to

the desired field to see how the XPath expression looks. You can then cut / paste this XPath expression into the **XML Population Logic Field**.

Function - Receive Fields

Use this page to add or update the fields received from the service. Open this page using **Admin** > **Function** > **Search** and then navigate to the **Receive Fields** tab.

NOTE: Displaying a specific field. Rather than scrolling through each field, you can navigate to a field by clicking on the respective node in the tree on the Main tab. Also note, you can use the Alt+right arrow and Alt+left arrow accelerator keys to quickly display the next and previous entry in the scroll.

NOTE: You're defining the application service's output fields. On this tab, you define which fields are populated in the XML document that is received from the service. Essentially, these are the service's output fields.

Description of Page

Use **Sequence** to define the order of the **Receive Fields**.

Enter a unique **Function Field Name** and **Description** for each field received from the service. Feel free to enter **Comments** to describe the potential values returned from the service.

Turn on **Required** if the invoker must use the field.

Regardless of the Field Value Source, use **XML Population Logic** to define the XPath expression used to retrieve the field's value from the XML document received from the service.

NOTE: Usability suggestion. You retrieve a field's value in an XML document by specifying the appropriate XPath expression for the field. Rather than referring to an XPath manual, the system can create the XPath expression for you. To do this, click the adjacent **View XML** hyperlink. This will display the XML document used to communicate with the **Service** defined on the Main page. After the XML document is displayed, click the **XPath** hyperlink adjacent to the desired field to see how the XPath expression looks. You can then copy / paste this XPath expression into the **XML Population Logic Field**.

NOTE: Fields in multiple lists. Note that the XPath expression generated in the application viewer refers to lists using a generic “list” reference. If a field within the list is unique across the service, the generic list reference is sufficient for the XML population logic. However, if the field you are trying to reference is in multiple lists, the XPath must include the list name. Adjust the Application Viewer’s generated XPath by adding the list name, which can be found in the overview panel in the Service XML viewer. For example, instead of **/pageBody/list/listBody/field[@name=’FIELD_NAME’]**, the XPath Population Logic must read **/pageBody/list[@name=’LIST_NAME’]/listBody/field[@name=’FIELD_NAME’]**.

Chapter 11

Attachments

Some implementations may require that attachments be available from the application. These attachments can be stored in the Attachment table and then linked to other objects if applicable.

Attachment Overview

The following topics provide additional information regarding attachment functionality.

Attachment Types

The system supports several different attachment content types, for example:

- PDF Document
- Excel Spreadsheet
- Jpeg Image
- Text Document

The attachment data itself may be text or binary. When storing the data in the application however, it is stored as text information only. As a result, the upload of an attachment that is a binary type requires a conversion prior to storing the data. When viewing the attachment, the data is converted again for display.

Each type of attachment is defined using an attachment business object. The business object includes configuration defining the supported file extensions, whether the data is binary or not and the content type that represents the type of data for the attachment.

NOTE: To view the attachment business objects provided with the base product, navigate using **Admin** > **Business Object** > **Search** and search for business objects related to the Maintenance Object for Attachments (**F1-ATCHMT**).

Owned Attachments

Attachments can be either ‘owned’ or ‘common’. An owned attachment is one that is related to a specific record. For example, the specific test results for a given device can be uploaded and linked to that device or to its test records. These types of attachments are typically uploaded and maintained via the object that owns it.

Common Attachments

Common attachments are ones that are uploaded independent of any transaction in the system. They can be used for general system or company information. Or they can be linked to more than one transaction. For example, instructions for performing a certain type of task can be uploaded as an attachment and linked to a task type where those instructions are relevant. These types of attachments are uploaded and maintained in the central Attachment portal. Objects that may refer to the attachments may link the attachments via characteristics or some other appropriate mechanism.

Emailing Attachments

The system supports a business service that may be used by system processing to send an email. The business service **F1-EmailService** supports receiving the IDs of one or more attachments as input parameters.

Refer to *Sending Email* for more information.

Configuring Your System for Attachments

In order to link attachments to objects in the system, there may be some configuration or implementation required to support the link. It is possible that one or more objects in your product already support attachments out of the box. Consult the product documentation for the specific object for confirmation. For objects in the system that do not support attachments out of the box, the following sections provide some guidelines for enabling support for attachments. Contact product support for more information.

Supporting Common Attachments

The attachments themselves are created / uploaded using the attachment portal. Refer to *Attachments — Main* for more information.

If your implementation has a use case where one or more common attachments may be linked to an object (and the object does not already support this functionality), the object may need to be extended to capture the attachments.

- If the object includes a characteristic collection, this is a recommended way to capture attachments. A characteristic type should be defined for each type of attachment. The characteristic type should be a foreign key type and should reference the Attachment FK reference. The characteristic entity collection should include the object that the common attachment will be linked to.
- Most characteristic collections are sequence based characteristics and would support multiple entries for the same characteristic types, if multiple attachments are applicable.
- If the object to support the attachments is governed by a business object, the implementation must extend the business object to define one or more appropriate elements used to capture the attachments. If only one attachment of a certain type is allowed, a single flattened characteristic may be used. If multiple attachments of a certain type are allowed, the BO schema may define a “flattened list” exposing the sequence and the characteristic type.
- If the object is maintained on a “fixed page” with a generic characteristic collection, no additional configuration is needed to allow users to link attachments to that object.

Supporting Owned Attachments

When creating an attachment for a specific record, the attachment itself captures the information about the related record, namely its maintenance object code and its primary key. For these types of attachments, no configuration is needed on the related business object to capture the attachments, as was the case with common attachments.

However, it is recommended to configure the user interface of the related object so that the owned attachments can be viewed and maintained from that page. This typically entails the creation of a special zone that retrieves a list of existing attachment records that reference the current record as its foreign key (owner). The zone would include links or buttons to add, upload or view an attachment.

If your product already has support for viewing and maintaining owned attachments on one of the base portals, that may be used as an example to follow. If your product does not have support for owned attachments, contact customer support for more information.

Defining a New Attachment Type

As mentioned, the product provides support for several content types. If your implementation needs to support attachments for a content type not currently supported, create a new business object copying the configuration of an existing attachment business object.

Configure the following option types for the BO:

- **Binary** indicates whether the attachment data must be converted from binary format. Binary attachments are stored in the database as text, and are then converted back to the original format when retrieved.
- **Content Type** represents the browser's mime type of the attachment.
- **Supported File Extension** specifies the valid file extensions for the content type.

Once the business object is defined, it is ready for use.

Maintaining Attachments

This section describes the functionality supported for maintaining attachments.

Attachment - Query

Use the attachment query to search for an existing attachment. Navigate to this page using **Admin** > **Attachment** > **Search** .

Once an attachment has been selected, you are brought to the maintenance portal to view and maintain the selected record.

Attachment - Main

This portal appears when an attachment has been selected from the Attachment Query portal or when drilling into a common attachment.

The attachment zone provides basic information about an attachment, including the ability to upload the file and to view an uploaded file.

How to Add an Attachment

Use the following steps to create a new common attachment.

1. Launch the add dialogue either from the Add hyperlink on the Attachment Search zone or via the menu using **Admin** > **Attachment** > **Add** .
2. When prompted, choose the appropriate attachment business object for the type of file you are attaching. Click **OK**.
3. Enter the Attachment Name and click **Save**.
4. After the attachment page displays, click **Upload**.
5. In the upload dialogue, use the **Browse** button to find the file to attach and then click **Upload**.

The attachment zone provides basic information about an attachment, including the ability to upload the file and to view an uploaded file.

Chapter 12

Data Synchronization

NOTE: The information in this section applies to the integration with Oracle Utilities Meter Data Management and/or Oracle Utilities Network Management System.

Defining Sync Requests

This section highlights functionality provided to help an implementation manage the communication of data to an external system.

The Big Picture of Sync Requests

Your implementation may need to communicate certain data to external systems. This may be part of a data warehousing requirement or an integration effort. The synchronization process has two main parts. First, the change to the data must be detected and captured. Once that is accomplished, the next step is to manage the communication of that change to the external systems involved. The changes must be captured in chronological order so as to avoid systems going out of sync.

Capturing the Change

The base product has provided the **Audit** plug-in spot on the MO to allow for processing to be called by the system upon detecting a change to an MO. The framework calls the algorithm defined on this plug-in spot in the event a change to the MO has been detected. The base product comes with a generic change data capture algorithm **F1-GCHG-CDCP** that makes use of this plug-in spot to create the sync request BOs. The **Sync Request BO** MO Option is used to define these sync request BOs. The generic algorithm loops through this list of BOs and instantiates them, if no instances of those BOs yet exist in the initial state. The MO and primary key used for these sync requests will be the MO being modified and its primary key.

Sometimes it is necessary to instantiate a sync request whose MO is different from the MO being modified. As an example, a change to the Person record may need to instantiate an SA sync request. In this case, your implementation will have to

create your own Audit algorithm that captures this idiosyncratic logic. For examples on how this is done, please refer to the *Oracle Utilities CCB-MDM Integration Implementation Guide*.

Sync Request Maintenance Object

The sync request maintenance object (MO) is used to communicate data elements to an external system. It holds the data elements that are to be synchronized or extracted out of your system and also holds the MO and primary key constituents that drive the sync request. If your organization wishes to use this MO, you can either use the business object (BO) supplied in the base product or configure your own BO.

This MO provides the following functionality:

- A business object option **External System** is provided for business objects of this type. The value specified on this option is used by the base product to create an entry in the JMS Queue.
- A business object option **Outbound Message Type** is provided for business objects of this type. The value specified on this option is used by the base product to create an entry in the JMS Queue.
- An **Information** algorithm is provided (**F1-SYNC-INFO**).
- The standard **Determine BO** algorithm is provided.
- The standard logs are provided.
- The standard characteristics collection is provided.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference F1-SYNC REQ.

For more information about this MO and to review the business objects defined for this MO, navigate to **Admin** > **Maintenance Object** > **Search** and view the **MO F1-SYNC REQ**.

Sync Request Business Object

The sync request business object (BO) contains the rules that govern the processing of a sync request. The algorithms in this BO control what data elements are synchronized to an external system; how this data is to be communicated to the external system; and how the communication of this information is controlled chronologically to ensure that changes are processed in the right order. The base product provides the BO **F1-SyncRequest**, which contains the following life cycle.

- **Pending**. The initial state of the sync request BO. At this point, a pre-processing algorithm would have already taken a snapshot of what the relevant information looked like prior to the change. A deferred monitor algorithm is specified on this state so as to accumulate all changes prior to sending the sync request to the external system. Furthermore, in order to prevent the sync request from transitioning to the next state while another sync request may not have completed its processing yet, the base package provides the monitor algorithm **F1-WAITRELSR** that skips transitioning if it detects another instance of the same BO in a non-final state for the same MO and primary key combination.
- **Determine If Sync Needed**. This transitory state will contain the algorithm that takes the final snapshot of the relevant information so that it may be compared against the initial snapshot to see if the change affected any of the relevant data elements. The base product provides the algorithm **F1-COMPSNAPS** for this purpose. Your implementation may also add your own rules here to discard the sync request if certain criteria are met.
- **Discarded**. If the comparison of the initial and final snapshots yields no changes, or if it fails any other checks your implementation has introduced, the sync request is transitioned to this state. If any processes are dependent on the sync request transitioning to a final state, your implementation may plug in an enter algorithm on this state to kick off the dependent process.

- **Send Request**. This transitory state is used to hold the algorithm that facilitates the sending of the sync request to the external system. The algorithm used here may write a message to a JMS Queue or write a record to the general process MO or perform any other task that is suited to how your implementation communicates the change.
- **Awaiting Acknowledgement**. This state can be used to hold the sync request from further state transitions until an acknowledgement is received from the external system. We strongly recommend designing some form of acknowledgement that the external system has processed the information as this helps control the chronological flow of information. The base package provides the business service **F1-UpdateSyncRequest** that transitions the sync request to either the next default state (in this case the Synchronized state) if a positive acknowledgement is received; or the state associated with the Rejection transition condition (in this case the Error state) if a negative acknowledgement is received.
- **Synchronized**. Once the information has been processed and acknowledged by the external system, the sync request is transitioned to this state. If any processes are dependent on the sync request processing successfully, your implementation may want to plug in an enter algorithm on this state to kick off the dependent process.
- **Error**. If a negative acknowledgement is received or if the acknowledgement is not received within the prescribed period of time, the sync request is transitioned to this state. Your implementation may wish to create a to do entry when the sync request transitions to this state. The base product provides the enter algorithm **F1-TDCREATE** for this purpose. The exit algorithm **F1-TODOCOMPL** has also been provided to complete any to do entries when leaving this state.
- **Canceled**. A user may transition the sync request to this state from either the **Awaiting Acknowledgement** state or the **Error** state if he wishes to discontinue processing of the sync request for any reason. Your implementation will have to configure your own cancel reasons to document the transition. If any processes need to be triggered when the sync request reaches this state, your implementation can plug in an enter algorithm containing such logic.

Designing Your Sync Requests

The starting point for designing sync requests is the analysis of the external system's data requirements. Your implementation will need to analyze the external system's data model and compare it against your system's data model. Where the relevant data reside in your system will need to be identified. If the data reside in more than one MO, a decision must be made to either combine the information in one sync request or to break up the sync request into several ones.

For each sync request identified, the driving MO must be determined. The BO for each sync request can then be created as a sub-class of the base BO **F1-SyncRequest**. For an example of how these BOs are configured, please refer to *Oracle Utilities CCB-MDM Integration Implementation Guide* or *Oracle Utilities CCB-NMS Integration Implementation Guide*.

Sync Requests

The Sync Request portal is used to view information about data synchronizations from Oracle Utilities Customer Care and Billing to Oracle Utilities Meter Data Management, or from Oracle Utilities Customer Care and Billing to Oracle Utilities Network Management System. Navigate to this portal by using **Menu** > **Data Synchronization** > **Sync Request Search** .

Sync Request Query

Use the *query portal* to search for an existing Sync Request. Once a Sync Request record is selected, you are brought to the maintenance portal to view and maintain the selected record.

Sync Request Portal

This portal appears when a sync request has been selected from the Sync Request Query portal.

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

Sync Request Actions

This is a standard **actions zone**.

If the Sync Request is in a state that has valid next states, buttons to transition to each appropriate next state are displayed.

Related Sync Request

The Related Sync Request zone shows all related sync requests that are in a non-final state. Refer to the zone's help text for additional information about this zone's fields.

Sync Request

The Sync Request zone shows display-only information about the sync request. Refer to the zone's help text for additional information about this zone's fields.

Sync Request - Log

The sync request log page contains an entry for every recorded event during the lifecycle of the sync request. There are two general types of log entries:

- **Automatic entries**. The system automatically creates an entry in the log when usage is requested or there is a status change or when a related entity is created. This also includes any implementation-specific log entries. Users cannot modify or delete these log entries.
- **Manual entries**. Users can add manual entries to record significant events at their discretion.

Chapter 13

Application Viewer

The Application Viewer allows you to explore meta-data driven relationships and other deliverable files online.

> **NOTE:** Running Stand-Alone. You can also launch the Application Viewer as a stand-alone application (i.e., you do not need to start it from within the system). Refer to *Application Viewer Stand-Alone Operation* for more information about running the Application Viewer as a stand-alone application.

To open the application viewer from within your application, navigate to **Admin** > **Application Viewer** . The application viewer may also be launched from other locations for example when viewing a section of the online help files that contain hypertext for a table name, clicking on that hypertext brings you to the definition of that table in the data dictionary.

Application Viewer Toolbar

The Toolbar provides the main controls for using the Application Viewer. Each button is described below.

Data Dictionary Button

The **Data Dictionary** button switches to the Data Dictionary application.

Physical and Logical Buttons

The **Physical** button changes the display in the List Panel from a logical name view to a physical name view. Note that the Tables are subsequently sorted by the physical name and therefore may not be in the same order as the logical name view. Once clicked, this button toggles to the Logical button.

The **Logical** button changes the display in the List Panel from a physical name view to a logical name view. Note that the Tables are subsequently sorted by the logical name and therefore may not be in the same order as the physical name view. Once clicked, this button toggles to the Physical button.

These buttons are only available in the Data Dictionary.

Collapse Button

The **Collapse** button closes any expanded components on the list panel so that the child items are no longer displayed.

This button is only available in the Data Dictionary viewer.

Attributes and Schema Button

The **Attributes** button changes the display in the Detail Panel from a related tables view to an attribute view. Once clicked, this button toggles to the Schema button.

The **Schema** button changes the display in the Detail Panel from an attribute view to a related tables view. Once clicked, this button toggles to the Attributes button. Note that only tables have this view available. Columns are always displayed in an attribute view.

These buttons are only available in the Data Dictionary.

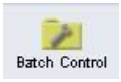
Maintenance Object Button

The **Maintenance Object** button switches to the Maintenance Object viewer application.

Algorithm Button

The **Algorithm** button switches to the Algorithm viewer application.

Batch Control Button



The **Batch Control** button switches to the Batch Control viewer application.

To Do Type Button

The **To Do Type** button switches to the To Do Type viewer application.

Description and Code Buttons

The **Description** button changes the display in the List Panel to Description (Code) from Code (Description). Note that the list is subsequently sorted by the description. Once clicked, this button toggles to the Code button.

The **Code** button changes the display in the List Panel to Code (Description) from Description (Code). Note that the list is subsequently sorted by the Code. Once clicked, this button toggles to the Description button.

These buttons are only available in the Batch Control and To Do Type viewers.

Service XML Button

The **Service XML** button switches to the Service XML viewer. This button is not available when you are already in the Service XML viewer.

You are prompted to enter the name of the service XML file you want to view. The name of the service XML file should be entered without the extension.

Select Service Button

The **Select Service** button loads another service XML file that you specify. This button is only available in the Service XML viewer.

You are prompted to enter the name of the service XML file you want to view. The name of the service XML file should be entered without the extension.

Java Docs Button

The **Java Docs** button switches to the Java Docs viewer.

Classic Button

This button is only available in the Java Docs viewer.

The **Classic** button launches the classic Javadocs viewer on a separate window. If you are more comfortable with that look you can use this viewer instead.

Preferences Button

The **Preferences** button allows you to set optional switches used by the Application Viewer. Refer to *Application Viewer Preferences* for more information.

Help Button

The **Help** button opens the Application Viewer help system. You used this button to access this information.

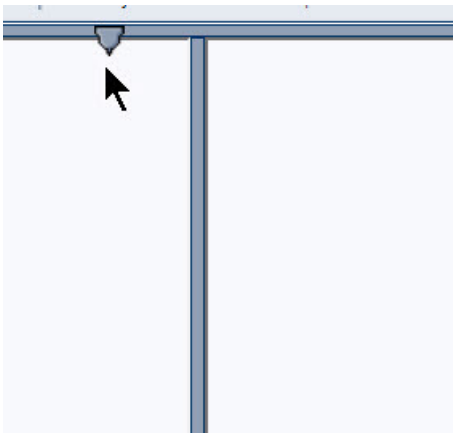
About Button

The **About** button opens a window that shows when was each Application Viewer data component recently built.

Data for all application viewer components may be regenerated to incorporate up-to-date implementation-specific information. Refer to *Application Viewer Generation* for further details.

Slider Icon

This "slider" icon allows you to resize the list panel and detail panel to your preferred proportions.



Data Dictionary

The data dictionary is an interactive tool that allows you to browse the database schema and to graphically view relationships between tables in the system.

To open the data dictionary, click the *Data Dictionary button*. You can also open the data dictionary by clicking the name of a table in other parts of the application viewer or in the online help documentation.

> **NOTE: Data Is Generated.** A background process generated the data dictionary information. Refer to *Application Viewer Generation* for further details.

Using the Data Dictionary List Panel

The list panel displays a list of tables and their columns. The list panel can list the table names by either their logical names or their physical names. Click the appropriate *button* on the toolbar to switch between the two views. The list is displayed in alphabetical order, so the order may not be the same in both views. Both views function in a similar manner.

In the list panel, you can navigate using the following options:

- Click the right arrow icon to expand a table to show its columns.
- Click the down arrow icon to collapse the column list for a table. Optionally, collapse all column lists by using the **Collapse** button.
- Click the column name to display information about the column in the detail panel.
- If the detail panel is in *related table* view, click the table name to view its related tables. If the detail panel is in *table detail* view, click the table name to display its information.

Primary And Foreign Keys

The columns in the list panel may display key information as well as the column name:

- A yellow key indicates that the column is a primary key for the table.
- A light blue key indicates that the column is a foreign key to another table. If you hover the cursor over the icon, the tool tip indicates the foreign table.
- A dark blue key indicates that the column is a conditional foreign key. A conditional foreign key represents rare relationships between tables where a single field (or set of fields) may reference multiple primary key constraints of other tables within the application as a foreign key.
- A red key indicates that the column is a logical key field. A logical key represents an alternate unique identifier of a record based on a different set of fields than the primary key.

If you hover your cursor over an icon, the tool tip indicates the key type.

Field Descriptions Shown

The language-specific, logical name of each field is shown adjacent to the physical column name in the data dictionary. You can enter an override label for a *table / field's* to be used throughout the system as the field's logical name. Here too it is the override label that is shown.

> **NOTE: Regenerate.** You should regenerate the data dictionary after overriding labels. Refer to *Application Viewer Generation* for further details.

Using the Data Dictionary Detail Panel

The Data Dictionary detail panel displays the details of the selected item. There are three main displays for the Detail Panel:

- Related tables view
- Table detail view
- Column detail view

Related Tables View

The Related Tables view displays information about the table's parent tables and child tables. Click the *Schema* button in the toolbar to switch to related tables view.

In the related tables view, you can navigate using the following options:

- Click the left arrow and right arrow icons to view the related tables for that linked table. The List Panel is automatically positioned to the selected table.
- Click the maintenance object icon () to view the table's maintenance object.
- If you want to position the *List Panel* to view the columns for different table click the name of the table for which you want to view the columns.

Table Detail View

The table detail view displays information about the selected table. Click *Attributes* (in the toolbar) to switch to the table detail view.

In the table detail view, you can navigate using the following options:

- If user documentation is available for the table, click the View User Documentation link to read the user documentation that describes the table's maintenance object.
- If the table has an associated Language Table, click the link to view the Language Table details.
- If there is an associated Maintenance Program, click the link to view the source code for the maintenance program (you are transferred to the *Java Docs Viewer*).
- If there is an associated Key Table, click the link to view the Key Table details.

Column Detail View

Click on a column name in the list panel to switch to the column detail view. The Column Detail view displays information about the selected column.

In the column detail view, you can navigate using the following options:

- If user documentation is available for the column, click the View User Documentation link to read about the column's related maintenance object.
- If the column is a foreign key, click the table name to switch to the Table Detail view for that table.
- If the column has a Value List available (normally only present for a subset of flag and switch fields), click the link to view the source code for the copybook (you are transferred to the *Java Docs Viewer*).

Lookup Values

If the selected column is a lookup field its valid values are also listed. Notice that you can enter an override description for *lookup values*. In this case the override description is shown.

> **NOTE: Regenerate.** You should regenerate the data dictionary after overriding lookup value descriptions. Refer to *Application Viewer Generation* for further details.

Maintenance Object Viewer

The maintenance object viewer is an interactive tool that allows you to view a schematic diagram of a maintenance object. A maintenance object is a group of tables that are maintained as a unit.

To open the Maintenance Object Viewer, click the *Maint. Object* button in the application viewer or click a *maintenance object icon* in the Data Dictionary.

> **NOTE: Data Is Generated.** A background process generated the maintenance object information. Refer to *Application Viewer Generation* for further details.

Using the Maintenance Object List Panel

The list panel displays a list of maintenance objects. In the list panel, you can click the maintenance object name to display information about the maintenance object in the detail panel.

Using the Maintenance Object Detail Panel

The Maintenance Object detail panel displays a schematic of the selected maintenance object.

In the detail panel, you can navigate using the following options:

- Click a table name to transfer to the Data Dictionary *table detail view* for a table. (Click the *Maint. Object* button in the toolbar to return to the maintenance object.)
- Click the service XML icon () to view the XML file of the Service Program used to maintain the displayed object. (Click the *Maintenance Object* button in the toolbar to return to the maintenance object.)

Algorithm Viewer

The algorithm viewer is an interactive tool that allows you to view algorithm types (grouped by their plug-in spot) and their related algorithms.

To open the Algorithm Viewer, click the *Algorithm* button in the application viewer. The Algorithm viewer may also be opened from certain locations in the online help documentation.

> **NOTE: Data Is Generated.** A background process generates algorithm information. Refer to *Application Viewer Generation* for further details.

Using the Algorithm Viewer List Panel

The list panel displays a list of algorithm types and their related algorithms, grouped by their plug-in spot.

In the list panel, you can navigate using the following options:

- Click the algorithm plug-in spot description to display information about the plug-in spot in the detail panel.
- Click the right pointer ▸ icon to expand a plug-in spot and view its algorithm types and their related algorithms.
- Click the down pointer ▾ icon to collapse the list of algorithm types for a plug-in spot.
- Click the algorithm type name to display information about the algorithm type in the detail panel.
- Click the algorithm name to display information about the algorithm in the detail panel.

Using the Algorithm Plug-In Spot Detail Panel

The Algorithm plug-in spot detail panel displays further information about the selected plug-in spot.

Using the Algorithm Type Detail Panel

The Algorithm Type detail panel displays further information about the selected algorithm type.

In the Algorithm Type detail panel, you can navigate using the following options:

- Click on the program name to view its source in the Java docs viewer.

Using the Algorithm Detail Panel

The Algorithm detail panel displays further information about the selected algorithm.

Batch Control Viewer

The batch control viewer is an interactive tool that allows you to view batch controls.

To open the Batch Control Viewer, click the *Batch Control* button in the application viewer. The Batch Control viewer may also be opened from certain locations in the online help documentation.

> **NOTE: Data Is Generated.** A background process generates batch control information. Refer to *Application Viewer Generation* for further details.

Using the Batch Control Viewer List Panel

The list panel displays a list of batch controls. The list panel can display the list of batch controls sorted by their code or sorted by their description. Click the appropriate *button* on the toolbar to switch between sorting by the code and description.

In the list panel, you can click the batch control to display information about the batch control in the detail panel.

> **NOTE: Not All Batch Controls Included**. Note that the insertion and key generation programs for conversion (CIPV*) are not included.

Using the Batch Control Detail Panel

The batch control detail panel displays further information about the selected batch control.

In the batch control detail panel, you can navigate using the following options:

- Click on the program name to view its source in the Java docs viewer.
- If a To Do type references this batch control as its creation or routing process, click on the To Do type to view its detail in the To Do type viewer.

To Do Type Viewer

The to do type viewer is an interactive tool that allows you to view to do types defined in the system.

To open the To Do Type Viewer, click the *To Do Type* button in the application viewer. The To Do Type viewer may also be opened from certain locations in the online help documentation.

> **NOTE: Data Is Generated.** A background process generates To Do type information. Refer to *Application Viewer Generation* for further details.

Using the To Do Type Viewer List Panel

The list panel displays a list of To Do types. The list panel can display the list of To Do types sorted by their code or sorted by their description. Click the appropriate *button* on the toolbar to switch between sorting by the code and description.

In the list panel, you can click the To Do type to display information about the To Do type in the detail panel.

Using the To Do Type Detail Panel

The To Do type detail panel displays further information about the selected To Do type.

In the To Do type detail panel, you can navigate using the following options:

- If the To Do type references a creation process or a routing process, click on the batch process to view its detail in the batch control viewer.
- Click on the table listed in the drill key section to view its detail in the data dictionary.
- Click on the field(s) listed in the drill key section to view its detail in the data dictionary.

Service XML Viewer

The service XML viewer is an interactive tool that allows you to browse the XML files of service programs that execute on the application server.

You can access the service XML viewer as follows:

- The maintenance object viewer allows you to view the XML file of the maintenance object's service program. This feature is implemented by viewing the maintenance object and then clicking on the *Service XML icon*.
- When viewing a maintenance object on the *Maintenance Object* page, clicking the **View XML** hyperlink causes the service's XML document to be displayed in the Service XML Viewer.
- When viewing a business service on the *Business Service* page, clicking the **View XML** hyperlink causes the service's XML document to be displayed in the Service XML Viewer.
- When setting up a *Function*, you may want to view the XML document used to pass data to and from the service. Clicking the **View XML** hyperlink causes the XML document to be displayed in the Service XML Viewer.

Using the Service XML Viewer Overview Panel

The overview panel displays a high level nodes and list names structure of the XML document.

In the overview panel, you can click on any node item to position the detail panel to view that item.

Using the Service XML Viewer Detail Panel

The detail panel displays nodes and attributes of the selected XML file.

Click the **xpath** button to view the XML path that should be used to reference the selected node in the XML document. The box at the top of the overview panel changes to display this information.

NOTE: Fields in multiple lists. Note that the generated XPath expression refers to lists using a generic "list" reference. For example: **/pageBody/list/listBody/field[@name='FIELD_NAME']**. If a service has a field that appears in more

than one list, the above XPath may not be sufficient for referencing that field. In this case, references to the XPath should be adjusted to include the list name. The list name is visible in the overview panel. To add the list name, use [@name='LIST_NAME']. For example: **/pageBody/list[@name='LIST_NAME']/listBody/field[@name='FIELD_NAME']**.

Java Docs Viewer

The Java Docs viewer is an interactive tool that allows you to browse Java documentation files (Javadocs) for Java classes that execute on the application server.

NOTE: Proprietary Java Classes. A small number of Java classes have been suppressed due to their proprietary nature.

NOTE: **Classic view.** If you are more comfortable using the classic Javadocs viewer you may use the *Classic* button.

To open the Java Docs viewer from within the application viewer, click the *Java Docs button*. Additionally, the *algorithm viewer* and the *batch control viewer* allows you to view the Javadocs of a program written in Java.

Using the Java Docs Viewer List Panel

The list panel displays a tree of Java packages where each package may be expanded to list the Java interfaces classes it includes.

In the list panel, you can navigate using the following options:

- Click the right arrow icon to expand a Java package to view the Java interfaces and classes it includes.
- Click the down arrow icon to collapse the list for a Java package. Optionally, collapse all lists by using the **Collapse** button.
- Click the Java interface or class name to display information about it in the detail panel.

The list details panel designates the interfaces and the classes as follows:

- A green dot indicates Java interfaces.
- A blue key indicates Java classes.

If you hover the cursor over the icon, the tool tip indicates whether it's an interface or a class.

Using the Java Package Detail Panel

The package detail panel displays a summary of the various Java classes that are included in the selected Java package.

Click the Java class name to display information about the Java class in the detail panel.

Using the Java Interface / Class Detail Panel

The detail panel displays Java documentation information about the selected Java interface or class.

You can navigate using hyperlinks to other locations in the current detail panel or to view the details of other Java interfaces / classes.

Application Viewer Preferences

This panel displays the Available Languages and allows you to select the language in which the labels and buttons are displayed. Select your desired language and click OK.

Application Viewer Stand-Alone Operation

You can run the Application Viewer as a stand-alone application (i.e., you do not need to launch it from the online application environment). To run it as a stand-alone application, you should copy the Application Viewer files (all files in the appViewer directory) and the online help files (all files in the help directory) to the server on which you want to run the Application Viewer.

> **NOTE:** Online Help. If you do not copy the online help files, online help will not be available for the Application Viewer, nor will you be able to view business descriptions of the tables' maintenance objects.

To start the application viewer in stand-alone mode, launch the appViewer.html file (located in the appViewer directory).

Stand-Alone Configuration Options

You can configure the Application Viewer for stand-alone operation by modifying options in a configuration file. The Application Viewer comes with a default configuration file called config_default.xml (located in the appViewer\config directory). Create a copy of the default configuration file and rename it to config.xml. Modify the options described in the following table to suit the needs of your installation.

> **NOTE:** Default Configuration. If you do not create the config.xml file, the Application Viewer launches with its default (internal) configuration.

Option	Description
defaultLanguage	The default language used when the application viewer is started. Available values are those marked as language enabled on the language page.
defaultView	The default view then the application viewer is started. Available values include: - Data Dictionary - Source Viewer
dataDictionary	Whether the Data Dictionary is available or not: - Y - N
sourceCode	Whether the Source Code Viewer is available or not: - Y - 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">Y</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.

Chapter 14

Defining and Designing Reports

This section describes how to configure your third party reporting tool and how to define your reports in the system to enable users to submit reports online.

The Big Picture Of Reports

The topics in this section describe the approach for designing and defining your system reports.

Integration with BI Publisher and Business Objects Enterprise

Your DBMS, your product, and BI Publisher or Business Objects Enterprise / Crystal Reports can work together to produce reports. You may choose to use a different reporting tool, but this may not be a trivial effort. This section provides high-level information about some of the business requirements that are being solved with the reporting solution.

Reports Must Be Multi-Language

The system supports a multi-language implementation and the reporting solution for the system must also support a multi-language implementation.

- If a French-speaking user requests a given report, all labels, headings and messages are in French. If the same report is requested by an English-speaking user, all information is in English
- Different fonts may be necessary for a given report (the font for Chinese is rather different than the font for English)
- Dates and numbers must be formatted as per the user's profile in your product
- Currency must be formatted as per the currency definition in your product

In order to provide the above functionality, the third party reporting tool must do the following:

- Access the system's field and message metadata for labels, headings and messages (as different values can be defined for different languages in system's metadata)

- Retrieve the appropriate font, size, and layout based on the requested report and the user's language
- Use the currency code information in the system to format currency-oriented information
- Use the user's display profile to format date and number fields

Requesting Reports from The System

Although reports are rendered in your reporting tool, users must be able to request ad-hoc reports from within the system (assuming users have the appropriate security access).

- The prompts for the input parameters must be shown in the user's language
- Users should be able to use the standard search facilities to find parameter values
- Plug-ins can be optionally used to cross-validate input parameters
- Application security must authorize ad-hoc report requests

Overview of the Data - BI Publisher

The following diagram provides an overview of where data is stored for your reports for integration with BI Publisher.

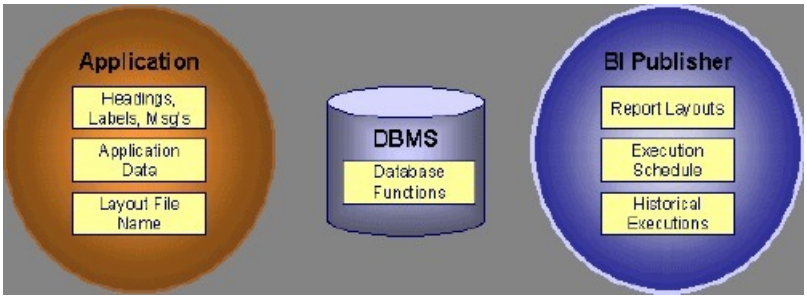


Figure 11: Application and BI Publisher

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 a Character Large Object (CLOB) can be displayed on reports produced using BI Publisher. See your product's *Optional Products Installation Guide* for information on this configuration.

Overview of the Data - Business Objects Enterprise

The following diagram provides an overview of where data is stored for your reports for integration with Business Objects Enterprise.

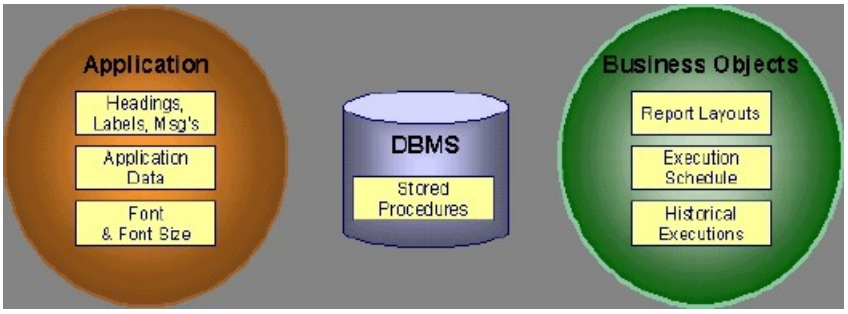


Figure 12: Application and Business Objects Enterprise

The application contains:

- The application data that appears on your reports.
- The language-specific headings, labels and messages on your reports.
- For Business Objects Enterprise, the font and size in which each report is rendered.

Business Objects Enterprise contains:

- How your reports look.
- When your reports are produced (the batch scheduler).
- Historical images of reports.

The DBMS contains the SQL used to retrieve the data on your reports (residing in stored procedures).

How To Request Reports

A user may request an ad hoc report from within your product:

- A *report submission* page enables a user to choose the desired report and enter the parameter values for the report
- The user must be granted security access to the report
- The request is passed to the reporting tool real time. Refer to *Configure The System to Invoke BI Publisher* or *Configure The System to Invoke Business Objects Enterprise* for more information.
- The reporting tool generates the report and displays it in a new browser window

The reporting tools' scheduler creates reports (as per your schedule)

- This function is entirely within the reporting tool. No scheduling functions reside within your product.

A user can request an ad-hoc report from within the reporting tool

- Note, the user's ID must be supplied as a parameter to the report in order for the user's profile to be used to format dates and numbers

Viewing Reports

As described above, ad-hoc reports requested from within your product are displayed immediately after they are generated in a new browser window

Crystal's report repository can be used to retrieve historical versions of a report. The *Report History* page allows users to open the Crystal's report execution history page and request a view of this report.

NOTE: The *Report History* page currently does not display historical reports for BI Publisher.

Configuring The System To Enable Reports

Configuring BI Publisher Reports

This section contains topics specific about configuring the product to interoperate with BI Publisher.

Configure the System to Invoke BI Publisher Real-time

The base product provides an *installation algorithm* plug-in spot called Reporting Tool. This plug-in spot should contain an algorithm that invokes the third party reporting tool real-time.

For BI Publisher, the system provides an algorithm type called *F1-BIPR-INV*, which invokes BI Publisher.

These algorithms rely on information defined in the *Reporting Options* table: the reporting server, reporting folder and the user name and password for accessing the reporting tool. The values in the reporting options should have been set up when the system was installed. Contact your system administrator if there are any problems with the values defined on the reporting options.

To use the algorithm types to invoke BI Publisher, perform the following steps:

- Create an *algorithm* for the appropriate algorithm type.
- On the *installation options*, add an entry to the algorithm collection with an algorithm entity of **Reporting Tool** and indicate the algorithm created in the previous step.

Batch Scheduling in BI Publisher

For many of your reports, you probably want the report to be produced on a regular basis according to a scheduler. The reporting solution relies on the BI Publisher software to provide the batch scheduler functionality. Refer to BI Publisher documentation for details about configuring the batch scheduler.

NOTE: The *report history* page currently does not display historical reports for BI Publisher.

Configuring Business Objects Enterprise Reports

This section contains topics specific about configuring the product to interoperate with Business Objects Enterprise.

Configure the System to Invoke Business Objects Enterprise Real-time

The base product provides an *installation algorithm* plug-in spot called Reporting Tool. This plug-in spot should contain an algorithm that invokes the third party reporting tool real-time.

For Business Objects Enterprise, the system provides an algorithm type called *RPTE-INV*, which invokes Business Objects Enterprise.

These algorithms rely on information defined in the *Reporting Options* table: the reporting server, reporting folder and the user name and password for accessing the reporting tool. The values in the reporting options should have been set up

when the system was installed. Contact your system administrator if there are any problems with the values defined on the reporting options.

To use the algorithm types to invoke one of the reporting tools, perform the following steps:

- Create an *algorithm* for the appropriate algorithm type.
- On the *installation options*, add an entry to the algorithm collection with an algorithm entity of **Reporting Tool** and indicate the algorithm created in the previous step.

Batch Scheduling in Business Objects Enterprise

For many of your reports, you probably want the report to be produced on a regular basis according to a scheduler. The reporting solution relies on the Business Objects Enterprise software to provide the batch scheduler functionality. Refer to Business Objects Enterprise documentation for details about configuring the batch scheduler.

The product provides a *report history* page to display report instances that were produced via the batch scheduler and are stored in a repository. The report history page relies on the *reporting tool algorithm* to invoke Business Objects Enterprise and display the historic instances for the selected report.

Defining Reporting Options

The reporting options are provided as a mechanism for defining information needed by your reporting solution. The base product uses the reporting options to define information needed to access the reporting tool from within the system using the algorithm defined on the installation option.

Navigate to this page using **Admin** > **Reporting Options** .

Description of page

The following information must be defined to interface with BI Publisher real-time. Contact your system administrator to report any problems with the settings defined here.

Reporting Folder Defines the shared folder where reports are stored.

For Business Objects Enterprise, defines the name of the virtual directory on the server where Java Service pages (JSP) are located. The reporting tool algorithm uses this information to construct the URL to launch the reporting tool. The reporting tool algorithm assumes that a JSP named "logon.jsp" is located there.

Reporting Server Defines the URL of the web application where the reporting tool is installed. For example, using BI Publisher, the format is: `http://<BI Publisher Server>:<port>`.

Reporting Tool User ID Not applicable when integrating with BI Publisher.

For Business Objects Enterprise, defines the user id to use when logging in.

Reporting Tool Password Not applicable when integrating with BI Publisher.

For Business Objects Enterprise, defines the password to use when logging in.

> **NOTE: Customize Options**. The reporting options are customizable using the Lookup table. This field name is **RPT_OPT_FLG**. The reporting options provided with the system are needed to invoke the reporting tool. If your implementation requires other information to be defined as reporting options, use the lookup table to define additional values for the reporting option flag.

Where Used

This information is used by the reporting tool algorithm on the *installation option* to invoke the reporting tool software.

Implementations may use reporting options to record other information needed for their reporting tool.

Defining Report Definitions

For each report supplied by your installation, use the report definition page to define various attributes of the report.

Report Definition - Main

Navigate to this page using **Admin** > **Report Definition** > **Search** .

CAUTION: Important! If you introduce new report definitions, you must prefix the report code with **CM**. If you do not do this, there is a slight possibility that a future release of the application could introduce a new system report with the name you allocated.

Description of page

Enter an easily recognizable **Report Code** and **Description** for each report. Use the **External Reference ID** to define the identifier for this report in your external reporting tool.

Define an *application service* to enable users to request submission of this report online or to view report history for this report. Once you define an application service for each report, use *application security* to define which users may access this report.

NOTE: **Access Mode**. The access mode for application services related to reports must be set to **Submit/View Report**.

If you have more than one parameter defined for your report and you wish to perform cross-validation for more than one parameter, provide an appropriate **Validation Algorithm**. Click *here* to see the algorithm types available for this system event.

Enter a **Long Description** to more fully describe the functionality of this report. This information is displayed to the user when attempting to submit the report online or when viewing history for this report.

For BI Publisher, if you want to use one of the sample reports provided by the system, but with a different layout, indicate the layout to use for the report in the **Customer Specific Font/ Layout** field and BI Publisher uses this information instead. The name for base report layout is <report code>_Base. For example, a base layout for CI_VACANT is named CI_VACANT_Base.

For Business Objects Enterprise, the **Report Font** and **Report Font Size** are used to control the display of the report information. If you wish to use one of the sample reports provided by the system, but wish to use a different font and font size, indicate your **Customer Specific Font** in the **Customer Specific Font/Layout** field and Business Objects Enterprise uses this information instead.

Report Definition - Labels

Navigate to this page using **Admin** > **Report Definition** > **Search** and go to the **Labels** tab.

NOTE: **Company name and logo.** Note the company name used as a title in the sample reports is defined as a message on the *installation options*. For information about installing the company logo, refer to the *Reports Configuration* chapter of the *Installation Information*.

Description of Page

In order to provide multi-language capability for each report, the labels used for the report must support multiple language definitions. For each label used by your report, indicate a unique **Sequence** and the *Field* used to define the **Label**. The label defined here should be the same label that is defined in your report layout defined in the external reporting tool.

When rendering an image of the report, the external reporting tool retrieves the appropriate label based on the language used for the report.

Report Definition - Parameters

Navigate to this page using **Admin** > **Report Definition** > **Search** and go to the **Parameters** tab **.**

Description of Page

The **Parameters** scroll contains one entry for every parameter defined for the report. To modify a parameter, simply move to a field and change its value. To add another parameter, click + to insert a row and then fill in the information for each field. The following fields display:

Parameter Code The identifier of the parameter. This must correspond to the parameter definition in the reporting tool.

Required Turn on this switch if the user must supply a value for the parameter when submitting the report.

Sort Sequence Indicate the sort sequence for this parameter. This sequence must match the parameter order defined in the reporting tool's report. It is also used when displaying the list of parameters on the *report submission* page.

Characteristic Type Indicate the characteristic type used to define this parameter.

Default Value Use this field to define a default value for this parameter. Default values are displayed to the user when the report is chosen on the *report submission* page.

Description Define a brief description of the parameter. This description is used when displaying the parameter on the *report submission* page.

Long Description Define a detailed description of the parameter. This description is used on the *report submission* page when the user requests more information for a given parameter.

Sample Reports Supplied with the Product

The system provides several sample reports that may be used by your organization 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.

NOTE: Additional sample reports. Your specific product may supply additional sample reports. Information about any additional reports, if applicable would be found in your specific product's documentation. Open the help index and navigate to the index entry labeled **reports / sample reports for < product>**.

How to Use a Sample Report Provided with the System

If you would like to use any of the sample reports, you need to perform some steps to be able to execute them in an implementation environment. This section walks you through the steps needed.

Steps Performed at Installation Time

The *Installation Guide* provides instructions for setting up and configuring your product and reporting tool to use the sample reports provided with the system. The following steps are described there.

- Setting up the stored procedures used by the sample reports.
- Defining the company title and logo used by the sample reports. Note the company name used as a title in the sample reports is defined as a message on the *installation options*. For information about installing the company logo, refer to the *Reports Configuration* chapter of the *Installation Information*.

- Defining a user for integration with your product.
- Publishing the sample reports in BI Publisher or Business Objects Enterprise.

Contact your system administrator to verify that the above steps have occurred.

Subreports Used with Crystal Reports

The sample reports supplied with the system use several common subreports. Subreports are used in Crystal Reports to retrieve common data such as, labels and your company title. They are shared for all reports and may be reused for customer reports. Implementers may also use these subreports when designing new reports.

NOTE: Specific Subreports. This section only includes common subreports that may be reused by new reports. You may notice that other subreports are supplied with the system. These subreports provide functionality for a specific sample reports and are not meant for reuse.

Display Company Logo and Title

The subreport **CIZCOMP** receives the user id as a parameter and calls the stored procedure **CIZCOMP** . It retrieves the company's title in the user's language from the appropriate *installation message* record.

FASTPATH: Refer to the **Reports Configuration** chapter of the installation guide for more information about defining the location for the company logo.

Format Report Information

The subreport **CIZINST** defines shared variables that are used for formatting fields in the main report. It calls the stored procedure **CIZINST** . This subreport receives the user id and report code as parameters. It retrieves the font and font size from the *report definition* . It retrieves the format date/time and number format from the user's *display profile* . Finally, it retrieves the currency from the *installation* record and retrieves the currency symbol and position from the currency's record.

NOTE: Multi-currency. All reports support multiple currencies. Currency information is returned for each row by the appropriate stored procedure. This subreport retrieves the currency code from the Installation Options and should only be used in a report if there is no other currency information available.

Labels

The subreport **CIZLABEL** keeps all labels used in the main report. It calls the stored procedure **CIZLBALL** with the user ID as a parameter. This stored procedure returns all labels defined for all reports. The subreport selects labels specified for the current report and sets shared variables L001...L100 to store the labels. If more than 100 labels are required for a new report, the version of the CIZLABEL subreport used for the new report should be changed to add additional shared variables.

How To Define A New Report

Use a Sample Report as a Starting Point

- Make a copy of the report and save it in an appropriate directory. Prefix the new report name with **CM**.
- Review the stored procedure(s) used for this report. Refer to the installation guide for information about where the stored procedures should be defined. If you want to change the data that is being accessed, copy the stored procedure, prefixing the new stored procedure with **CM**. Make the appropriate changes in the new version of the stored procedure. Contact your database administrator to find out the procedure for creating a new stored procedure.

  NOTE: Performance considerations. When designing a stored procedure, you must consider the performance of the report when executed. Consult your database administrator when designing your database access to ensure that all issues are considered.

  NOTE: Defining Messages. The stored procedures provided with the system use messages defined in message category 30. If your new stored procedures require new messages, use message category 90000 or greater, which are reserved for implementations.

- Review the parameters used by the report. Make appropriate changes to the parameters required by the report. This affects how you define your report. Refer to *Designing Parameters* for more information.
- Determine whether or not you require cross validation for your report parameters. If any cross validation is necessary, you should design an appropriate validation algorithm to be executed when requesting a report in your product. Refer to *Designing Validation Algorithms* for more information.

  NOTE: Cross Validation for On-line Submission Only. The cross validation algorithm is only executed for ad-hoc report submissions via your product. If you submit this report through your reporting tool, this algorithm is not executed.

- Review the labels used by the report. Labels and other verbiage are implemented in the sample reports using a reference to the field table in the system. This enables the report to be rendered in the appropriate language for the user. For any new report label you require, you must define a new field entry. Refer to *Designing Labels* for more information.
- Review the layout of the report and make any desired changes based on your business needs.

When you have finished designing and coding your new report in your reporting tool, you must do the following in order for it to be usable:

- Publish the report in BI Publisher or Business Objects Enterprise. Refer to the documentation for these products for details about publishing a report. Refer to *Publishing Reports in BI Publisher* and *Publishing Reports in Business Objects Enterprise* for configuration information specific to publishing a report for integration with your product.
- Define the report. Refer to *Designing Your Report Definition* for more information.

Publishing Reports in BI Publisher

Please refer to the documentation for BI Publisher for more information about publishing a report in this system. The remaining topics in this section provide information about settings needed to ensure that the report is accessible using BI Publisher.

BI Publisher Database Access

When publishing a report in BI Publisher, you are asked for database logon information. The logon user name and password must be the user name and password that has access to the database functions related to this report in your database.

Verify BI Publisher User Access Rights

To verify the user's access rights to folders in BI Publisher:

- Open the BI Publisher Enterprise Security Center.
- Check that the role for the user has access to the appropriate report folders.

For more information, refer to the "Understanding Users and Roles" section in the Oracle Business Intelligence Publisher User's Guide.

Publishing Reports in Business Objects Enterprise

Please refer to the documentation for Business Objects Enterprise for more information about publishing a report in this system. The remaining topics in this section provide information about settings needed to ensure that the report is accessible using Business Objects Enterprise.

Business Objects Enterprise Database Access

When publishing a report in Business Objects Enterprise, you are asked for database logon information. The logon user name and password must be the user name and password that has access to the stored procedures related to this report in your database.

Verify Parameter Definition

This section describes how to verify parameter definitions in the Crystal Management Console (CMC).

- Once your report has been published, navigate to the CMC. This is the web-based administration component for Business Objects Enterprise and provides access to all administrative functions.
- To verify/change the settings of a report in the CMC go to the Objects management area and select the desired report by clicking its link located in the Object Title column.
- Once you have selected your report, click the **Parameters** tab to change the settings.
- If your report requires parameters to be provided by the user, you must configure the parameter settings in the CMC to ensure that parameter values are passed from the system when submitting the report via the *Report Submission* page. If you plan to submit reports from within your product, the **Prompt the user for new value(s) when viewing** check box should be checked.
- Click **OK** on this window and then click **Update.**

NOTE: Submitting Reports Through Business Objects Enterprise. If you plan to submit reports from Business Objects Enterprise, you must also define an appropriate initial value for each parameter, if applicable.

NOTE: User ID. The user id is defined as the first parameter in every sample report. This parameter is hidden when the report is submitted from within your product, but it must be defined in the Crystal report.

Verify Business Objects Enterprise User Access Rights

To verify the access rights for a user in CMC:

- Navigate to the **Rights** tab in the Objects management area of the CMC and check that the user has correct security level for the report.
- Integration with your product requires an access level of **View On Demand** for the user.

Designing Your Report Definition

When adding a new report, you must define it in the system to allow users to request ad-hoc reports from on-line and to take advantage of the multi-language provisions in the system. The following topics illustrate the steps to take to correctly configure your report definition.

Designing Main Report Definition Values

Refer to field description section of the *report definition* main page for information about defining general information about the report.

For the validation algorithm, preliminary steps are required. Refer to *Designing Validation Algorithms* for more information.

For the application service, preliminary steps are required. Refer to *Designing Application Services* for more information.

Designing Characteristic Types

The parameter tab on the report definition page uses *characteristic types* to define the report parameters. For each report parameter that you plan to use, you must define a characteristic type.

You do not need a unique characteristic type for each report parameter. For example, if Start Date and End Date are parameters your report, only one **Report Date** characteristic type needs to be defined. This characteristic type would be used on both date parameters.

Each characteristic type to be used as a report parameter must indicate a characteristic entity of **Report**.

To illustrate the characteristic type definitions, let's look at the sample report Tax Payables Analysis. It needs the following parameters: From Date, To Date, GL Account Type Characteristic Type and Account Type value.

> **NOTE: Account Type Parameters**. The tax payables report must find general ledger entries that have posted to a certain distribution code. In order to find the appropriate distribution code, the report expects each distribution code to be defined with a characteristic indicating its GL account type (for example, **Revenue**, **Asset**, etc.) The report needs to know the characteristic type used to define this entry.

To support the required parameters, the following characteristic types are needed.

Char Type	Description	Type	Valid Values	Char Entities
CI_DATE	Date Parameter	Ad-hoc	(Uses validation algorithm to validate proper date entry)	Report
CI_CHTYP	Characteristic Type	FK Reference	**CHAR_TYP**	Report
CI_GLTY	GL Account Type	Pre-defined	A- Asset, E- Expense, LM- Liability/ miscellaneous, LT- Liability/taxes, R-Revenue	Distribution Code, Report

Highlights for some of the above settings:

- We have defined a characteristic type for defining a characteristic type. This is to allow the user to indicate which Char Type on the Distribution Code is used for the GL account type. This is an FK reference type of characteristic.
- The GL account type characteristic type is referenced on both the Distribution Code entity and the report entity.

Designing Parameters

Your report definition parameters collection must define a unique parameter entry for each parameter sent to the reporting tool. The sequence of your parameters must match the sequence defined in your reporting tool.

Continuing with the Tax Payables Analysis report as an example, let's look at the parameter definitions.

Parameter Code	Description	Char Type	Default Value
P_FROM_DT	From Date	CI_DATE	N/a
P_TO_DT	To Date	CI_DATE	N/a
P_CHAR_TYPE	Account Type Characteristic	CI_CHTYP	CI_GLTY
P_TAX_ACCTY_CHAR	Account Type Char Value for Tax Related GL Account	CI_GLTY	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 table using **Admin** > **Field** > **Add** .
- 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.

Chapter 15

External Messages

This section describes mechanisms provided in the product that enable an implementation to configure the system to communicate with an external application.

Incoming Messages

This section provides information about support for incoming messages.

Inbound Web Services

Inbound web service functionality is provided to support receiving web service requests from an external system.

Overview

The following topics provide overview information about inbound web services (IWS).

Multiple Operations

An inbound web service supports the configuration of one or more operation per web service. Each operation defines the schema-based object to invoke to perform the desired function. An operation may refer to a Business Service, a Business Object, or a Service Script. If the IWS supports multiple operations, each operation can refer to the same or a completely different schema-based object from other operations within the IWS. In addition, each operation may define a transaction type, which is essentially an action that is supported by the schema-based object.

By default, Inbound Web Services uses the Schema Name to dictate the Request and Response for the service. The API can be overridden with custom formats by specifying Request and Response Schemas with the appropriate Request and Response XSL to transform into the relevant schema formats.

Annotations Used for Security

When preparing to deploy inbound web services, the security aspects of the service must be decided. There are three options available:

- Attach a security policy to the IWS via a Web Service Annotation. The product supplies the security policy **Username Token Policy** that may be used. This attaches a policy file (UsernameToken.xml) to the Inbound Web Service at deployment time.
- Attach security policies to the Inbound Web Service via the J2EE Web Application Server. This allows for multiple policies to be attached as support by the J2EE Web Application Server.
- A combination of the internal and external security policies.

Inbound Web Service Deployment

Once an Inbound Web Service it must be deployed to the J2EE Web Application Server in order for it to be available to the Web Service Clients to access the system. Refer to *Deploying Web Services* for more information.

Configuring Inbound Web Service Options

This topic describes the configuration needed for using inbound web services.

Technical Configuration

In order to use inbound web services, there are tasks a system administrator must perform.

Refer to the Server Administration Guide for technical details of each of these processes.

Maintaining Web Service Annotation Types

The product provides the annotation type **Policy Annotation** that defines the WS-Policy annotation. If your implementation wishes to define additional annotation types, use the Web Service Annotation Types portal. Open this page using **Admin** > **Web Service Annotation Type** > **Search** .

You are taken to the query portal where you can search for an existing web service annotation type. Once an annotation type is selected, you are brought to the maintenance portal to view and maintain the selected record.

NOTE: Use of custom policies should only be considered if the policies supplied by the J2EE Web Application Server are not sufficient for your implementation’s needs.

Web Service Annotation Type Portal

This page appears when viewing the detail of a specific web service annotation type.

It contains a zone that includes display-only information about the web service annotation type along with the actions that are available for maintaining an annotation type.

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

Maintaining Web Service Annotations

The product provides the annotation **Username Token Policy**, which references the annotation type **Policy Annotation** and enables WS-security. If your implementation wishes to define additional annotations, use the Web Service Annotation portal. Open this page using **Admin** > **Web Service Annotation** > **Search**

The topics in this section describe the base-package zones that appear on the Web Service Annotation portal.

Web Service Annotation List

The Web Service Annotation *List zone* lists every annotation. The following functions are available:

- Click a *broadcast* button to open other zones that contain more information about the adjacent annotation.
- The standard actions of **Edit, Delete** and **Duplicate** are available for each annotation.
- Click the **Add** link in the zone's title bar to add a new annotation.

Web Service Annotation

The Web Service Annotation zone contains display-only information about an annotation, including its parameters. This zone appears when an annotation has been broadcast from the Web Service Annotation List zone or if this portal is opened via a drill down from another page.

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

Maintaining Inbound Web Services

Inbound Web Services are used to define a specific message that your implementation will receive from an external system and provides configuration needed to process the inbound message.

The product provides several inbound web services out of the box. By default, no annotations are defined for the base inbound web services. You may modify the message options or the annotations for any base IWS record. In addition, you may define additional IWS records for other incoming messages supported by your implementation.

To view an inbound web service, navigate using **Admin** > **Inbound Web Service** > **Search** .

Inbound Web Service Query

Use the *query portal* to search for an existing inbound web service. Once an inbound web service is selected, you are brought to the maintenance portal to view and maintain the selected record.

Click the **Add** link in the zone's title bar to add a new inbound web service.

Inbound Web Service - Main

This portal appears when an inbound web service has been selected from the Inbound Web Service portal or when drilling into the record from another page.

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

Inbound Web Service

The Inbound Web Service zone contains display-only information about selected inbound web service.

Note that the display for the base business object includes a deployment status. Refer to *Inbound Web Service Deployment* for more information about the value of this status.

Also note that once an IWS record has been deployed, certain updates to the configuration require re-deployment. Refer to the inline help and *Inbound Web Service Deployment* for more information.

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

Inbound Web Service - Log

To view the Inbound Web Service - Log page, open **Admin** > **Inbound Web Service** > **Search** select an inbound web service, and then navigate to the **Log** tab.

This page contains a standard *log zone*.

Deploying Web Services

This topic describes the configuration needed for using inbound web services.

Once an Inbound Web Service is defined it is not automatically available to the Web Service Clients to access the system. The Deployment Status and the Active flag (set to true) indicate whether a Web Service is available or not. The last step is to deploy the Inbound Web Services to the J2EE Web Application Server. This deployment phase has a number of steps that are automatically performed when a deployment is initiated:

- The Web Service files are generated and policies are attached.
- The WSDL is generated with appropriate annotations and enumerations.
- The necessary java stub code to implement the Web Service in the J2EE Web Application Server is generated and compiled.
- The Web Services are built into a valid Web Application Archive (WAR) file.
- Optionally, the newly created Web Services WAR file is deployed to the J2EE Web Application Server. This can also be done manually for clustered deployments, if desired.

There are two methods available for deploying inbound web services:

- Deployment at the command line using the **iwsdeploy[.sh]** command as outlined in the Server Administration Guide. This method is recommended for native installations and production implementations.
- Deployment using the Inbound Web Service Deployment portal. This method is recommended for development and non-production environments.

Inbound Web Service Deployment

To use the online Inbound Web Service Deployment portal, navigate using **Admin** > **Inbound Web Service Deployment** .

The following sections describe the base zones that are provided on the portal.

Deploy Inbound Web Services

This zone provides information about the last deployment. Use the **Deploy** button to deploy or re-deploy inbound web services. All inbound web services whose Active switch is Yes will be deployed. All whose active switch is No will be undeployed.

NOTE: When an Inbound Web Service is deployed, the value of its service revision field is captured. Certain changes to configuration will require re-deployment to take effect. When any of the following changes occur, the IWS service revision value is incremented. This will cause the deployment status to show **Needs Deploy**.

- Active switch is changed

- An Annotation is added or removed
- An Operation is added or removed.
- The Operation Name, Schema Type / Schema Name, Request or Response Schema, Request or Response XSL for an Operation is changed.

Deployment Status

This zone displays a list of inbound web services in the product, including the deployment status.

The deployment status is determined by comparing the internal Service Revision field on each IWS against the value captured at the time of deployment.

- **Deployed**. Indicates that the IWS has been deployed and no changes have been detected to the configuration.
- **Needs Deploy**. Indicates that the IWS has never been deployed or has been deployed but in the meantime, changes have been detected to the configuration that require redeployment.
- **Undeployed**. Indicates that the IWS is marked as inactive and the IWS is not found to be deployed at this time.
- **Needs Undeploy**. Indicates that the IWS is marked as inactive but the IWS is found to be deployed at this time.

Use the broadcast button adjacent to any of the inbound web services listed in the zone to view the details of the IWS record.

Inbound Web Service

This zone is visible if any of the inbound web services in the deployment status zone have been broadcast. It is the same zone that appears on the *Inbound Web Service — Main* portal.

Guaranteed Delivery

There are alternatives for sending messages to the system besides using inbound web services. An external system may be able to send messages to the system in a generic manner where a new web service does not need to be defined for every new type of message. These types of messages may provide a payload (the message) and the service script or business service to invoke. An example of this type of communication is a message sent from a mobile application using RESTful operations.

The external system may have no mechanism for retrying failed messages. For this situation, the product provides an algorithm that may be used to capture incoming messages that should ‘guarantee delivery’. A servlet processing this type of message may invoke the *installation algorithm* - Guaranteed Delivery, passing the details of the message and an indication if a response should be returned. The algorithm is responsible for storing the message information in a table so that it can be subsequently processed.

NOTE: The framework does not provide a sample algorithm for this plug-in spot. Your specific product may provide an algorithm and additional support for guaranteeing messages. Refer to your product documentation for more information.

Outgoing Messages

"Outgoing messages" is the term used to describe messages that are initiated by our system and sent to an external system. Messages may be sent real time or near real time. The system provides the following mechanisms for communicating messages to external systems.

- **Outbound Messages.** This method allows implementers to use configurable business objects to define the message format and to use scripts to build the message. If sent near real-time the message is posted to the outbound message table

waiting for Oracle Service Bus to poll the records, apply the XSL and route the message. If sent real time, the message dispatcher routes the message immediately.

- **Web Service Adapters.** Using a web service adapter, an implementation can consume a WSDL from an external system and create an “adapter” record that references the URL of the external system and creates appropriate request and response data areas to expose the payload information in a format understood by configuration tools. A script may then be written to build the request information and initiate a real-time web service call from within the system.
- **Send Email.** The system supplies a dedicated business service that may be used to send an email real-time from within the application.

All these methods are described in more detail in the following sections.

Outbound Messages

Outbound messages provide functionality for routing XML messages to an external system real-time or in near real time. In addition the functionality supports batching related messages to then be sent to an external system as a consolidate XML message.

For each outbound message that your implementation must initiate you define a *business object* for the outbound message maintenance object. Using the business object's schema definition, you define the fields that make up the XML source field. These are the fields that make up the basis for the XML message (prior to any XSL transformation).

Each outbound message requires the definition of its schema by creating a business object whose schema describes the information that is provided to the external system. An XSL transformation may then be performed when routing the message to an external system.

For each external system that may receive this message, you configure the appropriate message XSL and routing information.

Because the outbound message type is associated with a business object, your implementation can easily create outbound message records from a script using the **Invoke business object** *step type*. Such a script would

- Determine the appropriate *outbound message type* and *external system* based on business rules
- Access the data needed to populate the message detail
- Populate the fields in the schema and use the **Invoke business object** script step type for the outbound message type's business object to store the outbound message.
- The resulting outbound message ID is returned to the script and the scriptwriter may choose to design a subsequent step to store that ID as an audit on the record that initiated this message.

The following topics provide more information about functionality supported by outbound messages.

NOTE: For implementations that continue to use MPL and XAI, there is additional functionality related to outbound messages. Refer to *Outgoing Messages* for more information.

Polling Outbound Messages Using OSB

If the outbound message that needs to be sent to an external system can be sent as an asynchronous message (in ‘near real time’), the process initiating the outbound message should create a record in the outbound message staging table. Oracle Service Bus (OSB), is the recommended tool to use to process outbound messages in near real-time.

Outbound messages that should be processed by OSB should be configured with a processing method defined as **SOA** for the external system / outbound message type. No other information is required for defining outbound message types that are processed by OSB.

For the OSB part of the processing, the product provides a custom transport: OUAF Outbound Message that may be used by an implementation to define messages to process and how to process them.

This section provides an overview of steps required to develop OSB integrations for outbound messages created by your product.

Before developing OSB integrations, a developer should be familiar with OSB development such as creating proxy services, business services, and message flow/routing. These terms are defined as follows:

Proxy Service: In OSB, a Proxy Service is the entity that processes a given type of message and routes it to a Business Service. A separate proxy service should be defined for each type of outbound message. If a given outbound message type may be routed to different external systems, it is the responsibility of the proxy service to query the external system defined on the outbound message and invoke the appropriate business service (see below). If any transformation is required prior to routing a message to a business service, it is the proxy service's responsibility to perform the transformation.

Business Service: In OSB, a Business Service is an entity that receives a message from OSB and routes it to the appropriate destination. This should not be confused with the Business Service object provided in the product in the configuration tools.

FASTPATH: Refer to the whitepaper OSB Integration for more information.

Batch Message Processing

Your implementation may be required to send messages to the same destination as a single XML file with multiple messages include. The following points describe this logic:

- The individual messages that should be grouped together must have a processing method of **batch** on the external system / outbound message type record. The appropriate batch code that is responsible for grouping the messages must also be provided.
- A separate "consolidated message" outbound message type should be configured for the external system with a processing method of **SOA**.
- When outbound message records are created for the individual messages, the batch code and current batch run number are stamped on the record.
- When the batch process runs it is responsible for building the XML file that is a collection of the individual messages. This batch process should include the following steps:
  - Format appropriate header information for the collection of messages
  - Apply the individual message XSL to each message before including the message
  - Insert a new outbound message for the external system with the "consolidated message" outbound message type.
- The consolidated message is ready to be processed by Oracle Service Bus.

NOTE: **No process provided**. The system does not supply any sample batch job that does the above logic.

Real Time Messages

The system supports the ability to make web service calls, i.e. sending real time messages, to an external system. The configuration of real time messages is similar to the configuration of near real-time ones, with the following exceptions:

- The processing method defined for the outbound message type and an external system must be **Real-time**.
- The *message sender* defined for the outbound message type and an external system has to be set up with **Real-time** invocation type.

Just like near real-time messages, you can easily create outbound message records from a script. When a real time message is added, the system immediately routes it to the external system. If the external system provided a response message back, the system captures the response on the outbound message. If the outbound message type for the external system is

associated with a response XSL it is applied to transform the response. In this case the system captures the raw response as well on the outbound message.

Any error (that can be trapped) causes the outbound message to be in a state of **Error**. It is the responsibility of the calling process to check upon the state of the outbound message and take a programmatic action. When the outbound message state is changed back to **Pending** the message will be retried.

The base package provides a business service called "Outbound Message Dispatcher (**F1-OutmsgDispatcher**)" that further facilitates making web service calls, allowing the calling script to configure the following behavior:

- Whether or not the sent message is captured as an actual outbound message record.
- Whether or not exceptions encountered while sending the message are trapped. Trapping errors allows the calling script to interrogate any errors encountered and take some other programmatic action.

Refer to the description of the business service for a better understanding on how it works.

NOTE: HTTP Sender. In the current release of the product, only senders that communicate via HTTP are supported.

Configuring the System for Outbound Messages

The following sections describe the setup required when using *outbound messages* to communicate with an external system. The configuration walks you through the steps to configure a single external system and all its messages.

Define the Outbound Message Type

For each outbound message that must be sent to an external system, create a *business object* for the outbound message maintenance object. Using the business object's schema definition, your implementation defines the fields that make up the XML source field. These are the fields that are the basis for the XML message. XSL transformations may be applied to this XML source to produce the XML message.

Once you have your business object and schema, define an *outbound message type* for each unique outbound message.

Define the Message Sender

When messages are routed to an external system real-time using the outbound message dispatcher or using the real-time send email business service, there must be a *Message Sender*, which tells the system how to send the message. The two different mechanisms differ in how the system knows which sender to use:

- For sending emails, the default sender may be defined on the *message option* table. Alternatively, the message sender can be provided to the business service as input. Refer to *Sending Email* for more information. Senders of this type should be configured with the **RTEMAILSNDR** class.
- For other outbound messages that are routed using the real-time outbound message dispatcher, the sender to use is defined on the *external system* / outbound message type collection. Refer to *Real Time Messages* for more information. Senders of this type should be configured with the **RTHTTPSNDR** class.

Define the External System and Configure the Messages

Define an *external system* and configure the valid outgoing messages and their method of communication (processing method). Refer to *Outbound Messages* for more information.

Message Sender

Message Sender - Main

To define a new sender, open **Admin** > **Message Sender** > **Search** .

Description of Page

Enter a unique **Message Sender** and **Description**.

Use **Invocation Type** to define whether the sender is a **Real-time** sender or called by **MPL** to route near real-time messages.

> **NOTE:** The MPL invocation type remains in the product for upgrade purposes but is not recommended.

Indicate the **Message Class** for this sender. The class should be one that is defined for a sender. The real-time sender classes are **EMAILSENDER**- Email sender and **HTTPSNDR**- HTTP sender.

The following sender classes are related to MPL processing and remain in the product for upgrade purposes: **DWNSTGSNDR**- Download Staging sender, **FLATFILESNDR**- Flat file sender, **JMSSENDER**- JMS Queue sender, **STGSENDER**- Staging Upload sender, **TPCSNDR**- JMS Topic sender and **UPLDERRHNDLR**- Upload Error Handler.

Indicate whether or not this sender is currently **Active**.

Indicate whether the **MSG Encoding** is **ANSI message encoding** or **UTF-8 message encoding**.

If the Message Class is **JMSSENDER** or **TPCSNDR** indicate the appropriate **XAI JMS Connection**

> **FASTPATH:** Refer to *XAI JMS Connection* for more information.

If the Message Class is **JMSSENDER**, use the **XAI JMS Queue** to define where the response is to be sent.

> **FASTPATH:** Refer to *XAI JMS Queue* for more information.

If the Message Class is **TPCSNDR**, use the **XAI JMS Topic** to define where the response is to be sent.

> **FASTPATH:** Refer to *XAI JMS Topic* for more information.

If the Message Class for this sender is **STGSENDER** indicate the **XAI JDBC Connection**.

> **FASTPATH:** Refer to *XAI JDBC Connection* for more information.

Message Sender - Context

The sender may require context information to define additional information needed by the system to successfully send outgoing messages. Open **Admin** > **Message Sender** > **Search** and navigate to the **Context** page.

Description of Page

Define the **Context Type** and **Context Value**, which contain parameters for senders when more information is required. See below for some examples of context for different types of senders.

> **NOTE:** The values for the Context Type field are customizable using the Lookup table. This field name is SENDER_CTXT_FLG.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_SENDER*.

Email Sender Context

The email sender is used by the business service that *sends email messages real-time*.

An email sender must point to the Message Class **EMAILSENDER**. In addition, the following context records should be defined for senders of this type.

Context Type	Description
SMTP Host name	The SMTP server host name.
SMTP Username	The user ID used to access the SMTP server.
SMTP Password	The password used to access the SMTP server.
Response Time Out	The amount of time the system should wait for a real time response.

HTTP Sender

An HTTP sender is one that sends messages to an HTTP server using the HTTP protocol. HTTP senders should reference a Message Class of **HTTPSNDR**.

Various parameters are required to establish a session with the target HTTP server. You specify these parameters by defining a collection of context values for the sender. A set of context types related to HTTP variables is provided with the product. The following section describes the context types and where appropriate, indicates valid values.

Before defining the HTTP sender, you need to find out how the HTTP server on the other side expects to receive the request, and in particular, to answer the following questions:

- What is the address of the HTTP server?
- Is the HTTP server using a POST or GET HTTP method?
- If the server is using POST, how are message contents passed? Does it use an HTTP FORM or does it pass the data in the body of an XML message?

Context Type	Description	Values
HTTP URL1 - URL9	Used to construct the URL of the target HTTP server. 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. You may use substitution variables when entering values for URL parts.	
HTTP Method	The HTTP method used to send the message.	**POST** or **GET**
HTTP Proxy Host	If connecting to the remote system requires using an HTTP Proxy, then this context field can be used to configure the HTTP Proxy Host. If the Proxy Host is set, the Sender class must use the value specified to connect to the remote system via a proxy.	
HTTP Proxy Port	If connecting to the remote system requires using an HTTP Proxy, then this context field can be used to configure the HTTP Proxy	

Context Type	Description	Values
	Port. If the Proxy Port is set, the Sender class must use the value specified to connect to the remote system via a proxy. If the HTTP Proxy Host is not set, HTTP Proxy Port is ignored and the connection will be made directly to the remote system.	
HTTP Transport Method	Specifies the type of the message. You can either send the message or send and wait for a response.	**Send** or **sendReceive**
HTTP Form Data	Used when the message is in the format of an HTML Form (`Content-Type: application/x-www-form-urlencoded`). 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. The value of a form parameter takes the format of *x*=*y* where *x* is the form parameter name and *y* is its value. If *y* contains the string `@XMLMSG@` (case sensitive) then this string is replaced by the content of the service response XML message. 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. If a context record of this type is not defined for a sender, then the XML is sent with `Content-Type: text/plain`. When using POST it is put in the HTTP message body. 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. The MPL server builds formData by concatenating the individual parts. You may use substitution variables when entering values for Form Data.	
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.	

Context Type	Description	Values
HTTP Header	Sometimes the HTTP server on the other side may require the addition of HTTP headers to the message. For each HTTP header that has to be specified you should add a context record with a value having the following format *x:y* where *x* is the header name and *y* is the value for the header	
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 `;charset=x` where *x* 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 a real time response.	

Defining Outbound Message Types

Use this page to define basic information about an outbound message type. Open this page using **Admin** > **Outbound Message Type** > **Search** .

NOTE: This page is not available if the **External Message** module is *turned off*.

Description of Page

Enter a unique **Outbound Message Type** and **Description**. Use the **Detailed Description** to describe the outbound message type in detail.

Indicate the Business Object that defines business rules and the schema for outbound messages of this type.

Indicate the relative **Priority** for processing outbound message records of this type with respect to other types.

This bottom of this page contains a *tree* that shows the various objects linked to the outbound message type. You can use this tree to both view high-level information about these objects and to transfer to the respective page in which an object is maintained.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *F1_OUTMSG_TYPE*.

External Systems

Use this page to define an external system and define the configuration for communication between your system and the external system.

External System - Main

Open this page using **Admin** > **External System** > **Search** .

> **NOTE:** This page is not available if both the **External Message** and the **Open Market Interchange** modules are *turned off*.

Description of Page

Enter a unique **External System** and **Description**.

Use the field **Our Name In Their System** to specify the identity of your organization (i.e., your external system identifier) in the external system.

> **NOTE:** The workflow process profile and notification download profile are only applicable to products that support workflow and notification. They and are not visible in the product if the **Open Market Interchange** module is *turned off*.

If this external system sends inbound communications through notification upload staging, the type of workflow process that is created is controlled by the sender's **W/F (Workflow) Process Profile**.

If you send notifications to this external system, select a **Notification DL** (download) **Profile** that is used to define the configuration of the outgoing messages.

> **NOTE:** The remaining fields are not visible if the **External Message** module is *turned off*.

Set **Usage** to **Template External System** for external systems whose outbound message type configuration is inherited by other external systems.

If the outbound message type configuration should be inherited from a template external system, define the **Template External System**. If this field is specified, the outbound message type collection displays the data defined for the template system as display-only.

The **Outbound Message Type** *accordion* contains an entry for every type of outbound message defined for this external system. For each type of outbound message identify its **Outbound Message Type**.

Define the **Processing Method** for messages of this type. If the value is **XAI** or **Real-time**, indicate the appropriate **Message Sender**. If the value is **Batch**, indicate the appropriate **Batch Control**.

The **Message XSL** is the schema used to transform information from the format produced by the system to a format understood by the sender, who receives a message of this type. This is not applicable for Processing Method of **SOA**.

Enter the file name of the appropriate **W3C Schema** if you want to validate the message built for outbound messages for this external system / outbound message type prior to being routed to their destination. Refer to *Outbound Message Schema Validation* for more information. This is not applicable for Processing Method of **SOA**.

Response XSL will have the same search service as is used for the existing Message XSL field. This field will only be displayed when the processing method is **Real-time**. Refer to *Real Time Messages* for more information on how it is used.

External System - Template Use

If you are viewing an external system whose usage is a **Template External System**, use this page to view the other external systems that reference this one. Open this page using **Admin** > **External System** > **Search** and then navigate to the **Template Use** tab.

Description of Page

The tree shows every external system that references this external system as its template.

Managing Outbound Messages

Use this page to view information about outbound messages.

Outbound Message - Main

Open this page using **Menu** > **External Message** > **Outbound Message** .

Description of Page

Outbound Message ID is the system-assigned unique identifier of the outbound message. These values only appear after the outbound message is added to the database.

The **Processing Method** indicates whether this record will be processed by a **Batch** extract process, through the **XAI** tool or **Real-time**. The value defined on the external system / outbound message type collection populates this value.

When records are created with a processing method of **Batch**, the system sets Extract to **Can Be Extracted** . Change the value to **Not to be extracted** if there is some reason that processing should be held for this record.

For records with a processing method of **Batch**, **Batch Control** indicates the process that will extract this record. This value is populated based on the on the external system / outbound message type's value. **Batch Number** indicates in which batch run this record was extracted or will be extracted.

The **Retry Count** is used by the XAI tool to keep track of how many times the tool tried to process this record and could not process the record, resulting in an error.

The **Creation Date** indicates the date that this record was created.

If the processing method is **XAI**, **Status** defines the state of the outbound message record. Refer to *Lifecycle of Outbound Message* for more information.

For messages in **error** status, click **Pending** to change the status back to pending for reprocessing.

For messages in **pending**, **error**, or **in progress** status, click **Cancel** to cancel the message and prevent further processing.

Outbound Message - Message

Use this page to view the XML source used to build an outbound message. Open this page using **Menu** > **External Message** > **Outbound Message** and then navigate to the **Message** tab.

Description of Page

The **XML Source** is displayed.

If a message XSL is defined on the external system / outbound message type record linked to this outbound message, the **Show XML** button is enabled. Click this button to view the XML that is a result of applying the Message XSL to the XML source.

Outbound Message - Response

Use this page to display the XML response. Open this page using **Menu** > **External Message** > **Outbound Message** and then navigate to the **Response** tab.

Description of Page

The **XML Response** and optionally the **XML Raw Response** is displayed.

XML Response displays the response data from the system called by the real-time message. If a response XSL is defined on the external system / outbound message type record linked to this outbound message, a transform is performed and the XML Raw Response displays the original, unchanged response.

Message Option

The Message Option page defines various system settings used by the system when processing external messages.

Note that some of the options are only applicable to implementations still using the XAI and MPL servers. The settings here may be overridden by the *AdHoc Parameters section* of the XAIParameterInfo.xml or MPLParameterInfo.xml.

To define options for your environment, open **Admin** > **Message Option** .

Description of Page

The following options are supported.

Option	Description	MPL / XAI Option Name
Automatically Attempt Resend to Unavailable Sender (Y/N)	Set to **Y** if you wish to enable *Automatic Resend*. Set to **N** if you wish to log errors when the system fails to send an outgoing message.	shouldAutoResend
Default Email Sender	This is the default Message Sender used for sending e-mails when no explicit Message Sender is specified.	defaultEmailSender
Default Response Character Encoding	Determines the character encoding to be used when a response is sent. For example, you may specify **UTF-8** or **UTF-16**. If no value is specified then the default is **UTF-8**. If no special encoding should be done, then enter the value **none**.	defaultResponseEncoding
Default User	The default user is used by XAI to access your product when no other user is explicitly specified. Refer to *Server Security* for more information. Additionally, the Default User is used for MPL transactions where there is no facility to provide a User ID. For example, no facility exists to provide a user id when reading messages from a JMS Queue. In these messaging scenarios, the system will use the Default User for authorization purposes.	defaultUser
Email Attachment File Location	This is the default location of e-mail attachment files. If not specified, the e-mail service provided with the product assumes a full path is provided with each attachment file.	emailAttachmentFileLocation
Email XSL File Location	This is the default location of e-mail XSL files. If not specified, the e-mail service provided with the product assumes a full path is provided to an XSL file as part of an e-mail request.	emailXSLFileLocation
JDBC Connection Pool Max size	The MPL uses a pool of JDBC connections to connect to the database. This option determines the maximum number of JDBC connections that can be opened in the pool. The default value is 100.	JDBCConnPoolMaxSize
Maximum Errors for a Sender	This value is required if you have enabled *Automatic Resend*. It defines how many errors you receive from a sender when attempting to send an outgoing message before you mark the sender **unavailable**.	maxSendingErrors
Messages JDBC Connection	Specifies the JDBC connection that XAI uses to read the text for its messages.	messagesJDBCConnection

Messages Language	The default language to use for the messages.	language
MPL Administrator Port	The port number to be used for receiving MPL operator commands.	adminPort
MPL HTTP Server Authentication Method	This setting, along with the MPL HTTP Server User and Password are used to secure commands received by your MPL (such as those issued via *XAI Command*) through HTTP. Currently only **BASIC** authentication is supported.	MPLHTTPAuthMethod
MPL HTTP Server Password	This setting, along with the MPL HTTP Server Authentication Method and User are used to secure commands received by your MPL (such as those issued via *XAI Command*) through HTTP. The password should be in encrypted form, using the same encryption that is used for the database password. .	MPLHTTPAuthPassword
MPL HTTP Server User	This setting, along with the MPL HTTP Server Authentication Method and Password are used to secure commands received by your MPL (such as those issued via *XAI Command*) through HTTP.	MPLHTTPAuthUser
MPL Log File	The MPL Log File setting is used to specify the name of the file where MPL log information is to be written. The log contains technical information about the operation of the MPL.	MPLLogFile
MPL Trace File	The MPL Trace File setting is used to specify the name of the file where MPL trace information is to be written.	MPLTraceFile
MPL Trace Type	The MPL Trace Type is used to enable or disable tracing of the MPL. The possible values are **FULL**- All trace messages are written to the log file and **NOLOG**- No information is written to the log file.	MPLTraceType
Outbound Message Schema Location	Enter the full path of the virtual directory where valid W3C schemas are stored if your implementation wants to *validate outbound message schemas*. For example: http:/ localhost/cisxai/schemas.	xaiOuboundSchemaLoc
Privileged Users	Comma separated list of users that are allowed to specify an effective User or effective User Id via framework custom SOAP Headers.	superUsers
Records MPL Receiver Will Process At a Time	If your implementation has *configured multiple MPL servers*, indicate the number of records that each MPL receiver should process.	Not currently used
Schema Directory	The full path of the virtual directory where XML schemas are stored. For example: http:/ localhost/cisxai/schemas. If this option is not specified, the XAI uses the current directory, from where it is being run, to locate schemas.	schemaDir
Schema Validation Flag	Enter **Y** to turn on *schema validation for outbound messages*. Enter **N** to turn this off.	xaiSchemaValidationCheck
Send SOAP Fault as HTTP 500	Enter **Y** to ensure that a SOAP error is reported as an HTTP 500 "internal server error".	sendErrorAsHttp500
Sender Retry Seconds	This value is required if you have enabled *Automatic Resend*. It defines how many seconds to wait after marking a sender **unavailable** before you mark the sender **available** again (and retry sending messages to it).	senderWaitTime

System Error JDBC Connection	When a request fails to execute due to a system error, the MPL retries its execution several times. The MPL registers the system error in a table and uses this table for the retries. This setting specifies the JDBC connection required to access this table. Only enter a value in this field if it is different from the database environment used to read the XAI registry.	systemErrorTableJDBCConnection
System Error Max Retry	When a request fails to execute due to a system error, the MPL retries its execution several times until a maximum number of retries is reached. This option specifies the maximum number of retries.	systemErrorMaxRetries
System Error Retry Interval	When a request fails to execute due to a system error, the MPL retries its execution several times. This option specifies the number of seconds the MPL server waits between retries.	systemErrorRetryInterval
Thread Pool Initial Size	The MPL uses a thread of pools to enhance performance. The MPL starts with a minimum number of threads and grows/shrinks the pool based on the MPL system load. This option specifies the initial number of threads in the thread pool. The minimum number of threads is 12.	threadPoolInitialSize
Thread Pool Max Size	This option specifies the maximum number of threads in the thread pool.	threadPoolMaxSize
Thread Pool Non Activity Time	This option specifies how long a thread in the pool may be inactive before it is timed out and released from the pool.	poolNoneActivityTime
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
WSDL Service Address Location	Specifies the SOAP address location that XAI uses in creating a WSDL. If no value is present, the XAI's URL is used.	wsdlAddressLocation
XAI Authentication Password	The multi-purpose listener uses this field in combination with the **XAI Authentication User** when attempting to communicate with the XAI server over HTTP, which is running on a secured servlet and requires authentication.	HTTPBasicAuthPassword
XAI Authentication User	The multi-purpose listener uses this field in combination with the **XAI Authentication Password** when attempting to communicate with the XAI server over HTTP, which is running on a secured servlet and requires authentication.	HTTPBasicAuthUser
XAI Trace File	The full path name for the file, where the XML messages should be written. For example: c:\inetpub\wwwroot\cisxai\xai.log.	traceFile
XAI Trace Type	Use this option to specify the level of logging. The possible values are **FULL**- All XML messages are written to the log file and	traceType

	NOLOG- No information is written to the log file. **FASTPATH:** Refer to *Server Trace* for more information about tracing.	
XSL Directory	The full path of the virtual directory where XSL transformation scripts are located. XSL transformation scripts can be defined for each service. By default, this is the same directory as the schemas directory.	XSLDir

Where Used

Used by the XAI tool to obtain various required settings and locations.

Web Service Adapters

The base product provides a configuration object called Web Service Adapter that is used to help build configuration objects to allow for functionality in the system to initiate a web service call from within the system. A Web Service Adapter provides the following functionality:

- WSDL (web service description language) import. An implementer can use the WSDL import functionality to read the details of a WSDL into the system
- Internal API generation. The system generates internal data areas that have two main purposes: they provide the API for custom code to define the appropriate input and they provide output data for the web service call using Oracle Utilities Application Framework schema language. In addition, the web service dispatcher uses element mapping defined in the data areas to transform the internal XML into the structure expected by the external system as described in the WSDL.
- Defines the URL needed to perform the web service call at runtime.

Understanding Web Service Adapters

The following topics describe the system functionality in more detail.

Importing a WSDL

Configuring a Web Service Adapter starts by identifying the WSDL (the web service description language document used to define the interface) that will be provided by the external system. The following steps describe the base product functionality provided to allow a user to import a WSDL.

- Navigate to the **Web Service Adapter** page in add mode and select the appropriate base business object.
- Enter a meaningful Web Service Name and appropriate descriptions.
- Provide the URL of the given WSDL.
- Click **Import** to retrieve the details of the WSDL. The system then parses the WSDL details and populates the WSDL Service Name, WSDL Source, WSDL Port, URL and a list of Operations (methods) defined in the WSDL.
- Determine which Operations should be **active** based on the business requirements for invoking this web service. **Active** operations are those that the implementation is planning to invoke from the system. These require appropriate request and response data areas generated for them. The following section provides more information about that.
- Specify the appropriate Security Type to configure the type of security to use when invoking this web service.
- Click Save.

At this point, a web service adapter record is created in pending status. The next step is to generate the request and response data areas for the operations configured as active.

Generating Request and Response Data Areas

Each **active** operation for the web service adapter requires a pair of data areas, request and response, that represent the request and response XML messages for the operation.

The base product provides steps to generate the data areas as follows:

- As described in the Importing a WSDL section above, the operations listed in the WSDL are generated for the web service adapter and the implementer should indicate which operation to activate.
- After saving the **pending** web service adapter, the display lists all the active operations and for each one includes a **Generate** button.
- After clicking **Generate** for an operation, a window appears where the names of the new Request and Response Data Areas may be defined. Click **Save** to generate the data areas.

The generated data areas provide the API for the implementer to use when implementing the web service call in an appropriate algorithm or service in the system. The data areas contain the appropriate mapping from the elements that the implementer works within the code that invokes the web services and the WSDL definitions.

To facilitate generating the request and response data areas, the base product invokes a special business service used to create the appropriate mapping. The business service is defined as a BO option on the Web Service Adapter business object. This allows an implementation to provide a custom business service to further enhance the request and response mapping where appropriate.

NOTE:
Generated data areas. It is possible to edit and modify the generated data areas after they are created. An implementer can change element names or remove unneeded elements if desired. Manually changing the generated data areas must be done only when absolutely necessary. This is because the system is not able to validate manual changes and issues with the data areas would only be detected at run time.

Activating Web Service Adapters

The business objects provided by the base package for web service adapters include a simple lifecycle of **Pending** and **Active**. Configure the web service adapter and its data areas while in **Pending** status and activate it when it is ready to be implemented in the appropriate system functionality.

Invoking Web Services

To make a call to a web service using a web service adapter, the system has provided a Web Service Dispatcher business service (**F1-InvokeWebService**) to submit a web service call. The calling program is responsible for retrieving all the information to correctly populate the request data required by the web service call before invoking the business service.

NOTE:
Refer to the detailed description of the business service for more information.

Limitations

The following points highlight limitations associated with the types of web services that the system supports:

- It is possible for one WSDL document to contain definitions for several web services. The system currently supports only one port or service per WSDL document.
- It is possible for a WSDL to support multiple message patterns. The system currently supports only request / response.

Setting Up Web Service Adapters

Use the Web Service Adapter portal to define the configuration needed to communicate with an external system using a web service call. Open this page using **Admin** > **Web Service Adapter** > **Search** .

This topic describes the base-package zones that appear on the Web Service Adapter portal.

Web Service Adapter Query

Use the query portal to search for an existing web service adapter. Once a web service adapter is selected, you are brought to the maintenance portal to view and maintain the selected record.

Web Service Adapter Portal

This page appears when viewing the detail of a specific web service adapter.

Web Service Adapter - Main

The Web Service Adapter main page contains a zone that includes display-only information about the web service along with the actions that are available for maintaining a web service adapter.

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.

Web Service Adapter - Log

This page contains a standard log zone.

Sending Email

The framework provides the ability to initiate an email from within the system. The following topics highlight the functionality available.

- Sending email “real time”. The framework provides a business service **F1–EmailService** that supports sending an email. The schema supports elements for all the information required to create an email. It also supports sending attachments. The SMTP information (host, user name and password) may be provided or may be defined on a message sender, that may be provided as input. Review the business service schema for information about the input elements.

  > **NOTE:** Validating attachments. If a Validate Email Attachment algorithm is plugged into the *installation record*, it is called to validate the attachments supplied, if applicable.

- Staging records to be emailed in “near real time”. This option may be used by options that are creating emails in bulk. This technique uses the outbound message table to capture the details of the email message.

XML Application Integration

This section describes the XML Application Integration (XAI) utility, which enables you to configure your system to receive information from and to send information to external applications using Extensible Markup Language (XML) documents.

NOTE: The XAI functionality is legacy functionality and not recommended for new implementations. The topics for the functionality outlined in the previous sections describe the recommended features for supporting sending and receiving external messages. The XAI information remains in the documentation for upgrade purposes.

The Big Picture of XAI

The XML Application Integration (XAI) module provides the tools and infrastructure that businesses require for integrating their applications with your product. The integration your product with other systems across organizational boundaries or business is made possible, regardless of the platforms or operating system used. XAI provides an integration platform that enables the following:

- Integrate with Customer Relationship Management (CRM) systems
- Provide information feeds for web based customer portals
- Fit seamlessly with web based applications
- Facilitate fast implementation of batch interfaces
- Integrate with other XML compliant enterprise applications

XAI exposes system business objects as a set of XML based web services. The service can be invoked via different methods, e.g., Hypertext Transfer Protocol (HTTP) or Java Message Service (JMS). Consequently, any application or tool that can send and receive XML documents can now access the rich set of system business objects. Business-to-Business (B2B) or Business-to-Consumer (B2C) integration with other enterprise applications as well as the setup of web portals is made very simple and straightforward.

XAI Architecture

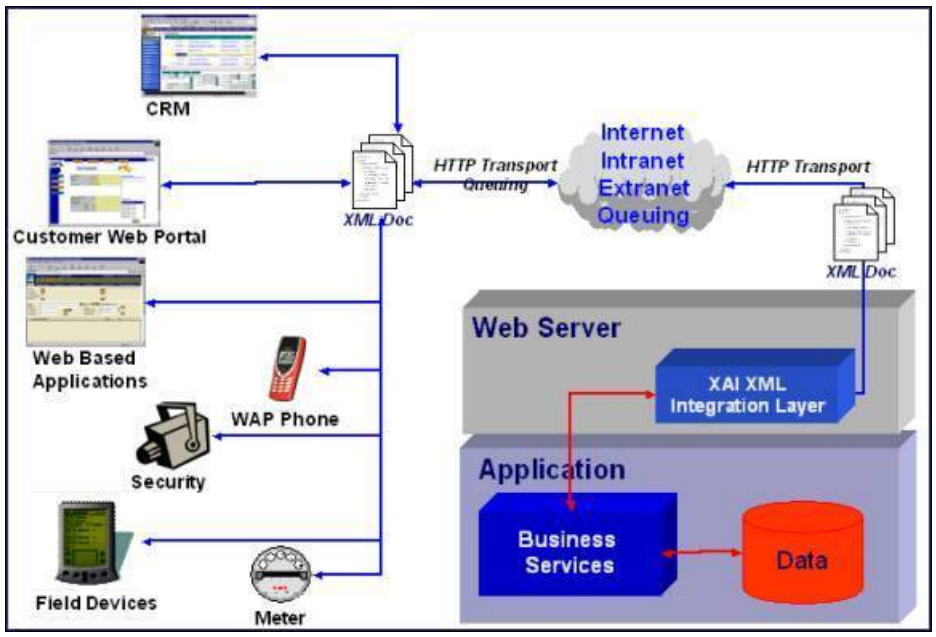
The XAI architecture contains 3 major components:

- The Core Server Component
- The Multi Purpose Listener (MPL)
- The Client Component

The Core Server Component

The core server component is responsible for receiving XML requests, executing the service and returning the response to the requester.

The following diagram shows the XAI tool operating on a web server and providing integration between the system business objects and various external applications.



The core is built in Java, using a layered, scalable architecture. The basic transport protocol supported by the core is SOAP/HTTP.

FASTPATH: Refer to *SOAP* for more information.

The XAI core server provides a Java servlet to process requests through HTTP. You may also use other messaging mechanisms such as message queuing, publish/subscribe or a staging table. The multi-purpose listener processes the messages.

The Multi Purpose Listener (MPL)

NOTE: Multi Purpose Listener functionality is legacy functionality that is not recommended going forward. The Oracle Service Bus (OSB) is the recommended tool. This section remains in place for upgrade purposes.

The Multi Purpose Listener (MPL) is a multi-threaded Java server that constantly reads XML requests from various external and internal data sources, such as a Java Message Service (JMS) message queue, a JMS topic or system staging tables.

The MPL can be used to process inbound messages (those sent by an external application to invoke a system service), or outgoing messages (those sent by your product to external applications). The MPL uses different receivers to process messages from different data sources.

A receiver is implemented using 3 distinct layers:

- The Receiving Layer
- The Execution Layer
- The Response Layer

The Receiving Layer

This layer deals with polling various locations to determine if new records, files or incoming requests exist. The various locations include:

- Staging tables, including *XAI staging control*, *XAI upload staging*, notification download staging (certain products only), and *outbound message*.
- An external directory that contains a file, for example a comma delimited file or an XML file.
- A JMS queue/topic.

A separate receiver is defined to read requests from each of these locations. When the MPL server starts, it looks for all defined active receivers in the *XAI Receiver* table, and for each receiver it starts a thread that constantly fetches messages from the message source.

Once a request message is read, it is passed to an execution thread that implements the execution layer. Each receiver references an *Executer* that is responsible for executing the request.

Configuring Multiple MPL Servers

A single MPL server may only run one of each of the above staging table receivers for a given JDBC connection. To enhance the performance of the processing of the staging tables, you may define multiple MPL servers where each one runs the active receivers defined in the receiver table.

To ensure that each staging table receiver processes its own set of records in the staging table, the receiver selects a set number of records (specified as *Message option* **Number of Records an MPL Receiver Will Process At a Time**) and marks those records with the IP address and port number of the MPL.

The Execution Layer

The execution layer sends the XML request to the *XAI core server* and waits for a response.

NOTE: Currently the only type of *executer* supported is the XAI servlet running either on an XAI server or locally under the MPL. However the architecture allows for executing a request on other execution environments.

The executer is invoked and is passed in an *XAI Inbound Service* that specifies an XML request schema and an *adapter*. Adapters tell XAI what to do with a request. The adapters point to a specific Java class that renders a service.

For example you can configure an Adapter to invoke any published application object (by pointing it to the appropriate java class). This adapter accesses system objects through the page service. You can think of an adapter as a plug-in component.

Once the executer processes the request and a response is received, it is transferred to the next layer, the *response* layer.

The Response Layer

The response layer is responsible for "responding" to the execution. The responses are handled by invoking an appropriate *sender* defined on the receiver's response information. Each sender defined in the system knows how to process its response. For example:

- The JMS queue sender and the JMS topic sender post responses to the appropriate queue / topic.
- The *staging control sender* handles errors received during the execution of the staging control request.
- The *upload staging sender* updates the status of the upload staging record based on the success or failure of the staging upload request.
- The download staging sender is a bit unusual because it is helping to build the message being sent (Oracle Customer Care and Billing only).

NOTE: There are some cases where a response is not applicable. For example, the file scan receiver creates a staging control record to process a file that exists in a directory. There is no "response" applicable for executing this request.

The XAI Client Component

The XAI Client component is the set of online control tables and tools used to manage your XAI environment.

The ***Schema Editor*** is a tool used to create and maintain XAI schemas.

> **FASTPATH:** Refer to *XAI Schema* for more information.

The **Registry** is a term used to refer to all the tables required to "register" a service in the system. It includes the *XAI Inbound Service* and a set of control tables defining various options.

The ***Trace Viewer*** is installed with your XAI client tools and is used to view traces created on the XAI server.

XML Background Topics

The following section introduces some background information related to XML.

XAI Schemas

> **NOTE: Business Adapter.** The **BusinessAdapter** adapter supports communication to configuration tool objects: *business objects*, *business services* and *service scripts* through their own schema API. When communicating to these objects, it is not necessary to create XAI schemas for the schemas associated with the objects. As a result, there is no need to use the XAI schema editor when defining *XAI Inbound Services* for this adapter.

At the core of XAI are XML web services based on XML schemas. XML schemas define the structure of XML documents and are the infrastructure of e-business solutions. They are used to bridge business applications and enable transaction automation for e-commerce applications. Industry standard schemas document common vocabularies and grammars, enhancing collaboration and standardization. Validating XML processors utilize XML schemas to ensure that the right information is made available to the right user or application.

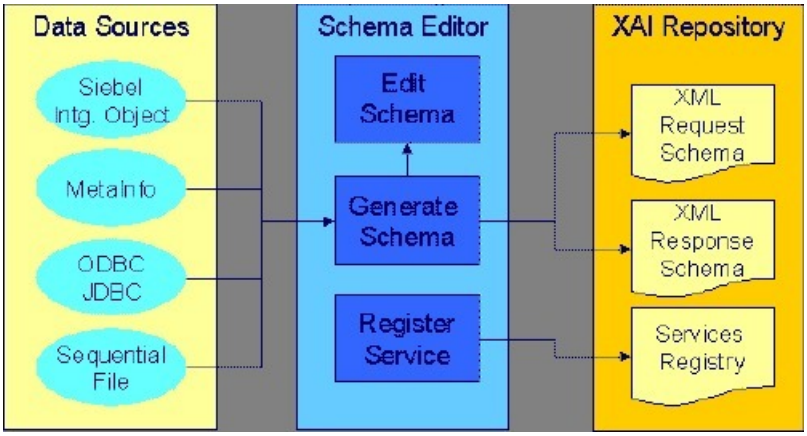
The system exposes its application objects as XML schemas that define the interface to system services. Every service (e.g., CreatePerson or AccountFinancialHistory) is defined using a pair of schemas: the Request Schema and the Response Schema. The request and response schema can be identical.

The Request Schema defines the XML document structure describing the "inputs" for a service.

The Response Schema defines the XML document structure describing the "outputs" of a service.

The Schema Editor

To facilitate the process of exposing business objects as XML schemas, we use the Schema Editor, a graphical tool to create, import and maintain schemas. The Schema Editor provides automated wizards to import schemas residing in existing data structures and documents. The Schema Editor can import schemas from the following sources: system business objects, ODBC data sources, sequential files.



Before the XAI tool can use a service, it must be registered or published.

FASTPATH: Refer to *Schema Editor* for more information.

XSL Transformations

XSL Transformations (XSLT) is a language used to transform an XML document into another XML document or another document format such as HTML. It is part of the Extensible Stylesheet Language (XSL) family of recommendations for defining XML document transformation and presentation. It is defined by the World Wide Web Consortium (W3C) and is widely accepted as the standard for XML transformations. Several tools are available on the market to generate XSLT scripts to transform an XML document defined by a source schema to an XML document defined by a target schema.

In XAI you can use XSL to:

- Transform an inbound message into the structure required by the XAI request schema for that service.
- Transform the response to an inbound message into a format defined by a schema provided by the requesting application.

  FASTPATH: Refer to *XAI Inbound Service* for more information.

- Transform an outgoing message before it is sent out.

  FASTPATH: Refer to *XAI Route Type* for more information.

- Transform data from an external source before it is loaded into the staging upload table.

  FASTPATH: Refer to *XAI Inbound Service Staging* for more information.

SOAP

SOAP stands for Simple Object Access Protocol. The SOAP "Envelope" is the wrapping element of the whole request document that may be used by messages going through the XAI tool.

The following diagram shows a simple XML request using the SOAP standard.

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Header>
    <CorrelationID>1234</CorrelationID>
    <SOAPActionVersion>1.2</SOAPActionVersion>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <CISAccount transactionType='Read'>
      <CISAccountService>
        <CISAccountHeader
          AccountID='1234'
        />
      </CISAccountService>
    </CISAccount>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

XPATH

The XML Path Language (XPath) is an expression language used by XSLT to access or refer to parts of an XML document. It is part of the XSL recommendations maintained by the W3C. XPath provides the syntax for performing basic queries upon your XML document data. It works by utilizing the ability to work with XML documents as hierarchically structured data sets.

In the following XML document, some examples of XPath references are:

- authors/author/name
- authors/*/name
- authors/author[nationality]/name
- authors/author[nationality='Russian']/name
- authors/author[@period="classical"]

```
<?xml version='1.0'?>
<authors>
  <author period="classical">
    <name>Sophocles</name>
    <nationality>Greek</nationality>
  </author>
  <author>
    <name>Leo Tolstoy</name>
    <nationality>Russian</nationality>
  </author>
  <author period="classical">
    <name>Plato</name>
    <nationality>Greek</nationality>
  </author>
</authors>
```

In the XAI tool, XPath is used to construct outgoing messages.

Server Security

XAI server security supports the basic HTTP authentication mechanism as well as web service security (WS-Security) to authenticate the user requesting service. When authenticating using WS-Security, the SOAP header contains the authenticating information.

The base package provides two XAI server URLs, one that uses basic HTTP authentication ('/classicxai') and another that supports both methods ('/xaiserver'). Regardless of which authentication method is practiced, it is the latter you should

expose as your main XAI server. The main XAI servlet gathers authentication information from the incoming request (HTTP or SOAP header) and further calls the internal ("classic") servlet for processing.

The "classic" XAI server security uses the basic HTTP authentication mechanism to authenticate the user requesting service. It assumes the requester has been authenticated on the Web server running the XAI servlet using the standard HTTP (or HTTPS) basic authentication mechanism. The authenticated user-id is passed to the application server, which is responsible for enforcing application security. This requires the system administrator to enable basic authentication for the Web server running the XAI servlet. To enable HTTP basic authentication, the XAI server '/classicxai' should be defined as a url-pattern in the web resource collection in the web.xml file. When the XAI server is not enabled for basic authentication, it transfers the user-id specified on the **Default User** *Message option* to the application server.

By default, the system would always attempt to further authenticate using SOAP header information. This is true even if the request has already been authenticated via the Web server. Use the **Enforce SOAP Authentication** *Message Option* to override this behavior so that a request that has been authenticated already by the Web server does not require further authentication by the system.

If SOAP authentication information is not provided, the system attempts to authenticate this time using information on the HTTP header. You can force the system to solely use SOAP authentication using the **Attempt Classic Authentication** *Message Option*.

Currently the system only supports the standard *Username Token Profile* SOAP authentication method where "Username", "Password" and "Encoding" information is used to authenticate the sender's credentials. The following is an example of a *Username Token Profile* in a SOAP header:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "urn:schemas-xmlsoap-org:envelope">
<SOAP-ENV:Header xmlns:wsse="http://www.w3.org/2001/XMLSchema-instance">
<wsse:Security>
<wsse:UsernameToken>
<wsse:Username>MYUSERID</wsse:Username>
<wsse:Password Type="PasswordText">MYPASSWORD</wsse:Password>
</wsse:UsernameToken>
</wsse:Security>
<SOAPActionVersion>2.0.0</SOAPActionVersion>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
...
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

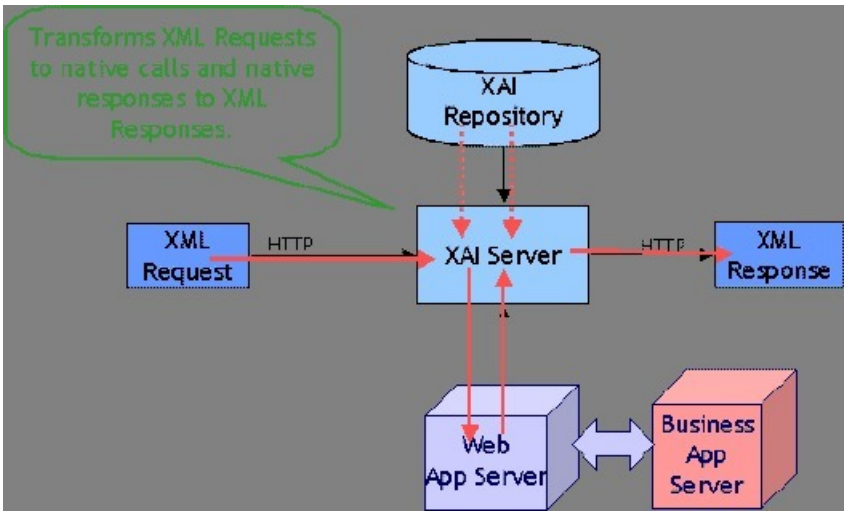
By default both user and password are authenticated. You can use the **System Authentication Profile** *Message Option* to change this.

NOTE: Custom authentication. You can override the base package user credentials authentication logic using the **System Authentication Class** *Message Option*.

Inbound Messages

Inbound messages are XML messages sent by an external application to invoke a system service. Inbound messages can be sent one at a time or in batches of several messages in a file. Third party integration points can import web services for inbound service calls. The standard method of doing this is by publishing a WSDL (Web Service Definition Language) definition for each exposed service.

Synchronous Messages



Synchronous service requests are sent using the HTTP protocol to the XAI servlet, which runs under a web application server such as WebLogic.

- The service request XML document must conform to the request schema that defines the service.
- The XAI servlet on the web server receives the service request XML document and based on the service name in the document identifies the appropriate XAI Inbound Service. Once the service is identified, the XAI servlet accesses the XAI Inbound Service record to find out the properties of the service.
- Based on the service properties, the XAI module loads the request and response schemas from the schemas repository (and caches them in memory).
- Based on the Adapter referenced by the service, it calls the appropriate adapter. The adapter performs most of the work to service the request.
  - The adapter connects to the application server, based on the connection information in the registry control tables.
  - It then parses the request document using the request schema.
  - Once the request document has been validated, the adapter converts the XML request document into a call to the application server.
- The response returned by the application server is then converted into a service response XML document, based on the response schema.
- The XML response document is shipped back to the caller over HTTP.

Using HTTP for Synchronous Service Execution

Invoking a service is not much different from sending a regular HTTP request. Here the HTTP request contains the XML as a parameter. The XAI server handling requests via the HTTP protocol is implemented using a Java servlet running on the web server.

Microsoft Visual Basic/C Example

Microsoft provides an easy way to send XML requests over HTTP. To send and receive XML data over HTTP, use the Microsoft XMLHTTP object

set xmlhttp = createObject("Microsoft.XMLHTTP")

xmlhttp.Open "POST", http://Localhost:6000/xaiserver, false

xmlhttp.Send XMLRequest

XMLResponse = xmlhttp.ResponseText

Here **http://localhost:6000/xaiserver** is the URL of the XAI server. **XMLRequest** contains the XML request to be processed by XAI and **XMLResponse** contains the XML response document created by the XAI runtime.

Java Example

Java provides a very simple way to send a request over HTTP. The following example shows how a request can be sent to XAI from an application running under the same WebLogic server as the one XAI runs. In this example, we use the "dom4j " interface to parse the XML document.

import com.splwg.xai.external.*;

import org.dom4j.*;

String xml;

xml = "<XML request>";

XAIHTTPCallForWebLogic httpCall = new XAIHTTPCallForWebLogic();

String httpResponse = httpCall.callXAIServer(xml);

Document document = DocumentHelper.parseText(httpResponse);

Asynchronous Messages

Various types of *receivers* running under the MPL server (rather than the XAI server) handle asynchronous inbound messages from several sources.

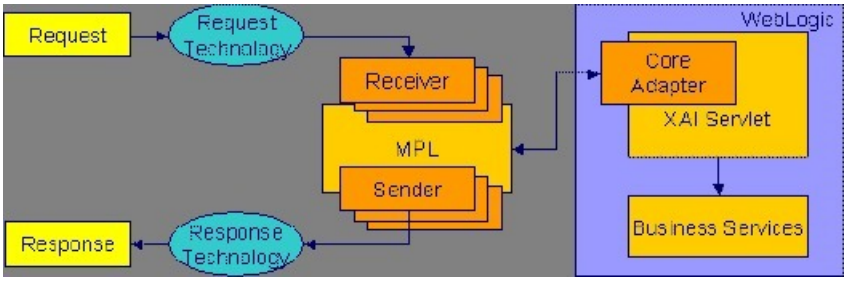
Requests may be received via the following:

- MQSeries, TIBCO or any JMS compatible messaging system
- The XAI Upload Staging table

The response is returned to the JMS queue/topic or to the staging table.

> **FASTPATH:** Refer to *Designing XAI Receivers* for more information about the different receivers provided by the product for the different data sources.

The following diagram shows the flow for asynchronous inbound messages.

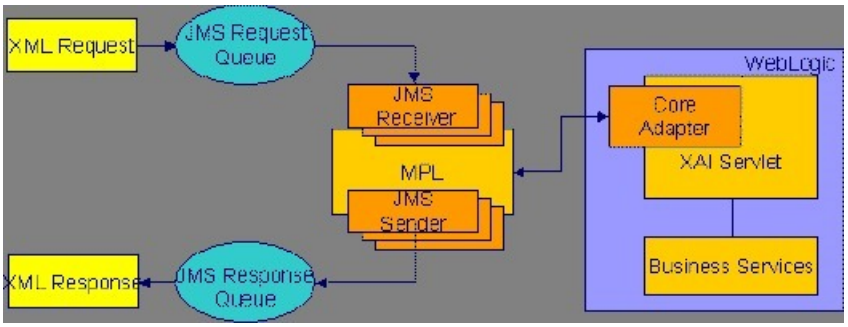


Using JMS

Java Message Services (JMS) is a standard Java 2 Platform, Enterprise Edition (J2EE) protocol to send/receive asynchronous messages using a queue or topic message protocol.

XML messages may be received and sent via JMS using either a JMS Queue or JMS Topic. In order to access a JMS provider such as MQSeries, TIBCO or WebLogic, the MPL must first connect to the appropriate server using a JMS Connection.

The following diagram depicts a message sent and received through a JMS Queue.



Using JMS Queues

JMS Queues are used to receive and send messages using the message queue protocol. Products that support this protocol include MQSeries.

The following describes events that take place when a JMS queue is used:

- The requester places the XML request message on a JMS queue. The request contains both the XML message and the name for the *reply queue*.
- The *JMS Queue Receiver* waits for messages on that queue. When the message arrives it is selected from the queue and executed using the adapter for the requested service.
- The XML response message is placed on the reply queue specified in the request. It is the requester's responsibility to fetch the response message from the queue.

The MPL uses a *JNDI server* to locate the queue resource.

FASTPATH: Refer to *Designing XAI JMS Queues* for more information.

Using JMS Topics

JMS Topics are used to send and receive messages using the publish/subscribe messaging model. Products that support this protocol include TIBCO, MQSeries and WebLogic.

The MPL uses a *JNDI server* to locate the topic resource.

- Define a *JMS Topic receiver*, listening to a predefined JMS Topic.
- The other application builds an XML message based on the schema defined for that service.
- It sends the XML request to the predefined topic and specifies the reply topic name.
- The MPL reads the message from the Topic, executes the service and returns the response to the reply topic specified in the inbound message.

FASTPATH: Refer to *Designing XAI JMS Topics* for more information.

Staging Upload

The system provides a staging table, where an interface can store XML requests to perform a service in the system.

Some external systems interfacing with the system are not able to produce XML request messages. Or you may have external systems that produce XML messages but the messages are sent in a batch rather than real time. The system provides the capability to read an external data source containing multiple records, map the data to an XML request and store the request on the *XAI upload staging* table. These records may be in XML requests, sequential input files or database tables.

The XAI upload staging table may be populated in one of the following ways:

- When a collection of messages in a file or database table must be uploaded, a *staging control* record should be created for each file/database table. The *Staging Control Receiver* processes each file or database table and creates records in the XAI upload staging table for each message.
- The *XML File Receiver* creates records directly in the XAI upload staging table. Note, in addition, the XML File Receiver creates a staging control record to group together these records.
- XAI creates records in the XAI upload staging table for *inbound messages in error* that are configured to post the error.
- It is possible that when a response to a notification download staging message is received (Oracle Customer Care and Billing only), the response requires some sort of action. If this is the case, the system creates an XAI upload staging record (and an XAI staging control record) for the response.

Staging Upload Receiver

Once the XML requests are in the staging table, the *Staging Upload Receiver* reads the requests from the XAI upload staging table and invokes the XAI server (via the executer) with the appropriate XAI inbound service. Inbound service records typically point to the core *adapter* used to invoke system services.

Staging Upload Sender

The staging upload *sender* handles "responses" to the execution of the message in XAI upload staging. If the execution is successful, the sender updates the status of the upload staging record to **complete**. If the execution is unsuccessful, the sender updates the status to **error** and creates a record in the *XAI upload exception* table.

NOTE: Configuration required. The above explanation assumes that you have correctly configured your upload staging receiver to reference the upload staging sender. Refer to *Designing Responses for a Receiver* for more information.

Staging Control

The *staging control* table is used to indicate to XAI that there is a file or table with a collection of records to be uploaded. The special *Staging Control Receiver* periodically reads the staging control table to process new records.

The XAI staging control table may be populated in one of the following ways:

- To process a specific sequential input file, manually create a staging control record and indicate the location and name of the file and the XAI inbound service to use for processing. Use this mechanism to process ad-hoc uploads of files.
- If a file is received periodically, you may define a *File Scan Receiver*, which periodically checks a given file directory for new files. The file scan receiver creates a new staging control record to process this file.
- The *XML File Receiver* processes a file containing a collection of XML messages to be uploaded. The XML file receiver creates a staging control record and creates records directly into the XML upload table. The staging control record is automatically set to a status of **Complete** and is used to group together the XML upload records. Refer to *Processing Staging Upload Records for a Staging Control* for more information.
- To upload records from a database, manually create a staging control record and indicate an appropriate XAI inbound service, which contains the information needed by the system to access the appropriate table. Use this mechanism to process ad-hoc uploads of files.

NOTE: To upload records from a database table, you must create a staging control record. There is no receiver that periodically looks for records in a database table.

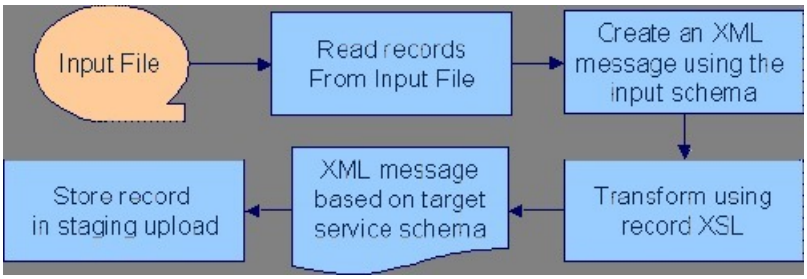
- To upload records from a Lightweight Directory Access Protocol (LDAP) store, use the *LDAP Import* page to select users or groups to process. Uploading records from this page causes a staging control record to be created. In addition, you may manually create a staging control record to upload data from an LDAP store and indicate the appropriate file to import. Refer to *How Does LDAP Import Work* for more information.
- Whenever an *XAI upload staging* record is created, an XAI staging control record is created as well.

FASTPATH: Refer to *Batch scenarios* for more information about configuring the system to populate the staging control with requests from external files.

Staging Control Receiver

The Staging Control *Receiver* processes staging control records and invokes XAI (via the executer) to execute the request. The executer uses the appropriate adapter to generate records in the *XAI Upload Staging* table - one for reach record in the file or table.

The diagram below illustrates the information used by the staging control receiver to load data onto the staging table from a sequential input file.



The staging control *adapter* does the actual work. It reads the individual records in the input file and applies the XSL transformation script indicated on the XAI inbound service record to the input data to produce an XML request in the XAI upload staging table.

Processing Staging Upload Records for a Staging Control

In some cases, a process may populate records directly into the XML staging upload table. An example of such a process is the *XML File Receiver*. In this case, a staging control record is also created and used to group together the staging upload records.

The staging control contains information needed to process a group of staging upload records:

- The user related to these records. This user is for application security purposes. The user indicated here must have the proper rights for the application service and transaction type to be executed by XML upload records processed for this staging control.
- An indication of whether or not the records should be processed *sequentially*.

Processing Staging Records in Sequential Order

In some cases, a collection of messages uploaded together in a file must be processed in the order the messages are received. For example, if messages to add a person and add an account for this person are received together, the message to add the person must be processed before the message to add the account.

If messages received in a file must be processed sequentially, turn on the Sequential Execution switch on the *staging control* record. When the staging control receiver creates records in the XML upload table, the identifier of each record is built as a concatenation of the staging control record and a sequential number. If your staging control record indicates that the XML upload records should be processed in sequential order, the records are processed in primary ID order.

NOTE: Non-sequential processing. If your staging control does not indicate that the related XML records should be processed in sequential order, the records are processed by the staging control receiver using a random, multi-threaded mechanism.

If you have defined a *receiver* to periodically search for files and populate records in the staging control table, you may turn on the Sequential Execution switch on the receiver. This ensures that records processed as a result of this receiver are executed in sequential order.

Staging Control Parameters

If your staging control accesses the data from a database table, you have the capability of defining the selection criteria. *XAI inbound services* that reference the **CISStagingUpload** adapter may contain a collection of fields that are used in an SQL WHERE clause. When adding a new *XAI Staging Control* record, you can define the values for the WHERE clause.

For example, imagine that you have a work management system where new premises are defined. Rather than waiting for this system to "push" new data to you, you design the interface to have the system "pull" the new data by looking directly at the "new property" database table.

The Request XML for your XAI service contains SQL statements used to access the data. You could define a Staging Control Parameter of "Add Date". When creating your staging control, you may enter a parameter value of today's date. When this record is processed, it only retrieves new properties from this work management table whose Add Date is today.

The following example shows part of the Request XML schema for an XAI Service that SELECTs premises based on postal code.

```
- <Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-
  com:datatypes" xmlns:d="urn:schemas-microsoft-com:datatypes" xmlns:b="urn:schemas-
  microsoft-com:BizTalkServer" b:root_reference="TestDuplicatePremise">
 - <ElementType name="TestDuplicatePremise" content="eltOnly">
    <b:property name="adapter">CISStagingUpload</b:property>
    <b:property name="stagingUploadType">DB</b:property>
    <element type="Request" />
    <element type="Response" />
   </ElementType>
 - <ElementType name="Request" content="eltOnly">
    <AttributeType name="Postal" d:type="string" d:maxLength="12" />
    <b:RecordInfo />
    <attribute type="Postal" />
   </ElementType>
 - <ElementType name="Response" content="eltOnly">
    <b:RecordInfo />
    <element type="Premises" />
   </ElementType>
 - <ElementType name="Premises">
    <b:RecordInfo />
    <b:property name="SQL">SELECT PREM_TYPE_CD, KEY_SW, OK_TO_ENTER_SW,
      TREND_AREA_CD, ADDRESS1, MAIL_ADDR_SW, POSTAL FROM CISADM.CI_PREM WHERE
      POSTAL = @TestDuplicatePremise/Request/Postal</b:property>
    <b:property name="tableName">CISADM.CI_PREM</b:property>
  - <AttributeType name="PREM_TYPE_CD" d:type="string" d:maxLength="008">
      <b:property name="dataType">string</b:property>
      <b:property name="uniqueId">PREM_TYPE_CD</b:property>
      <b:FieldInfo />
    </AttributeType>
  - <AttributeType name="KEY_SW" d:type="string" d:maxLength="001">
      <b:property name="dataType">string</b:property>
      <b:property name="uniqueId">KEY_SW</b:property>
```

The postal code value to substitute into the WHERE clause is defined on the individual XML Staging Control records.

FASTPATH: Refer to *XAI Staging Control* for more information.

Staging Control Sender

Before the staging control receiver invokes the executer, it changes its status to **complete**, assuming that there will be no problems. If the executer detects an error condition, the staging control sender updates the status of the staging control record to **error** and removes any XAI upload staging records that may have been created.

NOTE: Configuration required. The above explanation assumes that you have correctly configured your staging control receiver to reference the staging control sender. Refer to *Designing Responses for a Receiver* for more information.

Inbound Message Error Handling

For messages that are processed using the *staging upload* table, application errors that prevent XAI from successfully processing the message cause the staging record to be marked in error and highlighted via a To Do entry.

For messages that are not processed via staging upload, your implementation should consider what should happen to application errors. If the origin of the message is able to handle an immediate error returned by XAI, then no special configuration is needed. An example of this is an HTTP call to our system where the originator of the message is waiting for a real-time response.

Otherwise, for messages where errors should not be returned to the originator, but should be highlighted in this system for resolution, be sure to mark the **Post Error** switch on the *XAI inbound service*. When this switch is turned on and an application error is received by XAI when processing the message an *XAI upload staging* record is created (along with a staging control record) and marked in **error**.

For example, if a message is received via a JMS queue, application errors that prevent XAI from processing the message should not be returned to the queue because there is no logic to route the error to the sending system.

Integration Scenarios

Integration Using an EAI (or Hub)

It is possible for your various systems to be integrated with each other using a hub. The hub is implemented using an Enterprise Application Integration (EAI) tool provided by a third party vendor. Most hubs support HTTP and/or JMS and can work with XML schemas or document type definitions (DTDs).

- XAI services are presented to the EAI tool as schemas or as DTDs, immediately making the system callable from the hub.
- Integration scripts or workflow processes are defined in the EAI tools.
- At run time the hub uses HTTP or JMS to access the system using inbound messages.
- Outgoing messages are used to notify the hub about events occurring in the system. The messages are sent using HTTP or JMS.

Batch Scenarios

Messages may be sent in batch files, or may be retrieved from a database. In all cases, the system needs to be able to read the file and identify each individual message in order to create an XML request that can be processed by the XAI server. Once each individual message is identified, a request is stored on the XAI Staging Upload table for later execution.

XML Message File

It is possible for you to receive a file containing a collection of XML messages. The system identifies each separate message within the file and creates an entry for each message on the XAI upload staging table. It also creates a staging control record to group together each newly created XAI upload record. This staging control is created in **Complete** status and is not processed by the staging control receiver.

Since external applications may send messages in a format unknown to XAI, the system needs a mechanism for identifying the messages and mapping them to an XAI service.

XAI Groups
First the system associates the entire XML file with an XAI Group. You can think of the XAI Group as a categorization of the collection of messages. For example, you may have a separate XAI Group for each third party who sends you a collection of XML messages.

The system uses an *XPath* and XPath value to identify the correct XAI Group for the XML file.

Attachments and Rules
After identifying the appropriate group to which an XML file belongs, the system takes each message in the file and applies the appropriate XSL transformation to the message to produce a record on the upload staging table.

To process the messages in a file, the system needs to know how to identify each message in a file containing multiple messages. A file may use the same root element for each message or different root elements for different types of messages. For each XAI group, you must indicate the root element(s) that identifies a message by defining one or more attachments. Each attachment defines a root element, which tells the system when a new message begins.

Once the system has identified each separate message in the file, it must determine the correct XSL transformation script to apply. Once again the system uses an *XPath* and XPath value to identify the correct XSL to apply. For each XAI group, you define one XAI rule for every possible type of message you may receive in the file. Each XAI rule defines an XPath, XPath value and XSL transformation script.

Note you may assign a priority to each of your rules. The rules for more common messages may be assigned a higher priority. This enhances performance by ensuring that rules for more common messages are processed before rules for less common messages.

NOTE: Include parent elements. If the XML message includes parent elements (such as a transmission id or a date/time) that are needed for any of the separate child messages that are posted to the upload staging table, you can configure the appropriate attachment to include **parent** elements.

FASTPATH: Refer to *Designing an XML File Receiver* for more information about defining receivers that process XML files.

Sequential Input File

You may receive messages in a sequential input file, such as a comma-delimited file.

The following steps should be performed when configuring the system to enable data to be uploaded from an input file into the staging upload table:

1. Create an XML Schema that describes the records in the sequential input file.
2. Create the XSLT transformation that maps a record in the input file to an XML service request in your product.
3. Create an *XAI service* representing the batch process that loads the input file into the staging table.
4. If desired, create a new file scan receiver, which waits for an input file to appear in a particular directory. (If you do not take this step, then you need to create a *staging control* when you want a file to be processed.)

NOTE: Character Encoding. If the file is encoded with a specific character encoding, you may indicate the encoding as part of the file name to be uploaded. If the file name ends with **?enc=?x**, where x is the file character encoding, the adapter processes the file accordingly. For example, the file name may be specified as **premiseUpload.csv?enc=?UTF-8**. If the encoding is not specified as part of the file name and the file is in UTF-16 or UTF-8 with byte order mark, then the adapter can recognize the encoding.

Database File

It is possible for you to define an interface where inbound messages are retrieved by reading records in a database table.

The following steps should be performed when configuring the system to enable data to be uploaded from a database table into the staging upload table:

- Create an XML Schema that describes the records in the database table.
- Create the XSLT transformation that maps a record in the database table to an XML service request in your product.
- Create an *XAI service* representing the process that loads the records from the database table into the staging table.

LDAP Import

Refer to *Importing Users and Groups* for information about using XAI to import user and user group definitions from a Lightweight Directory Access Protocol (LDAP) store.

WSDL Catalog

Web Service Definition Language (WSDL) is a language for describing how to interface with XML-based services. It acts as a "user's manual" for Web services, defining how service providers and requesters communicate with each other about Web services.

The base package provides the ability to publish a WSDL definition for each service exposed as an *XAI Inbound Service*. In addition, it is possible to request a catalog of all the XAI Inbound Services and a link to each WSDL. To view the catalog, launch a new browser session and enter the URL **http://$host:$port/XAIApp/xaiserver?WSDL** where $host and $port are replaced by the appropriate values for the current environment.

Outgoing Messages

This section describes outgoing message functionality related to MPL, which is no longer recommended.

- Outbound messages. As described in *Outbound Messages*, outbound messages are supported for sending outgoing messages. This functionality includes support that is only related to MPL.
- Notification download staging (NDS) messages. This method is only supported by *Oracle Utilities Customer Care and Billing*. Using this method near real-time, a record is written to the NDS staging table referencing only key fields. MPL then polls the records, invokes a service to build the message, applies the XSL and routes the message. If sent real-time, no record is posted to the staging table but rather the message dispatcher routes the message immediately. Refer to Oracle Utilities Customer Care and Billing help, Workflow and Notifications, Notification and XAI for more information.

The following sections describe the outbound messages topics specific to MPL in more detail.

Outbound Message Receiver

The outbound message *receiver* processes records in the outbound message table that have a processing method flag equal to **XAI** and a status of **pending** and changes the status to **in progress**. The receiver then retrieves the message XSL and the message sender defined for the external system / outbound message type.

> **NOTE: Template external system**. If the outbound message's external system references a template external system, the outbound message type configuration for the template external system is used.

It applies the message XSL (if supplied). If the option to *validate outbound message schemas* is turned on, the schema validation is performed.

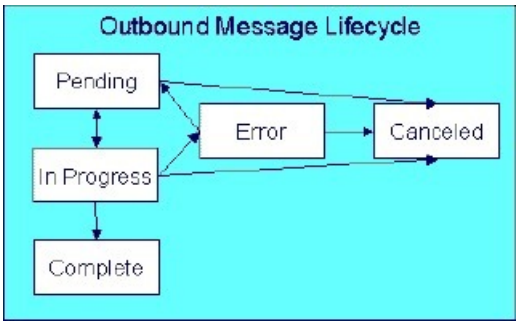
Refer to *Outbound Message Error Handling* for information about error handling.

If no errors are received, control is turned over to the *outbound message sender* for routing.

> **NOTE: Initialization of receiver**. During the initialization of this receiver (for example if there is a problem with MPL and it is restarted) any records that are found to be **in progress** are changed to **pending** so that those messages are sent properly.

Lifecycle of Outbound Message

The outbound message receiver processes outbound message records based on their status. The following diagram describes the lifecycle of an **XAI** type outbound message.



- Records are created in **pending** status.
- The outbound message receiver processes pending records and changes the status to **in progress**.
- If the message is sent successfully the system changes the status to **complete**.
- If there was a problem sending the message the system changes the status to **error**.
- When the user resolves the error they can change the status back to **pending**.
- A user can change the status of a **pending** or **error** record to **canceled**.
- For the rare cases where there is a problem with MPL and a message is left in the status **in progress**, users may manually change the status to **canceled**. In addition the outbound message receiver includes a step at startup to find **in progress** messages and change them to **pending**.

Outbound Message Sender

The outbound message sender is responsible for routing the message to the *Message sender* determined by the receiver. If the routing is successful the outbound message status is marked **complete**. If the routing is unsuccessful, the status is marked in **error**.

Refer to *Outbound Message Error Handling* for information about error handling.

NOTE: Automatic Resend. If you have configured the system for *automatic resend* and the system detects that the error is due to the sender being unavailable, the message remains in **pending** status.

NOTE: Configuration required. The above explanation assumes that you have correctly configured your outbound message receiver to reference the outbound message sender. Refer to *Designing Responses for a Receiver* for more information.

Outbound Message Error Handling

If the outbound message receiver or the outbound message sender detects an error while attempting to process the outbound message, it marks the message in **error**, captures the error message and its parameter values and creates a To Do entry using the To Do type specified in the Message option **To Do Type for Outbound Message Errors**.

A separate background process *F1-DTDOM* is responsible for completing To Do entries for outbound messages no longer in **Error**.

Outbound Message Schema Validation

The outbound messages that are generated by the system should be well formed and valid so that they do not cause any issues in the external system. To ensure this validity you may configure the system to validate messages before they are routed to their destination.

- Define a directory where the valid W3C schemas are located using the Message option **Outbound Message Schema Location**
- Each *external system* message must indicate the appropriate W3C schema to validate against

You may turn on or off this validation checking using an Message option **Schema Validation Flag**.

Automatic Resend

If a system error is received by the MPL when attempting to route the message to a sender, (using the outbound message method or the NDS message method), the system marks the appropriate table in **error**. This is true even if the reason for the error is that the connection to the sender is unavailable. When the connection is restored, a user must change the status of the appropriate record to **pending** (for outbound messages) or **retry** (for NDS messages) in order for the message to be resent.

Alternatively, you can configure your system to attempt to automatically resend the message. This section describes the logic available for auto resend. To enable automatic resend, you must set the flag **Automatically Attempt Resending to Unavailable Senders** on *Message option* appropriately.

If an error is received by the MPL when it attempts to invoke a sender and the auto resend option is on, the system does not mark the record in **error**. It continues to attempt sending messages to the sender until the number of errors has reached a predefined maximum error number (defined as an *Message option*). When the maximum is reached, the sender is marked as **unavailable** and an MPL log entry is created. The MPL ignores messages for **unavailable** senders.

The system tries to resend messages to this sender the moment the sender is reset to be **available**. The following points describe how a sender becomes **available**:

- MPL attempts to retry sending messages to unavailable senders every X minutes, where X is defined on *Message option*.
- On MPL startup all senders are marked as **available**
- A user may navigate to the *XAI Command* page and issue a command **MPL Refresh Sender** to refresh the cached information for a particular sender

Designing Your XAI Environment

This section guides you through the steps required to design the tables that control your XAI processing.

Installation

The XAI server is installed with default configuration. This section describes how you may customize the XAI server configuration.

Startup parameters are defined in two parameters files

- The XAIParameterInfo.xml file is used by the XAI HTTP server. This file is found within the XAI directory in the path (...\splapp\xai).
- The MPLParameterInfo.xml file is used by the MPL server. This file is found within the XAI directory in the path (... \splapp\mpl).

Both files store the parameters as XML files with the following elements (sections):

- Source
- ParameterVariables
- AdHocParameters

The XAI Source Section

The XAI tool accesses XAI *registry* information through the standard system programs. The `<Source>` section in the `XAIParameterInfo.xml` file tells XAI the user ID for accessing the registry information. It contains the following attributes:

Attribute Name	Description
Source Type	This should be set to **CorDaptix**, which tells XAI to access the registry through the standard access to system programs.
CorDaptixUser	The user ID to use when accessing the registry data.

The MPL Source Section

The `<Source>` section in the `MPLParameterInfo.xml` file defines the database connection information used to connect to the database storing the XAI table information. It contains the following attributes:

Attribute Name	Description
Source Type	Defines the source of the data, for example **ORACLE** or **DB2**.
jdbcURL	The URL used to connect to the product database. For example: **jdbc:oracle:thin:@//server-name:1234/DBNAME**
databaseUser	The Oracle User Id used to connect to the database. For example: `sysuser`

databaseUserPassword	The Oracle password used to connect to the database. For example: `sysuserpassword`

The Parameter Variables Section

When defining values for fields in certain control tables in the registry, you may reference substitution variables that point to the `<ParameterVariables>` section of the installation files. Substitution variables provide for dynamic substitution of values based on parameters provided at server startup.

To specify a substitution parameter in a string value you enter the name of the substitution parameter enclosed with `@`.

For example if you have a field in the XAI control tables that should contain the URL for the XAI HTTP servlet, you could enter the value in the following way: **http://@HOST@:@PORT@/xaiserver**.

In the parameters section, define the appropriate values for these parameters, for example:

```
<ParameterVariables>
<ParameterVariable name="HOST" value="localhost" />
<ParameterVariable name="PORT" value="8001" />
</ParameterVariables>
```

At run time, the system builds the URL as `http://localhost:8001/xaiserver`.

Every substitution parameter is defined using an `<ParameterVariable>` element with the following attributes:

Attribute Name	Description
Name	The name of the substitution parameter
Value	The value to replace an occurrence of the substitution parameter in an XAI control table field

NOTE: Substitution parameters can only be used for string fields, and not for fields that are foreign keys to other objects.

The AdHoc Parameters Section

The <AdHocParameters> section is used to provide registry definitions that override the existing ones. Unlike the <ParameterVariables> section, a whole registry object definition can be specified in this section. When the XAI server starts, it first reads the registry definitions from the database and then it reads the <AdhocParameters> section. If it finds an object definition in this section, it uses it to replace the one read from the database.

Attribute Name	Description
Object Name	The object name may be one of the following objects: Option Receiver Sender
Object Attributes	The attributes of the object. Each object type has its own set of attributes:
'Option' object attributes	
name	The option flag. Must be defined in the OPTION_FLG table
value	The value for that option
'Receiver' object attributes	
name	The receiver ID
Class	The JMS provider. May be 'MQ'
TargetClient	The client type writing/reading to the JMS queue/topic. May be 'JMS' or 'MQ'. Only relevant for interfacing with MQSeries

JMSProvider	The JMS provider. May be 'MQ'
TargetClient	The client type writing/reading to the JMS queue/topic. May be 'JMS' or 'MQ'. Only relevant for interfacing with MQSeries
Executer	The XAI Executer ID for this receiver
'Sender' object attributes	
Class	The JMS provider. May be 'MQ'
JMSProvider	The JMS provider. May be 'MQ'
TargetClient	The client type writing/reading to the JMS queue/topic. May be 'JMS' or 'MQ'. Only relevant for interfacing with MQSeries

NOTE: LDAP Import Mapping. If your organization integrates with an LDAP store, you must define your LDAP import mapping and create a reference to this mapping in the ad-hoc parameters section. Refer to *Including Your LDAP Import Mapping in the XML Parameter Information File* for more information.

Designing XAI Inbound Services

When designing your XAI environment, you should first identify the services that you would like to perform. Determining your services facilitates your design for the other registry options.

To design your inbound services,

- Determine each service that needs to be performed
- Determine the correct *adapter* that is needed by your service.
- Determine the required layout of the request and response messages and specify the request *schema* and response *schema*.
- If a transformation of the data is required, you need to design the appropriate request XSL and response *XSL transformation* scripts.
- If the service references the staging upload adapter, determine the staging file type and design the record XSL transformation script. In addition, determine if you want to enter an input file name and interface name. Finally, determine whether or not you need to indicate a special *JDBC connection*.
- As new releases of the system are installed, it may be necessary to modify your service for the new release. If this is the case, you need to design separate versions of the inbound service.

FASTPATH: Refer to *XAI Inbound Services* for more information.

Designing XML Schemas

You need XML schemas for the services you designed in *Designing XAI Inbound Services*.

For each message, identify what service you need to invoke and what action you need to perform. If you have multiple actions that you may need to perform for the same service, you may choose to create a single generic XML schema or you may choose to create multiple schemas, which are more specific. For generic messages, the transaction type, indicating the action to perform would be passed in on the XML request document to indicate what must be done. For more specific messages, you may be able to indicate the transaction type directly on the schema and it would not need to be overwritten at run time.

You need to create a response schema for each request schema. It is possible for you to use the same schema for both functions.

FASTPATH: Refer to *Schema Editor* for more information.

Designing XSL Transformations

You need an XSL transformation script for each service you designed in *Designing XAI Inbound Services*, where you determined a transformation is necessary. In addition, you need XSLT scripts for your outgoing messages. Each sender, which receives a message, probably requires a transformation of the message into a local format. Refer to *Outgoing Messages* for more information.

For each message requiring transformation, determine the format used by the external system. In most cases, it is not the same format recognized by the system. For each case, you must create an XSL transformation, which maps the message format from the external format to one expected by your product or from your product format to one expected by the external system.

When identifying the required XSL transformations, remember to take into consideration the data that is processed by the staging control table. This service reads data stored in a file or database table and uses the Record XSL to map the individual records to an individual service request.

FASTPATH: Refer to *XAI Inbound Service* for more information.

Designing Your Registry Options

The XAI registry is a set of control tables that is used to store service definitions as well as various system information required by the XAI and MPL servers. The following sections describe each table in the registry.

Designing XAI JNDI Servers

The XAI tool, including receivers and senders, uses a Java Name Directory Interface (JNDI) server to locate resources on the network. JDBC connections, JMS connections, JMS queues and JMS topics should be defined on the JNDI server. In addition, adapters that need to access your product information reference the JNDI server to determine where your product is running.

Design your JNDI server values as follows:

- Define a JNDI server that indicates where the system is running.
- If you are using JMS resources, such as JMS Queue or JMS Topic, and these are located in a server other than the one where the system is running, define the server where these resources are located.
- If you are using *LDAP Import*, define a JNDI server that points to the LDAP server.
- If there are any other resources that may be needed by XAI, define a JNDI server to indicate where these are located.

The product is shipped with a JNDI server running under the same Weblogic server used for the system.

When defining the URL, you may use substitution parameters such as those shown in the example below. XAI uses the parameter variables section of your start up parameters to build the appropriate URL. Refer to *Installation* for more information

JNDI Server	Description	Provider URL
WLJNDI	Weblogic JNDI Server	t3://@WLHOST@:@WLPORT@
ACTDIR	Active Directory Server	ldap://@LDAPHOST@:@LDAPPORT@

Designing XAI JDBC Connections

If you need to access a database table to process your messages, XAI needs to know the location of the database and how to access it. If the tables are located in the same database used for the system (defined in your *Installation*), then you do not need to enter any extra JDBC Connections. If you need to access data that lives in another database, design the additional JDBC Connections and determine the type of connection and connection information.

FASTPATH: Refer to *XAI JDBC Connections* for more information about defining XAI JDBC Connections.

Designing XAI JMS Connections

If you are using JMS to send and receive messages, then you must define a JMS Connection to indicate the JNDI server to use to locate these resources. For each JMS connection defined, the *MPL* server creates a pool of connections that are later shared by multiple threads.

FASTPATH: Refer to *XAI JMS Connections* for more information about defining XAI JMS Connections.

Designing XAI JMS Queues

If your business uses JMS Queues to send and receive messages, then you need to add an entry on the *XAI JMS Queue* page, defining the JNDI server and Queue Name.

Designing XAI JMS Topics

If your business uses JMS Topics to send and receive messages, then you need to add an entry on the XAI JMS Topic page, defining the JNDI server and Topic Name.

Designing XAI Formats

The Formats section of the registry is used to define data formats. Data formats can be used in schema definitions to specify data transformations. To determine what data formats you need to define for your XAI environment, you must review the expected format of data that you will be exchanging and determine whether or not data transformation is required.

The following sections describe the four different types of formats and some guidelines in their use.

Date Formats

Date formats may be specified using any valid Java format supported by the java.text.SimpleDateFormat class.

To specify the time format use a time pattern string. For patterns, all ASCII letters are reserved. The following usage is defined:

Symbol	Meaning	Presentation	Example
G	era designator	Text	AD
y	year	Number	1996
M	month in year	Text & Number	July & 07
d	day in month	Number	10
h	hour in am/pm (1~12)	Number	12
H	hour in day (0~23)	Number	0

m	minute in hour	Number	30
s	second in minute	Number	55
S	millisecond	Number	978
E	day in week	Text	Tuesday
D	day in year	Number	189
F	day of week in month	Number	2 (2 nd Wed in July)
w	week in year	Number	27
W	week in month	Number	2
a	am/pm marker	Text	PM
k	hour in day (1~24)	Number	24
K	hour in am/pm (0~11)	Number	0
z	time zone	Text	Pacific Standard Time
'	escape for text	Delimiter	
''	single quote	Literal	'

Currency Formats

Currency formats are used to specify formatting for elements representing currencies. They may include the following:

Symbol	Meaning
#	number place holder
,	thousands separator
.	decimal point
$	currency sign

For example to define the currency format for US dollar, indicate: $#,#.00

Phone Formats

Phone formats can be used to specify formats for telephone numbers. The supported format specification is limited to the following format characters:

Symbol	Meaning
0	number place holder
\0	0

Any other character appearing in the formatting expression is a placeholder for that character. To specify the '0' character, use '\0'.

Phone Format Example: (000) 000-0000

Text Formats

Text formats are used to specify formats for character string attributes. The following expressions are supported:

Symbol	Meaning
\cUpperCase	Translate the string to upper case.
\cLowerCase	Translate the string to lower case.
\cProperCase	Translate the string to proper case. The first character of every word is translated to uppercase. The remaining characters are translated to lowercase.

FASTPATH: Refer to *XAI Format* to define your XAI Formats.

Designing XAI Adapters

The product provides a set of adapters to process your XML requests. The adapters point to a specific Java class that renders a service. If you find that you need to use a protocol, which is not supported by the adapters provided, you will need to add a new Message Class (which points to a Java class) and a new XAI Adapter. It is recommended that your implementers contact customer support. The following adapter classes are provided.

- **BASEADA:** This is the core adapter class that provides access to any published system service. This adapter accesses system objects through the page server. Services with this adapter need to indicate the object (application service), which should be invoked.
- **BUSINESSADA**: This is the core adapter class that provides access to schema-based objects. This adapter accesses *business objects*, *business services* and *service scripts* through their schema API. Services with this adapter need to indicate the schema of the object, which should be invoked. When communicating to these objects, it is not necessary to create XAI schemas for the schemas associated with the objects. XAI is able to directly communicate with these objects using their existing schema definitions. As a result, there is no need to use the XAI schema editor when defining XAI Inbound Services for this new adapter.
- **STGUPADA**: This staging upload adapter class is used when an extra step is required prior to using a service with the core adapter. For example, perhaps you need to read a file, which is not in XML format, and convert it to an XML format recognized by the system. Services with this type of adapter do not need to indicate an application service but must indicate information about the file to be converted. Refer to *XAI Staging Control* for more information.
- **LDAPIMPRTADA:** This is a special adapter that is used when importing Lightweight Directory Access Protocol (LDAP) objects into the system. If your organization uses LDAP, you can import your existing LDAP users and groups into your product, so that you do not need to define them twice. The adaptor can process search, validate and import XML requests. Refer to *Importing Users and Groups* for more information.
- **XAICMNDADA:** This is an internal adapter. It is used to send commands to the XAI Server.
- **SIEBELADA:** This adapter is no longer supported.

Designing XAI Executers

The executer is responsible for executing messages received through a message receiver. The product provides an executer, which uses the XAI server; however the architecture allows for implementing additional execution classes. If you require a different executer and therefore a different execution class, it is recommended that your implementers contact customer support.

FASTPATH: Refer to *XAI Executer* for more information.

Designing Message Senders

This section only describes message sender functionality that is only related to the XAI/MPL processing. Refer to *Define the Message Sender* for details about other types of senders that are supported independent of XAI/MPL.

Message senders are responsible for define outgoing message destinations and for " *responding*" to the XAI executer.

- For NDS messages, the sender to use is defined on the *XAI route type* for the notification download profile.
- For outbound messages, the sender to use is defined on the *external system* / outbound message type collection.
- For responding to the XAI executer, the sender to use is defined on the *receiver*.

For each sender, you must reference an appropriate Message Class. The information in this section describes the sender classes that are provided with the system.

You must create senders to "respond" to the various staging table receivers in the system.

- Create a sender to be used for "responses" to messages processed by the staging control receiver. You should create one sender, which points to the Message Class **UPLDERRHNDLR**. Refer to *Staging Control Sender* for more information about this sender.
- Create a sender to be used for "responses" to messages processed by the upload staging receiver. You should create one sender, which points to the Message Class **STGSENDER**. Refer to *Staging Upload Sender* for more information about this sender.
- Create a sender to be used for messages processed by the download staging receiver. You should create one sender, which points to the Message Class **DWNSTGSNDR**. (DWNSTGSNDR is not supported in all products.)
- Create a sender to be used for messages processed by the outbound message receiver. You should create one sender, which points to the Message Class **OUTMSGSNDR**. Refer to *Outbound Message Sender* for more information about this sender.

Next, design the senders for "responses" to other receivers, for example the JMS queue receiver or JMS topic receiver. The system provides message classes to use for these senders. Use the class **JMSSENDER** for a JMS queue sender and **TPCSNDR** for a JMS topic sender.

Finally, review all your *outgoing messages* and determine the mechanism for communicating with the target system for each message.

- For all senders that are used for *real time messages*, define a context entry with a context type of **Response Time Out** to define the amount of time the system should wait for a real time response.
- An HTTP sender is one that sends messages to an HTTP server using the HTTP protocol. For an HTTP sender, reference a message class of **HTTPSNDR**. In addition, the context described in *Message Sender — Context* for the **RTHTTPSNDR** apply to this sender as well.
- An email sender allows for XML messages to be sent as email messages through an SMTP server. It can be used in notification download processes to send a response as an email message. The email sender supports standard email functionality such as "CCs" and attachments. The content of the email message is controlled by the XSL script defined in the *XAI route type* of the NDS message. The XSL script has access to all context records of the NDS message as well as the input XAI message that was created by processing the NDS. Reference a message class of **EMAILSENDER**. In addition, the context described in *Message Sender — Context* for the **RTHTTPSNDR** apply to this sender as well.
- If you want XML messages to be written to a flat file, use a flat file sender. For example, it can be used in the notification download process to write a response message to a flat file. Flat file senders should reference an Message Class of **FLATFILESNDR**. In addition, the following context records should be defined for senders of this type.

Context Type	Description	Values
Flat file output directory	Directory in the file system where to write the file	
Flat file filename pattern	The name of the output file. The file name may be a literal constant, or generated dynamically. To create a dynamic filename use <file name>$$ID, where $$ID is replaced at run time by the ID of the NDS message that triggered the response message. If no file name is defined for the sender, the XAI server generates a file name with the following format 'XAI$$ID.xai'.	
Append data to file	This parameter controls whether the content of the response message is appended to an existing file, or a new file is created (possibly replacing an existing one).	**YES** or **NO**

Context Type	Description	Values
Character Encoding	Indicates if the message should be sent with character encoding. The sender will write the content of the file with encoding specified in the context value. If no value is specified, the sender uses the default Java system encoding, which is determined according to the operating system locale.	**UTF-8** or **UTF-16**

The topics below describe configuration required for senders that route a message via an HTTP sender, a flat file sender or an email sender.

Designing XAI Groups

XAI groups are used by the system to process an XML file containing multiple messages to be uploaded into the system. One or more groups may be defined for an *XML file receiver*.

FASTPATH: Refer to *XML Message File* for more information about how groups are used to process an XML file.

When setting up your XAI environment, identify the interfaces that require uploading an XML file containing multiple XML messages into the system through XAI.

First you need to categorize the XML files that you may receive. Define an XAI Group for each logical categorization. For example, you may want to define a separate XAI Group for each third party who may send you a collection of XML messages. Or, if all third party service providers send direct access messages in a standard format, you may want to define a single XAI Group for direct access messages.

For each group, you need to identify the root elements that indicate when a new message is starting. This collection of unique root elements for a group is called the attachments.

For each group, you must identify every possible message that may be sent. For every message, define an XAI Rule. The rule indicates the XSL transformation script to be executed along with the XPath and XPath value that the system uses to identify each message.

FASTPATH: Refer to *XAI Group* to define groups, their attachments and their rules.

Designing XAI Receivers

Receivers define small pieces of code that wait for requests to be received through various sources. Each receiver references an Message Class where the small piece of code is defined. The following receiver classes are provided:

- **STGRCVR** The receiver that references this class polls the XAI upload staging table for new inbound requests.
- **STGCTLRCVR:** The receiver that references this class polls the XAI staging control table for new upload processes.
- **DWNSTGRCVR :** The receiver that references this class polls the notification download staging table (NDS) for new messages. (Not available in all products).
- **OUTMSGRCVR :** The receiver that references this class polls the outbound message table for new messages.
- **JMSRCVR**: Receivers that reference this class receive requests through a message queue that supports the JMS Queue interface, such as IBM MQSeries.
- **TPCRCVR:** Receivers that reference this class receive requests through a publish/subscribe model, such as TIBCO, or any system supporting the JMS Topic interface.
- **FILESCANRCVR:** Receivers that reference this class poll a given directory for files with a given file name pattern.

- **XMLFILERCVR:** Receivers that reference this class poll a given directory for XML files with a given file name pattern.

Multiple receivers may be defined for these receiver classes. For example, the XML file receiver defines the scan directory. If you have multiple directories that contain files to be uploaded, define a receiver for each directory.

All types of receivers reference an Message Class and XAI Executer. If you require a new Message Class or Executer because you use a protocol that is not currently supported, it is recommended that your implementers contact customer support.

Designing Responses for a Receiver

Once a request has been sent for execution to the XAI server (via the executer), *the response layer* processes the response. For some receivers, a response may not be applicable. For example, a file scan receiver reads flat files in a given directory and posts records to the XAI staging control table. Responses are not applicable for this type of receiver.

The response may be conditional on the outcome of the request and may be sent to more than one destination (*sender*). To design your receiver responses, determine the conditions under which a response should be sent for each request processed by each receiver:

- Never send a response
- Send a response if the request was successful
- Send a response if the request was unsuccessful due to a system error
- Send a response if the request was unsuccessful due to an application error

Once you determine when to send a response, you must determine where to send the response. Responses for different conditions may be sent to different Message Senders or to the same Message Sender.

Designing Receivers that Poll Staging Tables

The following receivers are needed to poll the various system staging tables.

- Create a *staging upload receiver* that references the Message Class **STGRCVR**. For responses, **All Events** should reference the *staging upload sender*.
- Create a *staging control receiver* that references the Message Class **STGCTLRCVR**. For responses, **All Events** should reference the *staging control sender*.
- If your implementation uses the NDS message method of communicating outgoing messages, create a staging download receiver that references the Message Class **DWNSTGRCVR**. For responses, **All Events** should reference the download staging sender. NDS messaging is not supported in all products.
- If your implementation uses the *outbound message* method of communicating outgoing messages, create an *outbound message receiver* that references the Message Class **OUTMSGRCVR**. For responses, **All Events** should reference the *outbound message sender*.

For all the above receivers, if you need to access multiple environments, simply create receivers for each JDBC connection. Note that you should not add more than one of each of the above receivers pointing to the same JDBC connection. To improve performance for a single JDBC connection you may *configure multiple MPL servers*.

NOTE: Sample Data for Initial Install. When first installing the system, records for each of the above receivers are provided. Your implementation may use these records or remove them and create your own.

Designing a JMS Queue Receiver

If you need to receive messages through a JMS compatible queue, you need to define a JMS Queue receiver. When designing a JMS Queue receiver you first need to design a *JMS Connection* and a *JMS Queue*.

If you would like to post *responses* back to the JMS queue, you may create an *Message sender* to send the response to the JMS queue.

Designing a JMS Topic Receiver

If you need to receive messages through a JMS Topic using the publish/subscribe model, you need to define a JMS Topic receiver, which receives messages published under a specific topic. When designing a JMS Topic receiver you first need to design a *JMS Connection* and a *JMS Topic*.

If you would like to post *responses* through a JMS Topic using the publish/subscribe model, you may create an *Message sender* to send the response to the JMS topic.

Designing a File Scan Receiver

The file scan receiver constantly looks in a given directory for files with a given pattern. When it finds a matching file, it creates a record in the *staging control* table to upload the contents of the file into the *upload staging* table.

When setting up your XAI environment, identify the interfaces that require uploading a file from a directory into the system through XAI. For each unique file, define a file scan receiver. For each receiver record, indicate the Scan Directory, where new files will be placed, the Scan File, which is the naming pattern to look for and the XAI Inbound Service to use for mapping the data into a system service.

In addition, if you want to specify a character encoding, the following Context record should be defined.

Context Type	Description	Values
Character Encoding	Indicates that the message is character encoded. When the receiver creates a staging control entry for a file, it will add ?enc=?x to the name of the file in the table where *x* is the value of this parameter. Refer to *Sequential Input File* for more information.	**UTF-8** or **UTF-16**

Designing an XML File Receiver

The XML file receiver constantly looks in a given directory for XML files with a given pattern. When it finds a matching file, it goes through steps to identify each separate message in the file, determine the appropriate XSL transformation and create a record in the staging upload table.

FASTPATH: Refer to *XML Message File* for more information.

When setting up your XAI environment, identify the interfaces, which require uploading an XML file containing multiple XML messages into the system through XAI. For each unique file, define an XML file receiver. For each receiver record, indicate the Scan Directory, where new files will be placed, the Scan File, which is the naming pattern to look for and the collection of XAI groups. XAI groups are used by the system to identify each separate message in the file and to determine the appropriate XSL transformation for each message.

FASTPATH: Refer to *Designing XAI Groups* for more information.

Configuring the System for NDS Messages

Refer to the documentation for your product to find out if NDS messaging is supported.

Schema Editor

The Schema Editor is a Graphical User Interface (GUI) tool to create XML schemas. The tool provides wizards to generate schemas from various sources.

Opening the Schema Editor

After launching the schema editor, you are asked to connect to a database. On the Connect dialog:

- Select an ODBC data source pointing to the desired database.
- Enter the data source, user ID, password and database owner required to log on to that database.
- Click **Connect**.

After connecting, the schema editor appears.

Use the File/Open dialogue to select a schema from the schema directory. Refer to *The Options Menu* for information about setting the default schema directory.

> **NOTE:** When opening a schema, the schema editor validates the schema and any errors are displayed. Refer to *Validating a schema* for more information.

Schema Editor Window

The schema editor allows you to modify individual elements and attributes of a given schema.

Description of Page

Refer to *System Wide Functions for Schema Editor* for information about the various menu options available for the schema editor.

Service Name Enter the name of the service to be created in the service name text box. This is the name of the first element under the Body element in the XML document.

> **CAUTION:** Important! When adding new schemas, carefully consider the naming convention for the Service Name. Refer to *System Data Naming Convention* for more information.

Adapter The adapter used to process services using this schema.

Internal Service Name If the schema is for an adapter that should invoke a system service, this is the internal name of the service.

Transaction Type Select the transaction type performed by the service. The available values are **Read**, **Add**, **Change**, **Update**, **Delete**, **List** and **Search**.

> **NOTE:** The difference between Change and Update is that for Change, all field values must be passed in with the request. Field values that are not passed in to the request are set to null. For Update, you need only pass the primary key field values and the values of the fields to be updated. All other fields retain their existing values.

Left Panel

The left panel of the schema editor displays a tree view of the hierarchical elements in the schema. The (+) expands a node, the (-) collapses a node.

Right Panel

The following attributes appear on the right panel of the Schema Editor. Some fields cannot be modified in the schema editor. The field description indicates when the field is protected.

Tag Name The XML element tag name. This field is protected, but you may modify this attribute to give the element a self-explanatory name by right-clicking on the element name in the left tree-view.

MetaInfo Name Maps the element to a fully qualified field name in the service, for example PER_ID. This field is protected.

Internal Type This property is populated automatically when you generate the schema from your product. The values further define elements and attributes. The values are **page**, **pageBody**, **list**, **listHeader**, **listBody**, **searchHeader**, **codeTableDesc**, **Private**. The values of **codeTableDesc** and **Private** are used to define special types of attributes.

Private attribute A field that does not exist on the server side, but one that you still want to have in the schema.

Description A description of this field.

Content The element type. This field is only available for elements. Possible values are **eltOnly**- element may contain only other elements and no text, **TextOnly**- element may only contain text.

Search Type Services, which perform a Search, may allow searching based on different criteria. The values are taken from the system meta information when the schema is generated. The possible values are **Main**, **Alternate1**, **Alternate2**, **Alternate3**, **Alternate4**, **Alternate5** and **Alternate6**.

> **NOTE:** You would not typically modify this value because it corresponds to a value in the meta information. However, the value is modifiable to accommodate the rare case in which a service may change in a new release. In this scenario, you may prefer to update the schema manually rather than regenerate a new schema for the new version.

Is Part of Primary Key Used to indicate to the XAI server whether or not this field makes up part of the primary key of the record. The values are taken from the metadata information when the schema is generated. Value may be **true** or **false**.

> **NOTE:** Typically you would not modify this value because it corresponds to a value in the meta data information. However, the value is modifiable to accommodate the rare case where a service may change in a new release. In this scenario, you may prefer to update the schema manually rather than regenerate a new schema for the new version.

Min Occurs This field is available for elements only and is used for repeating elements. It defines the minimum number of occurrences for an element. Value may be 0 or 1.

Schema Max Occurs This field is available for elements only and is used for repeating elements. It defines the maximum number of occurrences for an element. Value may be 0, 1 or *.

Limit Number of occurrences This field is available for elements only and is used for repeating elements. If the **Schema Max Occurs** field has been set to '*', define the number of max occurrences here.

XML Data Type The data type for the attribute. Possible values are **number**, **string**, **decimal**, **date**, **dateTime**, and **boolean**.

Server Data Type Indicates the data type of this attribute on the server. This field is protected.

Service Format The format expected by the service. At runtime, XAI converts the Tag format to the Service Format before executing the request. Formats are defined in *XAI Format*.

Tag Format The format used to format an element/attribute in the schemas. Formats are defined in *XAI Format*.

Min Length Use this property to define the minimum length of the attribute, if applicable.

Max Length Use this property to define the maximum length of the attribute, if applicable.

Precision This is used for decimal attributes to define the maximum number of digits.

Scale This field is used for decimal attributes to define the number of digits at the right of the decimal point.

Required A value of **Y** indicates that the element must appear in XML document. A value of **N** indicates that the element is optional.

Default value Default value to be used for Request schema, when the element is not supplied as part of the XML request document.

Fixed Value Fixed value to be used for Request schema. This value is used regardless of the value supplied in the request document.

Code Table Field This property is used for attributes that are descriptions of a code table, where the description is not automatically returned by the system service. Use this property to indicate the code whose description should be retrieved by the XAI server.

Code Table Program This property is used for attributes that are descriptions of a code table, where the description is not automatically returned by the system service. Use this field to indicate the program that XAI should call to access the description for the **Code Table Field**.

Creating a Schema

Usually you do not create schemas from scratch; rather you use Schema Creation Wizards to import existing data structure definitions from a variety of data sources:

- System services
- Comma Delimited Files (CSV)
- Database Extract
- Any XML document

Once a schema is created based on the existing data structure, it is displayed in a TreeView on the left panel. Once the imported schema has been edited, it serves as the basis for creating the request and response schemas. When imported, the schema exposes all fields defined in the service. You may want to remove some attributes/elements from the request or response schema.

> **NOTE:** Although the main purpose of the editing process in the creation of the request and response schemas is the elimination of elements, which makes the schema shorter and more understandable, it is not required for processing purposes. Therefore, if you don't mind that you have not used elements in your schemas, you could stay with one schema, which serves as both the request and response schema.

1. Save the Schema as a Request schema with an appropriate name, for example PersonInfoRequestSchema.xml

2. To create the Response schema, which is identical to the request schema, use the Save As Response menu option. This renames the top element of the schema to ServiceNameResponse, for example PersonInfoResponse and save the schema under a different name i.e. PersonInfoResponseSchema.xml. Note that if the request and response schemas are identical then one schema may be used for both and there is no need to create separate schemas.

3. Read in the Request Schema (File/Open) and modify its structure. Depending on the service type, you'll have to modify the contents of the Request Schema. This is usually required when the service is an "Info" service, which requires very few input elements. In such cases you'll delete most of the elements on the schema and only leave the necessary elements required to invoke the service. For example: in the PersonInfo request, you only need the PersonId and the Company elements in the request schema.

4. Read in the Response Schema (File/Open) and Modify its structure. Depending on the service type, you'll want to modify the contents of the Response Schema. This is usually required when the service is an "Add" or "Delete" service, which returns very few input elements. In such cases you'll delete most of the elements on the response schema and only leaves the necessary elements required by the requester of the service.

Adding an Element/Attribute

Usually, you won't have to add element or attributes to a schema. However if the schema already exists and you want to add an element/attribute, you can follow this procedure. Be aware that any element/attribute added here must also exist on the xml metainfo.

- Select the element's node on the TreeView.
- Right click on it and select the 'Add Element' or 'Add Attribute' option in the pop-up menu

- Enter the element/attribute name in the prompt dialog box and click OK.

Removing Elements/Attributes

When generating a schema using one of the wizards, the generated schema may contain information that you do not want to publish as part of the service, or is not required for a particular service. You can remove elements/attributes from the schema, and though these elements/attributes may still exist on the service they are not seen by the XAI service using this schema. To remove an element or attribute:

- Select the node to be removed.
- Right click and select "Delete" from the popup menu.

Renaming an Element

To rename an element:

- Select the element's node on the TreeView.
- Right click it and select the Rename option in the pop-up menu
- Enter the new name in the prompt dialog box and click OK.

> **NOTE:** The information in this table is cached by the XAI server and by the MPL server. Changes to values in this table do not affect the runtime copy of these servers. Refer to *XAI Command* for information about refreshing a schema.

Validating a Schema

Although a schema is validated against the metainfo XML file when it is read into the editor or before it is saved, you can perform the validation at any time while the schema is being edited. To validate a schema, click on the toolbar "Validate" button  . If the schema fails to validate the schema errors dialog is displayed.

Schema Validation Errors

When the editor fails to validate a schema against the xml metainfo file, it pops up a dialog that lists the errors found in the schema definition. These errors may be of two types:

- An element or attribute in the schema could not be found in the xml metainfo file. You can click **Remove** to remove the element from the schema.
- The data type of an attribute does not match the one defined in the metainfo file. You can click **Correct**, and the editor fixes the data type so it does match.

The **Correct All** button can be used to correct all fields that have data types that do not match the one in the XML metainfo file.

> **NOTE:** Correcting errors does not save the schema definition into a file. You have to save it manually.

If you **Exit** without correcting the errors, the schema displays with the mismatch information highlighted in red.

Registering a Service

Before a service can be used it must be defined in the *XAI Inbound Service* table in the XAI registry. A service can be registered in the XAI registry directly from the schema editor. Go to the menu item 'Register Service' in the 'Schemas' menu. The Register service UI page appears. Fill in the required fields.

> **NOTE:** The registry entry for the service can be later modified using the *XAI Inbound Service* page.

Testing a Schema

The schema editor provides a testing option.

> **NOTE: Testing in your product**. You may also test your schemas and your services using either *XAI Submission* or *XAI Dynamic Submission*.

System Wide Functions for Schema Editor

Because the schema editor is in an application outside of the standard products, this section introduces some general functions related to the application.

Application Standards

- The schema editor is a Multiple Document Interface (MDI) windows application. It may contain multiple active windows. You may jump from one window to the other using the Window menu option.
- The Editor window is always active. Closing the schema editor window quits the application.
- In the Editor window, a splitter bar is available to resize the schema tree view horizontally.
- Actions may be performed using menu items or by clicking on toolbar buttons.
- All messages are displayed on a status bar at the bottom of the screen. Some may also be displayed using message boxes.

The File Menu

Use the File menu to open existing schemas and to save a schema to a file.

Connect

Connects to the database.

Open - Loading an existing schema into the editor

You can read an existing schema into the editor.

1. Click on the Open toolbar button or select the File/Open menu option.
2. A file selection dialog is shown. Select the schema file name.
3. The editor first validates the schema against the xml metainfo file. If it fails to validate it shows the *Schema Validation Errors* dialog.

Save

Save the current schema to a file. Using the current file name.

To save a Schema, click on the Save toolbar button, or select the File/Save menu option. When you save a schema, the editor first attempts to validate the schema. If it fails to validate it against the XML Metainfo file, you are prompted to save it with inconsistency errors or to return to the editor.

Save As

Save the current schema to a file. Use a different file name.

Save As Response

Save a copy of the current schema to a file as a response schema. Use a different file name.

The View Menu

Use the view menu to perform actions on the Tree View nodes or to view errors. The following menu options are available:

Expand All

Expand all nodes in the Tree View

Collapse All

Collapse all nodes in the Tree View

Expand Branch

Expand the selected node and all the node's children

Search (`Ctrl+F`)

Find a node with a node name containing a given string

Search Again (`F3`)

Find the next node with containing the search string

View Schema Errors

Display the Schema Validation Errors dialog.

Web Browser

Display the current schema definition on a web browser page

The Schemas Menu

Use the Schemas menu to create, test, validate and register schemas.

- To create schemas from various sources use the **Create menu**.
- To *validate a schema* select the **Validate** option (`Ctrl+V`).
- To create a sample instance and test the schema, select the **Test** option (`Ctrl+T`).
- To create an entry in the service registry for a service represented by the current schema, select the **Register** Service option (`Ctrl+R`). Refer to *Registering a Service* for more information.

The Export Menu

The options on this menu are no longer supported.

The Options Menu

Always Save As W3C

Turn on this option to save schemas in W3C format by default instead of XDR format.

Always Save As DTD

Turn on this option to save schemas in DTD format by default instead of XDR format.

Preferences

The following options may be set on the preferences dialog (Select Options and then Preferences):

- The Default Date Time Format - used for schema date fields

- The default Schema Directory - used to read/save schemas
- The default Metainfo Directory - used to create/validate schemas
- The XAI Servlet URL - used when executing requests from the test schema dialog.
- The Default Account Id - used when generating sample instances of a schema
- Language - used to access language specific data

The Tools Menu

Schemas tools can be invoked from the **Schema Editor Tools** menu.

Converting Schemas to a W3C compatible schema

Schemas generated in the Microsoft BizTalk-compatible format (XDR format) may be saved in a format compatible with the *October 2000, W3C XML* schema standard. To save a schema in a W3C format:

- Select **Convert to W3C** from the **Tools** menu. The **Convert to W3C** dialog box appears.
- Select the schema(s) to be converted. Multiple schemas may be converted in a single step.
- The name of the W3C schema is the same name as the schema but with an ".xsd" file extension, and is saved to the same directory as the original schema.
- Click **Convert**.

Converting Schemas to a DTD

Schemas generated in the Microsoft BizTalk-compatible format (XDR format) may be saved as a DTD.

- Select the **Convert to DTD** option from the **Tools** menu. The **convert DTD** dialog box appears.
- Select the schema(s) to be converted. Note that multiple schemas may be converted in a single step.
- The name of the W3C schema is the same name as the schema but with a ".dtd" file extension, and is saved to the same directory as the original schema.
- Click **Convert**.

Validating multiple schemas

The schema validation tool can be used to validate the correctness of an XAI schema when compared to the metainfo xml definition used to generate the schema. For each validated schema, the validation tool scans the list of elements/attributes and compares them with those defined in the XML metainfo file. Select **Validate Schemas** in the **Tools** menu.

- The **Validate Schema** dialog box appears.
- The left list box is used to select the directory where the schemas to be validated are stored. By default this is the **Schema Directory** as defined in the preferences.
- On the right, the list of schemas is displayed on a grid.
- Multiple schemas may be validated in a single step.
- To select a schema click on the left button (the first column in the row).
- To select multiple schemas, hold the `Ctrl` key and select the required schemas.
- Click `Validate Schemas`.
- The schema grid is updated. When an attribute defined in the schema is missing from the metainfo file or when the properties of the element do not match defined in the metainfo, a button is displayed at the right of the schema name. Clicking on the **edit** button loads the schema into the editor and displays the *Schema **Validation Errors*** dialog for that schema. You can correct the schema and save it.

Setting Up Your XAI Environment

This section describes the control tables available to administer your XAI environment.

Message Class

The Message Classes are references to actual Java classes. The Message Class defines the Java class used to implement Receivers, Senders, Adapters and Executers. This information is provided with the system and does not need to be modified for a specific installation.

To view an Message Class, open **Admin** > **Message Class** .

Description of Page

The **Message Class** and **Description** are unique identifiers of the Message Class. The **Class Definition** indicates the Java class, implementing the adapter, receiver, sender or executer.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_CLASS*.

XAI Envelope Handler

To view your envelope handlers, open **Admin** > **XAI Envelope Handler** . This information is provided with the system and does not need to be modified for a specific installation.

Description of Page

Enter a unique **XAI Envelope Handler ID** and **Description**.

Indicate whether the **Envelope Type** is **Custom SOAP Envelope**, **Default** (no SOAP environment) or **SOAP Envelope**. Note that the values of **Siebel Integration Message** and **Siebel VBC** are not supported.

When the envelope type is **SOAP Envelope**, indicate the **Envelope URI**.

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

Setting Up Your Registry

The following section describes the control tables that are logically considered part of the XAI Registry.

XAI JNDI Server

To define a new JNDI Server, open **Admin** > **XAI JNDI Server** .

Description of Page

Enter a unique **XAI JNDI Server** and **Description**.

Indicate the Provider URL, which is the URL of the *JNDI server*.

Indicate the **Initial Context Factory**, which is a Java class name used by the *JNDI server* provider to create JNDI context objects.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_JNDI_SVR*.

XAI JDBC Connection

To view an XAI JDBC Connection, open **Admin** > **XAI JDBC Connection** > **Search** .

Description of Page

Enter a unique **XAI JDBC Connection** and **Description**.

Use the **Connection Type** to indicate how the JDBC connects to a database. The following connection types are valid:

- **Oracle Defined Connection** indicates the connection is to an Oracle database through a JNDI entry.
- **DB2 Defined Connection** indicates the connection is to a DB2 database through a JNDI entry.
- **JNDI Defined Connection** indicates the connection is using the MQ series classes implementing JMS.
- **Determined by parameter file** indicates that the connection information should be determined by looking at the parameters defined at *Installation*.

For connection types of **Oracle** or **DB2**, use the **JDBC URL** to indicate URL of the database connection to be initialized at XAI/MPL startup time. Indicate the **Database User** and **Database Password** required for accessing the database. The JDBC connection URL can either be a Type 2 or a Type 4. For example:

- Type 2: jdbc:oracle:oci8:@CD200ODV
- Type 4: jdbc:oracle:thin:@myhost:1521/ CD200ODV

For a connection type of **Determined by parameter file**, indicate the parameter substitutions, which should be accessed from the parameter file for the JDBC URL, database user and database password, for example, @JDBCURL@, @DBUSER@ and @DBENCPASS@.

When the connection type is **JNDI**, indicate the **XAI JNDI Server** and the **JNDI Data Source** name as defined in the JNDI.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_JDBC_CON*.

XAI JMS Connection

To define a JMS Connection, open **Admin** > **XAI JMS Connection** > **Add** .

Description of Page

Enter a unique **XAI JMS Connection** and **Description**.

Indicate the **XAI JNDI Server** to be used. Refer to *XAI JNDI Server* for more information.

Use the **JNDI Connection Factory** to indicate the lookup keyword in the JNDI server used to locate the JMS connection.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_JMS_CON*.

XAI JMS Queue

To define your JMS Queue values, open **Admin** > **XAI JMS Queue** .

Description of Page

Enter a unique **XAI JMS Queue** and **Description**.

Enter the **Queue Name** as defined in the JNDI server. This is the JNDI lookup name identifying the queue.

Use the **Target Client Flag** to indicate whether or not the target client is **JMS** or **MQ**.

Select the **XAI JNDI Server** where the queue is defined. Refer to *XAI JNDI Server* for more information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_JMS_Q*.

XAI JMS Topic

To define your JMS Topic values, open **Admin** > **XAI JMS Topic** .

Description of Page

Enter a unique **XAI JMS Topic** and **Description**.

Select the **XAI JNDI Server** where the topic is defined. Refer to *XAI JNDI Server* for more information.

Enter the **Topic Name** as defined in the JNDI server. This is the JNDI lookup name identifying the topic.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_JMS_TPC*.

XAI Format

Open **Admin** > **XAI Format** to define the various formats.

Description of Page

For each new format, specify a unique **XAI Format** name and **Description**.

Indicate whether the Format Type is a **Currency formatting string**, a **Date/Time formatting string**, a **Phone formatting string** or a **Text formatting string**.

Finally, indicate the **Format Expression**, which defines the formatting pattern to be applied.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_FORMAT*.

XAI Adapter

To define a new adapter, open **Admin** > **XAI Adapter** .

Description of Page

Indicate a unique **Adapter Name** and **Description**.

Indicate the **Message Class**, which is the name of the Java class, implementing the adapter. The class should be one that is defined for an adapter. The adapter classes provided with the product are **BASEADA**- Core Adapter, **BUSINESSADA**- Business Requests Adapter, **LDAPIMPRTADA**- LDAP Adapter, and **XAICMNDADA**- XAI Command Adapter. Note that the values **SIEBELADA** and **STGUPADA** are no longer supported.

FASTPATH: Refer to *Message Class* for more information.

The following fields are not applicable for the **BusinessAdapter** adapter.

Use the **XAI JNDI Server** to indicate the name of the WebLogic JNDI server running your product. Refer to *XAI JNDI Server* for more information.

Indicate the **Default User** to be passed to your product server when this adapter is executed.

> **NOTE:** If the XML request is sent over an HTTP connection, which has been authenticated, the authenticated User Id is passed to your product.

The **Default Date** format and the **Default DTTM** (date / time) **Format** specify date and date/time *formats* to use when a schema does not explicitly indicate formats.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_ADAPTER*.

XAI Executer

To define a new Executer, open **Admin** > **XAI Executer** .

Description of Page

Enter a unique **Executer ID** and **Description**.

Indicate the **Message Class** for this executer. The class should be one that is defined for an executer. The executer class provided with the product is **XAIURLEXEC**- XAI Executer.

Indicate the appropriate **Executer URL**.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_EXECUTER*.

XAI Group

XAI groups are used to process XML files, which contain a collection of *XML messages* to be uploaded in batch.

XAI Group - Main

To define your XAI groups, open **Admin** > **XAI Group** > **Search** .

Description of Page

Enter a unique **Group** and **Description** for the XAI Group.

Indicate the **Parser** used for this group. Possible values are **Dom Parser** and **StAX Parser**.

> **NOTE:** **Dom Parser** reads the full XML document into memory and therefore is not ideal for larger XML documents.

Indicate the **XPath** and **XPath Value**, which an XML file receiver uses to identify which group a given XML file belongs to.

- For **StAX Parsers** the XPath is limited to the root element.
- For **Dom Parsers**, the XPath supports defining elements at a lower level than the root element.

XAI Group - Attachments

Open **Admin** > **XAI Group** > **Search** and navigate to the **Attachments** tab to define attachments for your group.

Description of Page

For each entry in the attachments collection, indicate the **Sequence** and the **Root Element**. Use **Include Elements** to indicate if **Parent** elements should be included along with the current element when applying the XAI rules.

FASTPATH: Refer to *XML Message File* for more information about how this is used.

XAI Group - Rules

Open **Admin** > **XAI Group** > **Search** and navigate to the **Rules** tab to define rules for your group.

Description of Page

For each entry in the rules collection, indicate the **Sequence**, the **Priority**, the **XPath** name and **XPath Value** and the **XSL File Name**.

NOTE: **Include Parent**. If your attachment indicates that **Parent** elements should be included, be sure that the parent element is included in the XPath defined here.

FASTPATH: Refer to *XML Message File* for more information about how this is used.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_RGRP*.

XAI Receivers

XAI Receiver - Main

To define your XAI receivers, open **Admin** > **XAI Receiver** > **Search** .

Description of Page

Enter a unique **Receiver ID** and **Description** for the XAI Receiver.

Indicate the **Message Class** for this receiver. The class should be one that is defined for a receiver. The receiver classes are **DWNSTGRCVR**- Download Staging receiver, **FILESCANRCVR**- Upload Files from a directory, **JMSRCVR**- JMS Queue receiver, **OUTMSGRCVR**- Outbound Message receiver, **STGCTLRCVR**- Staging Control receiver, **STGRCVR**- Staging Upload Receiver and **TPCRCVR**- JMS Topic receiver, **XMLFILERCVR**- XML File receiver.

FASTPATH: For more information, refer to *Designing XAI Receivers* about different types of receivers.

Indicate whether or not this receiver is currently **Active**.

Identify the **Executer ID**. Select the XAILOCAL executer if the Message Class for this receiver is STGCTLRCVR. Select the BYPASSXAI executer if the Message Class for this receiver is OUTMSGRCVR. For all other receivers select the XAIURL executer. For more information, refer to *XAI Executer*.

Indicate whether the **MSG Encoding** is **ANSI message encoding** or **UTF-8 message encoding**.

The **Read Interval** indicates the number of seconds between read cycles.

Start At Time and **Duration** are not currently in use.

If the Message Class for this receiver is **FILESCANRCVR**, **STGRCVR**, **STGCTLRCVR** or **XMLFILERCVR**, indicate the **XAI JDBC Connection**.

> **FASTPATH:** Refer to *XAI JDBC Connection* for more information.

Turn on **Sequential Execution** if the received requests should be processed in *sequential order* (instead of multithreaded). If this value is turned on then XAI staging control records created by this receiver are marked for sequential execution.

JMS Information

The following information is only available if the Message Class is **JMSRCVR** or **TPCRCVR**.

Indicate the appropriate **XAI JMS Connection**

> **FASTPATH:** Refer to *XAI JMS Connection* for more information.

Indicate the appropriate **XAI JMS Queue.**

> **FASTPATH:** Refer to *XAI JMS Queue* for more information.

Indicate the appropriate and **XAI JMS Topic**.

> **FASTPATH:** Refer to *XAI JMS Topic* for more information.

File Information

The following information is only available if the Message Class is **FILESCANRCVR** or **XMLFILERCVR**.

Use the **Scan Directory** to indicate where to look for new files.

In **Scan File**, indicate the file pattern. All files with names matching the pattern are uploaded into the staging upload table. For each file found, a record in the staging control table is created.

> **CAUTION:** WARNING. MPL expects all files conforming to the Scan File pattern to be complete. If a file is in the process of being copied into the scan directory and its name conforms to the naming pattern, MPL still attempts to process it and may issue an error for the incomplete file. It is suggested that files first be copied into the scan directory with a different name that does not conform to the naming pattern, for example filename.xml.inprocess. Once the file copy/transfer is complete, rename the file to one that conforms to the naming pattern, for example, filename.xml.

The following information is only available if the Message Class is **FILESCANRCVR**.

Use the **XAI In Service Name** to indicate how the records in the file are mapped and how they are transformed to match a system service request structure.

XAI Receiver - Context

Open **Admin** > **XAI Receiver** > **Search** and navigate to the **Context** tab to define context for your receiver.

Description of Page

The Context collection enables you to define a collection of **Context Types** and **Context Values** defining. Use this collection when you need to store an attribute of a receiver that is not catered for in the current table.

> **NOTE:** The values for the Context Type field are customizable using the Lookup table. This field name is RCVR_CTXT_FLG.

XAI Receiver - Response

Open **Admin** > **XAI Receiver** > **Search** and navigate to the **Response** tab to define where to send responses to requests made by this receiver. Refer to *Designing Responses for a Receiver* for more information.

Description of Page

The response collection enables you to define the destination (**Message Sender**) where responses to a request may be sent under various circumstances (**Event**). The events currently defined with the product are **All events**, **Message executed OK**, **Application Error**, **System Error**.

NOTE: The values for this field are customizable using the Lookup table. This field name is ON_EVENT_FLG .

XAI Receiver - Groups

Open **Admin** > **XAI Receiver** > **Search** and navigate to the **Groups** tab to the valid XAI groups for an XML file receiver.

Description of Page

This collection is only available if the Message Class is **XMLFILERCVR**.

For each entry in the Group collection, indicate the **Priority** and the **Group**. Refer to *XAI Groups* for more information about defining groups.

Where Used

Receivers are used by the XAI server and by the MPL server to process messages sent to the system from various sources.

XAI Inbound Services

The XAI Inbound Services section in the registry is the main section of the registry. It is used to define the service characteristics. Basically, a service is defined by an Adapter responsible for executing the service, a pair of XML schemas and connection attributes. The Adapter defines the interface with the target application server, while the schemas define the structure of the request XML document expected by the service and the structure of the response XML document generated by the service.

XAI Inbound Service - Main

To update an inbound service, open **Admin** > **XAI Inbound Service** > **Search** .

CAUTION: Important! When adding new inbound services, carefully consider the naming convention of the XAI In Service Name. Refer to *System Data Naming Convention* for more information.

Description of Page

Define a unique **XAI In Service Name**. This information is used in the system to identify the service. The service name is also the first XML element after the <Body> element in the XML request/response document. The system generates a unique **XAI Service ID**, which serves as the primary key.

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

Indicate the **Adapter**, which defines the interface with the target application server.

FASTPATH: Refer to *XAI Adapter* for more information.

If adapter for this service should invoke a system service, then indicate the appropriate **Service Name**.

FASTPATH: Refer to *Service Program* for more information about defining services.

If adapter is the base package **Business Adapter** then **Service Name** does not appear. Instead, use **Schema Type** to indicate the type of object this service invokes and **Schema Name** to reference the object to invoke. Using this adapter, you may set up service to invoke *business objects*, *business services* and *service scripts*.

FASTPATH: Refer to *Designing XAI Adapters* for more information about the **Business Adapter**.

Use the **Description** and **Long Description** to describe the service.

Check the **Active** switch if this service is enabled and available for execution. If this switch is unchecked, then the service is defined in the registry, but not yet available for public use.

Check the **Post Error** switch to support *inbound message error handling* for messages that are not processed via the staging upload table.

Check the **Trace** switch if you would like the trace to be on for this particular service. If the general trace option is not activated, you can force a trace for a particular service.

FASTPATH: Refer to *Server Trace* for more information about trace functionality.

When the **Debug** switch is checked, debug information is generated on the XAI console when this service is executed. The debug information can be useful to resolve problems.

Schema Definitions

NOTE: Request Schema and Response Schema are not applicable to services invoking schema-based objects. They do not appear when the **Business Adapter** is used.

The next two properties define the request and response XML schemas. The schemas were created using the *Schema Editor* and are SOAP compatible. The schema XML files are expected to be stored in the Schemas Directory on the Web server running the XAI server.

The **Request Schema** is the XML schema defining the service request. The request sent to the server must adhere to the schema definition.

The **Response Schema** is the XML schema defining the service response. The response generated by the XAI server corresponds to the response schema definition.

The same service may perform several actions on a business object. Use the **Transaction Type** to define the default action performed by a service. The transaction type can be provided when invoking a service, by dynamically specifying a transaction type attribute on the Service element of the XML request. This field may take the following values: **Read**, **Add**, **Change**, **Update**, **Delete**, **List** and **Search**.

NOTE: The difference between **Change** and **Update** is that for **Change**, all field values must be passed in with the request. Field values that are not passed in to the request are set to null. For **Update**, you need only pass the primary key field values and the values of the fields to be updated. All other fields retain their existing values.

Services, which perform a Search, may allow searching based on different criteria. When the Transaction Type value is **Search**, use the **Search Type** to define the default search criteria. The possible values are **Main**, **Alternate1**, **Alternate2**, **Alternate3**, **Alternate4**, **Alternate5** and **Alternate6**.

NOTE: This is a default definition only and it may be overridden at run time when the service is invoked. To override the search type at run time, you should specify the searchType attribute on the Service element of the XML request.

XSL Transformation Definitions

Sometimes, the XML request document does not conform to the request schema, or the response document expected by the service requestor is not the one generated by the adapter. In such cases the request and/or the response documents must be transformed. The XAI server supports transformation through XSL transformation scripts. Transformation scripts may be applied to the request before it is passed to the adapter or applied to the response document before it is sent to the service requestor.

The **Request XSL** is the name of the XSL transformation to be applied to the request document before processing it. The transformation is usually required when the incoming document does not correspond to the XAI service request schema therefore it has to be transformed before it can be processed by the adapter.

The **Response XSL** is the name of the XSL transformation to be applied to the response document when the requester of the service expects the response to have a different XML document structure than the one defined by the response schema for the service.

Click the **WSDL URL** hyperlink to launch a separate window that contains the WSDL definition for the inbound service. Note that the server name and port number for the URL are built using a setting in the common properties file using the XAI HTTP Caller URL setting.

NOTE: Refer to *WSDL Catalog* for information on how to obtain the WSDL catalog for all XAI Inbound Services.

XAI Inbound Service - Staging

The staging tab is used to define parameters for services that use the Staging Upload adapter.

FASTPATH: Refer to *XAI Upload Staging* for more information.

Open **Admin** > **XAI Inbound Service** > **Search** and navigate to the **Staging** page to define attributes for your upload staging adapters.

Description of Page

Indicate the **Staging File Type** to be processed by the staging upload service. Possible values are **Comma Delimited file**, **Database Extract** and **Sequential file**.

The format of the records in the input file are not in an XML format and do not correspond to an XAI service schema. As a result, the input record must be transformed into an XML message that conforms to an XAI service request schema. Enter the **Record XSL**, which indicates the XSL transformation script used to transform the input record into the appropriate XML message.

For **sequential files** and **Comma delimited files**, indicate the **Input File Name** to be processed.

NOTE: This parameter can be overridden in the **Staging Control** table when a request to execute such a service is made.

When the service takes its input from a **Database extract**, indicate the **JDBC Connection** used to connect to the database that contains the input data.

NOTE: If this value is not populated XAI uses the default JDBC connection, which is the current product database.

FASTPATH: Refer to *XAI JDBC Connection* for more information about defining these values.

Use the **Interface Name** to provide a description of the interface being implemented through this service.

XAI Inbound Service - Parameters

This tab enables you to define parameters that are used as selection criteria by the DB Extract staging upload service.

Open **Admin** > **XAI Inbound Service** > **Search** and navigate to the **Parameters** page.

Description of Page

The **Parameters** that were defined under the Request element in the schema are displayed here. They are used to drive the extraction process. This tab only displays the list of parameters. The values for these parameters can later be entered when the control record to invoke this service is created.

> **FASTPATH:** Refer to *Staging Control Parameters* for more information.

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

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *CI_XAI_IN_SVC*.

> **NOTE:** The information in this table is cached by the XAI server and by the MPL server. Changes to values in this table do not affect the runtime copy of these servers. Refer to *XAI Command* for information about refreshing a service.

XAI Route Type

Refer to the documentation for your product to find out if XAI Route Type is supported.

Maintaining Your XAI Environment

This section describes various tools provided to enable your XAI administrators to more easily maintain your XAI environment.

XAI Submission

This page exists for testing purposes. It allows you to create an XML request document and submit it to the system, to ensure that the XML has been built correctly.

XAI Submission - Main

To submit an XML document for testing, navigate to **Admin** > **XAI Submission** and navigate to the main tab.

Description of Page

This page is used to test XML schemas, which are defined for the XAI tool. Enter an appropriate XML document in the **XML Request** field. Typically, you define the XML schema using the schema editor in the XAI application. Then you would copy and paste the document here, then modify the schema to enter actual data for testing purposes.

When you have entered the document, choose Save to submit this document to the system. Note that this request information is not saved anywhere. It simply calls the system with the appropriate service name and executes the XML request.

Navigate to the Response tab to view the response.

XAI Submission - Response

To view the response to a XML document for testing, navigate to the response tab.

Description of Page

After choosing Save on the main tab to submit a test for an XML request, the response to your request is displayed in the **XML Response** text box.

XAI Dynamic Submission

This page exists for testing purposes. It is similar to the XML Submission page, but it dynamically builds your XML document based on a selected XAI service and data that you enter.

XAI Dynamic Submission - Main

To create and submit an XML document for testing, navigate to **Admin** > **XAI Dynamic Submission** and navigate to the main tab.

Description of Page

Select the **XAI In Service Name**, which identifies the XAI Inbound Service called by the XAI tool to load/update the system data. After selecting, click **Load UI**. This button causes the system to read the XML schema associated with the service and build the screen with the associated prompts and fields, which enables a user to enter data.

> **NOTE: Testing schema.** This page tests submitting the schema referenced on an XAI inbound service. If your service references an XSL transformation script, that information is ignored. This page tests post-XSL transformation.

Select the **Transaction Type** associated with this XML request. The valid values are **Add**, **Change**, **Delete**, **List**, **Read**, **Search** and **Update**. This information is built into the XML request document.

Set the **Trace** option to **yes** to request level tracing to be executed inside the XAI tool. This results in information written to a file, which may be useful in debugging.

The bottom portion of the screen contains field prompts and input fields for the data associated with the XML request linked to this service. The system dynamically builds this portion of the page by reading the XML Request associated with the service. You can enter data in the displayed fields.

> **NOTE:** This page also generates other tabs dynamically for any collections that exist for the service being displayed. These tabs vary based on the service. The response tab is the only other tab that is always present.

Enter values in the appropriate fields. You need to have some knowledge of what information is needed based on the service and transaction type. For example, if your service is accessing the account record and you want to **read** the record, you only need to provide the account id. However, if your transaction type is **add**, you need to fill in all the fields necessary for adding an account. If you have not entered values correctly, you receive an error in the Response.

When finished entering values in the fields, click **Show XML** to see the XML request built based on the schema and the values of the fields entered.

If everything looks OK, click **Submit** to execute the XML request. Note that this request information is not saved anywhere. It simply calls the system with the appropriate service name and executes the XML request.

You are brought to the Response tab, where you may view the response.

XAI Dynamic Submission - Response

To view the response to the XML document submitted for testing, navigate to the response tab.

Description of Page

After choosing Submit on the main tab to submit a test for the XML request built dynamically, the response to your request is displayed in the **XML Response** text box.

Additional XAI Tools

This section introduces some additional tools to help you maintain your XAI environment.

XAI Service Export

The service export page allows you to export the definition of an XAI Inbound Service to a file. This function may be helpful if you need to copy the definition of this service to a separate environment. To export a service, open **Admin** > **XAI Service Export** .

Description of Page

Upon entry into this page, you are provided with the current list of **XAI In** bound **Services** and their **Description** s. Use the **XAI In Service Name** search field to find the XAI service that you would like to export.

Use the **Export?** column to indicate which XAI service(s) you would like to export. Once you have selected your services, choose **Save**.

> **NOTE:** If multiple services are selected, they are exported together into the same output file.

You are presented with the standard File Download dialogue where you can open or save the file.

XAI Service Import

The service import page allows you to import the definition of an XAI Inbound Service from a file into the XAI service table. This function may be helpful if you need to copy in the definition of this service from a separate environment. To import a service, open **Admin** > **XAI Service Import** .

Description of Page

Upon initial entry into this page, you are provided with an input field, where you can enter the file name to import. Click **Browse** to search for the desired file in a directory.

Once the file is identified, click **Read File**, to read in the contents of the file.

> **NOTE:** The format of the file must include tags indicating the column names for XAI Inbound Service table along with the values of the columns. For an example of how the format should be, simply go to the *XAI Service Export* page, export a Service and view the format of the resulting file.

Once the file has been read in, the list of XAI services found defined within the file is displayed in the Import grid, identified by their **XAI In Service Name** and **Description**. In the **Import?** column, indicate which services to import.

If a service with this service name already exists in the table, you must check the **Overwrite Existing** switch in order to indicate that the imported file information should replace the current service. An *XAI Inbound Service* that is provided as part of the system (i.e., with an owner of **Base Product**) may not be overwritten.

Click Save to proceed with the import. If any problems are found, information is displayed in the **Message Text** column.

XAI Command

Use the XAI Command page to send commands to the XAI and MPL server. To execute a command, open **Admin** > **XAI Command** .

Description of Page

The following operator commands may be sent to the XAI server. For each of these commands, you may check **Also Sent to MPL URL**, in which case, the command is also sent to the MPL server. You need to indicate the URL of the MPL server.

Display Registry Use this command to display the current registry information that the XAI instance is running with.

Refresh Code and Description This is related to an attribute in the schema editor where you may indicate that the description of a control table code should be returned along with the code itself. This information is kept in cache memory in the server. As a result, changes made to descriptions have no effect on the runtime server. This command clears the cache of control table codes and descriptions accessed by the server.

Refresh Registry The registry contents are kept in cache memory in the server. As a result, making changes to the registry control tables has no effect on the runtime server. Use this command to refresh the contents of the registry cache with the new values in the registry control tables. The command reloads all registry control table data into the server.

Refresh Schema Schema definitions are stored in cache memory on the XAI server. As a result, modifying a schema definition has no effect on the runtime server. To refresh schema definitions, use the Refresh Schemas command.

Refresh Service Service definitions are stored in cache memory on the XAI server. As a result, modifying an XAI inbound service definition has no effect on the runtime server. To refresh service definitions, use the Refresh Service command. You are prompted to indicate which service to refresh.

Refresh XSL XSL Transformation script definitions are stored in cache memory on the XAI server. As a result, modifying an XSL transformation definition has no effect on the runtime server. To refresh XSL transformation definitions, use the Refresh XSL command.

Trace On Use this command to start the XAI server trace.

Trace Off Use this command to stop the XAI server trace.

XAI Trace Clear Use this command to clear the contents of the trace file.

XAI Trace Swap Use this command to rename the current trace file by appending the date and time to the end. A new trace file is then created with the name as defined in the *Message option* page.

The following operator commands can be sent to the MPL server. You must set the URL of the MPL server first.

MPL Refresh Executer Executer definitions are stored in cache memory. As a result, adding or modifying executer definitions has no effect on the runtime server. Use this command to refresh executer definitions. You are prompted to indicate the executer to refresh.

MPL Refresh Receiver Receiver definitions are stored in cache memory. As a result, adding or modifying receiver definitions has no effect on the runtime server. Use this command to refresh receiver definitions. You are prompted to indicate the receiver to refresh.

MPL Refresh Sender Sender definitions are stored in cache memory. As a result, adding or modifying sender definitions has no effect on the runtime server. Use this command to refresh sender definitions. You are prompted to indicate the sender to refresh.

MPL Start Receiver Use this command to start a particular receiver. You are prompted to indicate the receiver to start.

MPL Stop Use this command to stop all MPL activity. It stops all receivers and waits for all executers and senders to complete.

MPL Stop Receiver Use this command to stop a particular receiver. You are prompted to indicate the receiver to stop.

MPL Trace On Use this command to start the MPL server trace.

MPL Trace Off Use this command to stop the MPL server trace.

When you have chosen the appropriate command and indicated any extra information, click **Send Command** to send the command to the server(s).

If you have sent a command to the XAI Server, then the bottom portion of the screen displays the response in the **XAI Response**. If you have sent a command to the MPL Server, then the bottom portion of the screen displays the response in the **MPL Response**. If you have sent a command to both servers, the bottom portion of the screen displays both responses.

MPL Exception

The MPL Exception table is used by the MPL to keep information about requests that resulted in a system error. These are errors that occurred inside the MPL. For example, if the MPL fails to send a request to XAI (maybe WebLogic is down), this is a system error, which would be logged in the MPL exception table.

There are errors that are defined recoverable. This means that the MPL will retry the action that failed, according to the parameters it received.

Server Trace

The XAI server traces every request and response. The requests/responses are written to a trace file on the server. The trace file may be viewed using the *Trace Viewer*.

Starting the Trace

The log starts automatically based on definitions in the *Message Options* in the traceType and traceFile options. To manually start the trace:

- Navigate to **Admin** > **XAI Command** .
- Select the **Start Trace** command from the command dropdown
- Click **Send Command**

Stopping the Trace

- Navigate to **Admin** > **XAI Command** .
- Select the **Stop Trace** command from the command dropdown
- Click **Send Command**

Trace Viewer

Use the Trace Viewer utility to view the log file. The Trace Viewer is installed when you install the XAI client tools. It can be found in the XAI program group under Start/Programs.

Main Page

When the Trace Viewer starts, select a trace file to view. A trace file may be opened in one of two ways:

- To open a trace file directly from its location on the web application server, use the **File, Open HTTP** menu item and provide the appropriate URL.

- To open a trace file on the local/network file system use the **File, Open** menu item

Description of Page

Once a trace file is opened, it displays a list of all the requests on the left side including the **Service Name**, the **Start Time** and the **End Time**.

To display the XML contained in the request and response entries for a displayed request, select a request entry.

Filtering Options

Since the trace file may contain a very large number of messages, the trace viewer limits the number of messages that can be displayed. It does that by displaying messages traced within the last x number of **Minutes**, **Hours** or **Days**.

Use the **Max Messages** to limit the number of messages displayed.

> **NOTE: Default Note**. By default, the Trace viewer displays the first **200** messages in the trace file.

To view only errors in the trace, check the **Show only Errors** option.

> **NOTE: Refresh Display**. After changing any of the above filtering options, click the refresh button in order to redisplay the request entries based on the new options.

The **First Message Found** field indicates the date and time of the earliest entry in the trace file.

Viewing as Text

To view the trace file as text rather than viewing each entry in its XML format, use the **View, As Text** menu option. The contents of the trace file are displayed in text format in a separate window.

Statistics Page

Use the **View, Statistics** menu item to view the statistic page, which displays performance statistics about the XAI services that were executed in the XAI trace file.

For each type of XAI Service and transaction type, it displays the following information based on the requests traced in the XAI trace file:

- The **Service Name** with the transaction type in parentheses
- The **Number of calls** for this service in the listed trace records
- The **Average** duration **Time** (in seconds)
- The **Max** imum duration **Time** (in seconds)
- The **Min** imum duration **Time** (in seconds)

> **NOTE: Requests Included in Statistics**. Only requests falling in the time selection criteria and listed on the main log viewer are processed for calculating the statistics.

To display a Duration Chart for a particular service, check the Service. A chart such as the one below is displayed.

Chapter 16

Importing Users and Groups

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

NOTE: The topics in this section are related to the implementation of the import using *XML Application Integration* (XAI). This is not recommended for implementations going forward. The documentation remains here for upgrade purposes. The recommendation is to use the **F1–LDAP** batch job. Please refer to the LDAP Integration white paper for more information.

NOTE: **Import only**. The system currently supports importing LDAP information. Exporting users and groups from the system to LDAP is not provided.

This section describes how to set up your system to import users and groups from an LDAP store as well as how to do the import.

How Does LDAP Import Work?

The LDAP import process uses a special XAI service (LDAP Import) that reads the information from the LDAP store and creates the appropriate security entries by calling standard *XAI services* to maintain users and groups. The entire import process may be more appropriately called synchronize because groups, users, and the connections between them are synchronized between the LDAP store and your product.

Invoking The Import Process

The *staging control receiver* invokes the LDAP Import service when it encounters a **pending** record on the *XAI staging control* table with a service ID of **LDAPImport**. To create a pending LDAP import staging control record, use the *LDAP Import* page to select the users or groups to be imported and click `Synchronize`. The LDAP Import page populates all necessary request parameters when creating the staging control record.

Processing LDAP Import Requests

The LDAP import service calls the *LDAP Import Adapter*, which performs the following actions:

- It reads the LDAP configuration information provided as XAI parameters to the request. Parameters include the Java Name Directory Interface (JNDI) server, user and password for the LDAP server, and the transaction type (e.g., *import*).
- It connects to the LDAP store using a *JNDI specification*.
- For each element (user or group) in the request, the LDAP is searched by applying the search filter (specified for the element) and the *searchParm* (specified in the request).
- The adapter goes through each entry found in the search and verifies whether or not there is already an entry in the system and whether a user belongs to a user group. From this information, it automatically determines the action to be taken:
  - Add
  - Update
  - Link user to group
  - Unlink user from group (by setting the expiration date)
- If the entry is a group, the adapter also imports all the users in LDAP that are linked to the group. If the entry is a user, the adapter imports the groups to which the user belongs in LDAP.
- For each imported entity, the adapter creates an appropriate XML request and adds it to the *XAI upload staging table*. If, for example, the action is to add a user, it creates an XML request corresponding to the *CDxXAIUserMaintenance* service; and if the action is to add a group, it creates an XML request corresponding to the *CDxXAIUserGroupMaintenance* service.

The *XML upload staging receiver* processes the upload records in sequential order (based on the upload staging ID).

NOTE: No Second Order Import. If a user is imported because the name belongs to an imported group, the adapter *does not* import all the other groups to which the user belongs. If a group is imported because the imported user belongs to it, the adapter *does not* import all the other users that belong to the group.

NOTE: Long User and Group Names. Users and groups whose names exceed the length limit in the system are not synchronized.

Setting Up Your System For LDAP Import

In order to set up your system for LDAP import, you must:

- Define a JNDI server that points to your LDAP server.
- Create an XML file that maps LDAP objects to Framework objects.
- Reference the LDAP mapping file in your XAI Parameter Information and MPL Parameter Information files.

The system is pre-configured for all other XAI components that are necessary for running LDAP Import, including:

- LDAP Import Adapter *XAI class*
- LDAP Import *XAI adapter*
- LDAP Import *XAI service*
- The necessary *XML request* and *XML response* schemas (LDAPImportRequest.xsd and LDAPImportResponse.xsd)

Defining a JNDI Server That Points to the LDAP Server

XAI uses Java Name Directory Interface (JNDI) servers to locate resources that are needed by XAI. To use LDAP Import, you must define a *JNDI server* that points to the LDAP server where the users and groups you want to import are located.

NOTE: JNDI Server Initial Context Factory. The JNDI server should have its Initial Context Factory field set to `com.sun.jndi.ldap.LdapCtxFactory`. This uses Sun's LDAP JNDI provider, which is provided with the application to access the LDAP store. If desired, you can use another JNDI LDAP provider by setting a different initial context factory and adding the classes of that provider to the classpath.

FASTPATH: Refer to *Designing XAI JNDI Servers* for more information.

Mapping Between LDAP Objects And Base Security Objects

An LDAP store 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 store 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 XAI service. As part of setting up your system for LDAP import, you need to define this mapping. The base package comes with a sample mapping file that can be used when your LDAP store is a Microsoft Active Directory Server (ADS). You can use this file as the basis for creating your own mapping file if you are using a different LDAP store (e.g., Novell Directory Server).

Attribute mappings are defined in the XML parameter information file under the LDAPImportAdapter section. Note that the mapping itself is in an external file that is included in the XML parameter information file.

The XML structure:

- Maps LDAP Entries to system objects
- Maps attributes in the LDAP entry to attributes in the system object
- Describes objects linked to the LDAP entry

Mapping an LDAP Entry to a Base Object

Each LDAP entry is mapped to a base product object (User or Group) using an <LDAPEntry> element that has the following attributes:

Attribute	Description
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: - User - UserGroup

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 search on the LDAP Import page.
Scope	Sets the scope of the search. Valid values are: - onelevel (the value normally used) - subtree

Mapping LDAP Entry Attributes to Base Object Attributes

Each attribute in the LDAP entry is mapped to attributes in the base object using an <LDAPCDXAttrMappings> section under the LDAP entry. Each <LDAPCDXAttrMapping> element has the following attributes:

Attribute	Description
ldapAttr	The name of the LDAP attribute to be mapped.
cdxName	The name of the base product attribute to be mapped. Note this is the name of the attribute in the XAI schema for the User or Group maintenance services.
javaClass	The name of a Java class to be called. To provide more flexibility in attribute mappings, especially when there is not a simple one to one attribute mapping, you can derive the value of a base product attribute by calling a method in a Java class. The Java class gets the LDAP entry as input and implement its own logic for computing the value of the attribute. The class should implement the ICDXValueObtainer interface whose source can be found in the cm_templates directory. The class should be available to both the defaultWebApplication and the XAIApp.
idParm	When the javaClass attribute is specified, the IdParm attribute contains a parameter value that is passed to the Java class method.
format	When the javaClass attribute is specified, the format attribute contains a parameter value that is passed to the Java class method.
default	The default value that will be assigned to the CDx 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. - The Java class method returns a null or empty value. Default values are applied only when creating a new entity and are not applied to updated entities.

Describing Linked Objects

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 link information is required when displaying the list of objects that is affected by the import.

Linked objects are described using the <LDAPEntryLinks> section under the LDAP entry. LDAP provides two methods to retrieve the linked objects:

- The linked objects are stored as multiple-value attributes on the entity (e.g., users contained in a group are stored as a multiple-value attribute on the Group entity).
- The linked objects are retrieved by applying a search filter on an LDAP entity (e.g., the groups to which a user belongs might be retrieved by applying a search filter to the Group entity).

Each <LDAPEntryLink> element has the following attributes:

Attribute	Description

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

Example XML Mapping

The following XML excerpt describes a mapping to the Microsoft ADS. 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 **member** attribute.
- The groups to which a user belongs are retrieved by applying a search filter.

<LDAPEntries>

<LDAPEntry name=" **User**" baseDN=" **CN=Users,DC=splwg,DC=com**" cdxEntity=" **user**" searchFilter=" **(& (objectClass=user)(name=%searchParm%))**">

<LDAPCDXAttrMappings>

<LDAPCDXAttrMapping ldapAttr=" **name**" cdxName=" **User**" />

<LDAPCDXAttrMapping cdxName=" **LanguageCode**" default=" **ENG**" />

<LDAPCDXAttrMapping cdxName=" **FirstName**" default=" **fn1**" />

<LDAPCDXAttrMapping cdxName=" **LastName**" default=" **fn2**" />

<LDAPCDXAttrMapping cdxName=" **DisplayProfileCode**" default=" **NORTHAM**" />

<LDAPCDXAttrMapping cdxName=" **ToDoEntries**" default=" **1**" />

<LDAPCDXAttrMapping cdxName=" **TD_ENTRY_AGE_DAYS2**" default=" **12**" />

</LDAPCDXAttrMappings>

<LDAPEntryLinks>

<LDAPEntryLink linkedToLDAPEntity=" **Group**" linkingLDAPAttr=" **memberOf**" />

</LDAPEntryLinks>

</LDAPEntry>

<LDAPEntry name=" **Group**" baseDN=" **CN=Users,DC=splwg,DC=com**" cdxEntity=" **userGroup**" searchFilter=" **(&(objectClass=group)(name=%searchParm%))**">

<LDAPCDXAttrMappings>

<LDAPCDXAttrMapping ldapAttr=" **name**" cdxName=" **UserGroup**" />

<LDAPCDXAttrMapping ldapAttr=" **description**" cdxName=" **Description**" default=" **Unknown**" />

</LDAPCDXAttrMappings>

<LDAPEntryLinks>

<LDAPEntryLink linkedToLDAPEntity=" **User**" linkingSearchFilter=" **(&(objectClass=user)(memberOf=%distinguishedName%))**" linkingSearchScope=" **onelevel**" />

</LDAPEntryLinks>

</LDAPEntry>

</LDAPEntries>

Including Your LDAP Import Mapping in the Parameter Information Files

After you have defined your *LDAP mapping* in an XML file, you need to include it in the <AdHocParameters> sections of the XAI Parameter Information file (XAIParmameterInfo.xml) and the MPL Parameter Information (MPLParameterInfo.xml) file. The definitions are made to the LDAPImport adapter under an <Adapters> section that you add.

NOTE: Update Both Parameter Information Files. Make sure that you include the same update in both parameter information files (XAIParameterInfo.xml and MPLParameterInfo.xml).

The following example shows an excerpt from the parameter information files:

<AdHocParameters>

<Adapters>

<Adapter name=" **LDAPImport**" maxReturnedResults=" **300**">

<?XAI includeURL= **file:///c:\Cdx\Product\Servers\MAIN_PROD_ORA\cisdomain\xai\ldapdef.xml**?>

</Adapter>

</Adapters>

</AdHocParameters>

In the example above, the mapping file is called ldapdef.xml.

The maxReturnedResults attribute limits the number of LDAP entries the adapter returns for the search request. This limit is only applicable to the *LDAP Import* user interface. The import process itself is not affected by this parameter.

FASTPATH: For more information about the XML Parameter Information file, refer to *XAI Installation*.

LDAP Import

Open **Admin** > **LDAP Import** to import users and groups into the system.

NOTE: LDAP Import Setup. Your system must be appropriately configured for this page to operate correctly. Refer to *Setting Up Your System For LDAP Import* for more information.

Description of Page

Enter the **LDAP JNDI** server that should be used to connect to the LDAP server from which you want to import users or groups. Refer to *Defining a JNDI Server* for more information.

If the LDAP server from which you are importing users or groups requires authentication, specify an **LDAP User** and **LDAP Password** for the server.

NOTE: Username and password. Contact your LDAP system administrator for the format of the username and password to use.

From the **LDAP Entity** drop-down list, select either **User** or **User Group** depending on whether you want to import users or groups from the LDAP server.

Select the name of the **User** or **Group** that you want to import. (The label of this field changes depending on the **LDAP Entity** you selected.) You can use the search button to search for a list of users or groups that exist in the LDAP store.

After all fields have been specified, click the **Find** icon to return a list of all the objects contained in the selected user or group. If a user is specified, all the groups to which the user belongs are displayed in the grid. If a group is specified, all the users that belong to the group are displayed in the grid.

Click the **Synchronize** button to import the specified user or group and its linked objects (shown in the grid).

The LDAP Import service creates the configuration parameters on the XML staging control table and the *XAI Staging Control* page opens to let you view the status of your request.

FASTPATH: For more information, refer to *How Does LDAP Import Work?*

Chapter 17

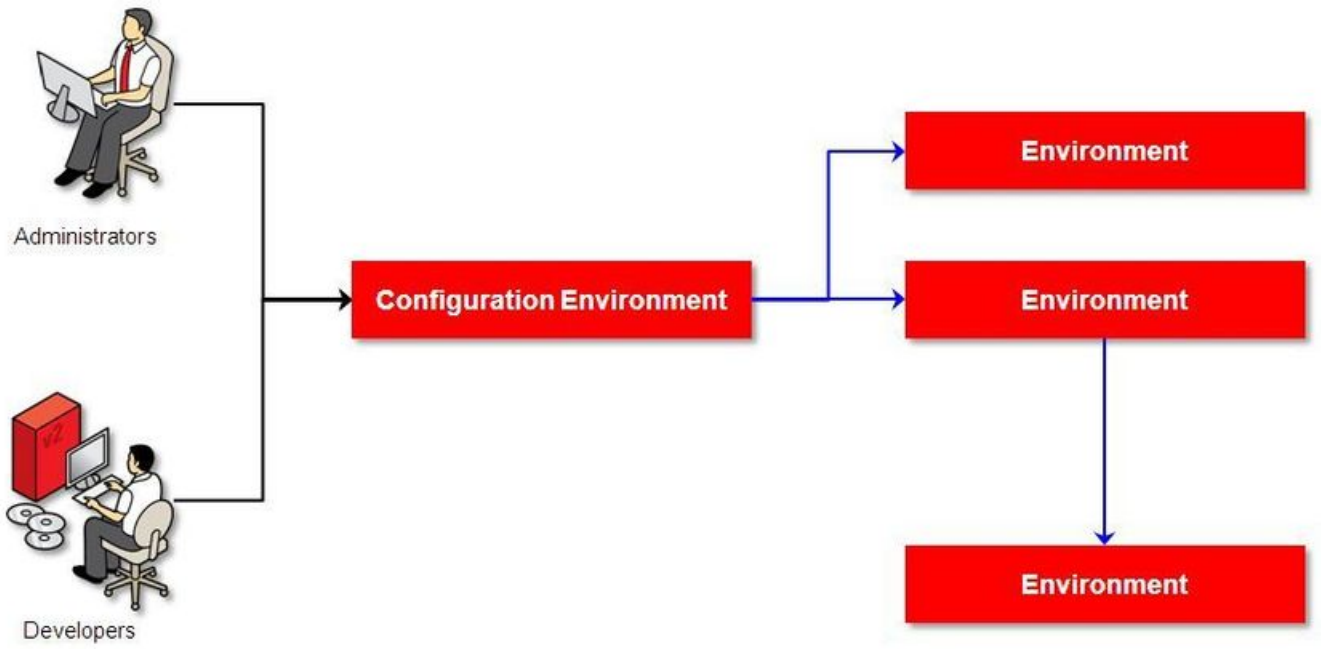
Configuration Migration Assistant (CMA)

This chapter describes the Configuration Migration Assistant (CMA), a facility to enable the export of customer-owned configuration data from one environment to another.

CAUTION: This chapter is intended for users responsible for testing configuration data and performing "what if" analysis in non-production databases.

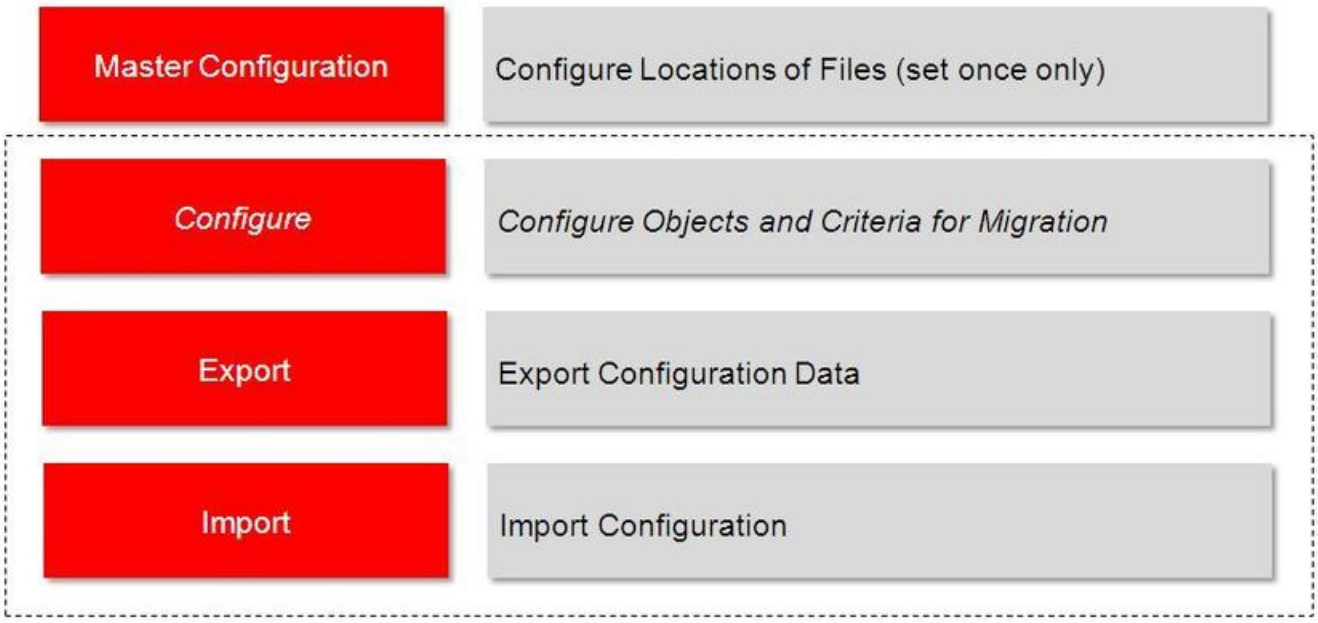
Understanding CMA

The Configuration Migration Assistant (CMA) provides implementers with a flexible, extensible facility for migrating configuration data from one environment to another (e.g., from a development environment to a production environment). Data is exported from the source system to a file. The file can then be checked in to a version control system for reuse, or can be immediately imported into the target system and applied.



NOTE: As used in this chapter, *source* systems are those on which export-related activities are conducted and *target* systems are those on which migration updates are to occur. The two system preparation tasks described in *Migration Assistant Configuration* must be performed on both source and target systems.

The CMA Process Flow



The high-level CMA process comprises the following tasks and flows. Each is described in more detail later in this section.

- **One-time tasks**, which must be applied on both source and target systems:

- Specify the source and target paths and a file extension for import and export data files in the **Migration Assistant Configuration** master configuration record.
- Create a thread pool that turns off the L2 caching. This is required to support the import step where cached data is being added or updated. It's recommended for all steps (including export and compare) to ensure that the algorithms are working with the data in the database rather than the cache. Refer to *Caching Considerations* for more information.

- **Configuration** steps are used to define the data to migrate. This task is performed on the source system and may be defined at any time. Note that the products provide base delivered configuration that may be used as is or used as a template for building more specific configuration for a given implementation.
  - Each type of record that may be copied requires a **Migration Plan**. The migration plan is used to identify the maintenance object (MO) for the record (using a business object) and allows for instructions to specify related records that may be included in the migration. Multiple migration plans may exist for a given maintenance object if there are different requirements for migrating records in that MO under different circumstances.
    - The primary instruction in the migration plan could define the physical BO of the record. This signals to CMA that all records for the MO are eligible to be migrated.
    - The primary instruction may define a specific BO, in which case only records that reference that BO in its parental hierarchy are considered.

      NOTE: Refer to *Understanding the BO Filtering Process* for more information.

    - Subsequent instructions may include references to related records. There may be some circumstances where a migration should include subsequent records and some circumstances where they shouldn't based on requirements.
  - A **Migration Request** is used to define a set of migration plans to be included together in a single migration attempt. In addition, the migration request may define selection criteria for each migration plan to limit the migration to a subset of data for each MO. Selection may be defined using SQL, an algorithm or explicit primary key values.

    FASTPATH: For more information, refer to *CMA Configuration*.

- The **Export** process includes all the steps needed to select records to be exported from the source environment and create the export file.
  - A **Migration Data Set Export** object is created for a specific migration request.
  - The lifecycle of the Migration Data Set Export business object includes algorithms that select the appropriate records according to the migration request, determine dependencies between records to build groupings of related objects and create the export file.
  - Because there may be a large volume of data in the export, many of the steps in the lifecycle of the migration data set export are executed via the **Migration Data Set Monitor**.

    FASTPATH: For more information, refer to *Exporting a Migration*.

- The **Import** processes include all the steps needed to read an imported file, compare the data in the file to the data in the target, review the proposed changes and apply the updates. The following is a high level overview of the process.
  - A **Migration Data Set Import** object is created for a given file to import.
  - The next step reads the import file and creates **Migration Transactions** and **Migration Objects** based on the information in the import file. A migration object is created for every maintenance object record to potentially be created or updated. The migration transaction is a grouping record that groups together related migration objects.
  - The next step is for the objects to **Compare** the data being imported against the data for that record in the target environment. If it is found that the migration object is new or represents a change to the existing record in the target

environment, appropriate SQLs are generated. If no changes are found, the object is marked “unchanged” and doesn’t progress.

- Once all the objects are compared, the user may review the objects for acceptance or rejection.
- When the migration objects are all accepted or rejected, the next step is to apply the objects and update the target environment.

FASTPATH: For more information, refer to *Importing and Applying a Migration*.

Migration Assumptions, Restrictions, and Recommendations

The following sections describe miscellaneous topics that are important to learn with respect to CMA.

Type of Data Migrated

CMA is designed to migrate configuration data only. The tool cannot be used to migrate master or transactional data that contains system-generated primary keys.

The comparison step of the import process will generate appropriate insert or update SQL statements for the following data found in the export:

- Configuration data in a maintenance object with no owner flag. This is purely implementation data.
- Configuration data in a maintenance object with owner flag, where the owner is **Customer Modification**. For example, implementer-specific business objects.
- Configuration data in a maintenance object with owner flag, where the main record is owned by the product but where a child record exists with an owner of **Customer Modification**. For example, implementer-specific algorithms added to a base owned business object.
- Customizable fields in a record that is owned by the product. For example, the priority of a based owned To Do type.

No Record Deletion

The CMA process allows users to define records to copy from a source environment to a target environment. In that way, the import process for the migrated records is able to identify objects to add and objects to change. There is no mechanism for indicating that records in the target environment should be deleted. The absence of those records in the import is not enough because the migration may be only importing a subset to add or update. If data on the target system must be deleted, users must delete the records in the target accordingly.

Note that CMA does support the deletion of child rows of an object as a result of a comparison. This is only applicable to child records that are owned by the implementation.

CMA Configuration

The following sections describe tasks required for CMA configuration.

Master Configuration - Migration Assistant

Before proceeding with your first migration, system wide configuration must be defined. This is captured in the **Migration Assistant Configuration** master configuration record. This must be defined in both the source environment and the target environment.

Navigate using **Admin** > **Master Configuration.** In the **Master Configuration** list, scroll to **Migration Assistant Configuration** and click the add icon if there is no record yet or the edit icon to modify an existing record.

For more information about specific fields in the master configuration, refer to the in-line help.

NOTE: This record can be updated at any time to change details. The new configuration takes effect on all subsequent exports and imports.

Migration Plans

A migration plan defines one or more types of objects that are eligible for migration. It is essentially a set of instructions describing how the data to be exported is structured, allowing objects to be migrated together as a logical unit to ensure consistency and completeness.

The following topics describe defining a migration plan as well as other topics for a migration plan.

Defining a Migration Plan

To view or define a migration plan, navigate using **Admin** > **Migration 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.

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 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 reference. 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**. When referring to a field on the parent instruction's object, use the syntax `#PARENT.TABLE_NAME.FIELD_NAME`. When referring to a filed 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":`#THIS.F1_MAP.MAP_CD = #PARENT.F1_BUS_OBJ_OPT.BUS_OBJ_OPT_VAL AND #PARENT.F1_BUS_OBJ_OPT.BUS_OBJ_OPT_FLG = 'F1DU'`.
- The **XPath** option lets you apply syntax in an XPath expression referencing elements in the instructions' referenced business objects. This is entered in the **XPath Traversal Criteria**. For example, the display map collection statement in the SQL example noted above would be written as follows in XPath: `#this/mapCd = #parent/businessObjectOption/businessObjectOptionValue AND #parent/businessObjectOption/businessObjectOptionType = 'F1DU'`. This technique allows foreign key references that are mapped inside a CLOB to be referenced.

  NOTE: The #parent expressions may access elements that are stored in the CLOB and described using mapXML and mdField. However, the #this expressions must refer to fields available in the business object using the mapField reference.

Defining **Next Migration Plan** provides the ability to indicate that in addition to copying the object defined in the instruction, any additional instructions included in that referenced migration plan will also be included in an export.

The **Algorithms** grid contains algorithms associated with each instruction. You must define the following for each algorithm:

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

System Event	Optional / Required	Description
Import	Optional	Algorithms of this type may be used to adjust the data after it is moved to the target system. Refer to *Adjusting Imported Data* for more information. Click *here* to see the algorithm types available for this system event.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *F1_MIGR_PLAN*.

Understanding the BO Filtering Process

Migration plan instructions require the definition of a business object to provide CMA with information about the record related to the instruction. If the business object is the physical business object for the maintenance object, then CMA assumes that the instruction applies to all records that satisfy the traversal criteria. CMA recognizes the physical BO by comparing the BO to the value defined in the maintenance object option. If the business object defined is not the physical BO, then CMA will limit the records in the instruction to those that explicitly reference this BO or reference a child of this BO as its "determine BO" value. (In other words, this BO must be in the parentage hierarchy of the records to be included in the instruction.)

For example, if you define a migration plan for Master Configuration and use the physical business object for the instruction (F1-MstCfgPhysicalBO) then all master configuration records are considered for the instruction. If instead the business object you define is Migration Assistant Configuration (F1-MigrationAssistantConfig) then only the record related to this business object is included in the instruction.

Migration Plans for Objects with CLOB-Embedded Links

When migrating objects where foreign key references are captured in the object's CLOB, subordinate instructions are needed to define the foreign key references in order for CMA to understand the relationships. This is in contrast to direct foreign keys where CMA can determine the relationships using constraints. The instructions provide two purposes. For wholesale migrations, where all data (or a large amount of data) is being migrated, the instructions allow CMA to group related objects into transactions. This helps in the apply process at import time to ensure that related objects are grouped together. However, the apply process includes iterative steps to try to overcome dependencies like this so defining the instructions is not critical for this type of migration. For piecemeal migrations, defining instructions ensures that the related objects are included in the migration, if appropriate.

The following are options for creating migration plans with CLOB-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 CLOB 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 CLOB. Rather than duplicating the base migration plan and adding additional instructions for the additional foreign keys, the implementation can create a second migration plan for the MO with the additional foreign keys defined. A migration request should be defined to include both migration plans. In this case if the implementation has only one custom BO, they can choose to use the custom BO as the primary instruction as described above in the first option.

Defining a Migration Request

Migration Requests are unordered lists of Migration Plans that are to be migrated together. Algorithms, SQL statements, or specific keys are used to specify the set of objects you want to export. When complete, the request describes the complete data set that is extracted to the migration export file when the request is executed.

To view or define a migration request, navigate using **Admin** > **Migration Request** .

Use the **Migration Request Query** portal to search for an existing migration request. Once a migration plan is selected, you are brought to the maintenance portal to view and maintain the selected record.

Define one or more **Migration Plans** for the migration request. Note that the order doesn't matter. During the export process, CMA looks at all objects that qualify for the export and will group them together based on their relationships.

For each option, define an appropriate **Selection Type**, which is used to filter which records in the migration plan's object should be selected. The valid values are **Algorithm**, **SQL Statement**, and **Specific Key**.

- For the option **Algorithm**, specify an **Algorithm** for the Key Selection. The algorithm is responsible for returning a list of key values to include in the migration.

  NOTE: The algorithms that are eligible to be plugged in here are those valid for the Migration Request — Select system event. Click *here* to see the algorithm types available for this system event.

- For the option **SQL Statement**, specify an **SQL Statement** for the key selection. may be used to limit the keys to include. To migrate all records in the table, the SQL statement "1=1" may be used. Refer to the inline help for other examples.
- For the option **Specific Key**, enter one or more primary keys to individually to define the records to include. Click the "+" to enter each additional key. Each row represents one key and supports a multi-part primary key.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *F1_MIGR_REQ*.

Wholesale and Piecemeal Migrations

The Configuration Migration Assistant is used for two general types of migrations: wholesale and piecemeal. The following sections provide some additional information about these concepts.

Wholesale Migrations

Wholesale migrations are used when migrating all the configuration and/or administrative data from one environment to another. For example, a wholesale migration might be used when migrating administrative data from a development or test environment to a production environment.

A wholesale migration may be comprised of one or more migration requests that in total include all the administrative data to move. Whether one migration request is used or multiple are used depends on the following considerations:

- For each migration request used, a separate migration data set export record (and therefore separate migration data set import record) is needed. The multiple records may cause more user interaction with the data to progress all the records to their final state.
- On the other hand, splitting data into multiple migration requests, grouping information logically together may allow for more reuse.
- In addition, you should consider that the framework product provides base migration requests and your specific edge product may provide base migration requests as well that may or may not include framework migration plans. Using the product provided migration requests is beneficial with respect to maintenance. As features are added to the product (including new administrative maintenance objects), any impact on CMA should be included in the product owned migration requests. If your implementation introduces new custom administrative maintenance objects that should be included in CMA, then custom migration plans and a custom migration request should be added. Your implementation may choose to include only migration plans for your custom MOs (and simply migrate the objects in this table as a separate step in the process) or include other migration plans from the product in your custom migration plan to have a consolidated plan.

Migration plans used in wholesale migrations may be designed to omit subordinate instructions related to explicit foreign keys that are identified through constraints as they are not needed, assuming that the data they are referring to will also be included in the migration.

NOTE: Refer to *Framework Provided Migration Objects* for information about migration requests provided in the framework product. Refer to your specific product's documentation for information about addition base provided migration requests.

Piecemeal Migrations

A piecemeal migration refers to migrating a targeted subset of data from one environment to another. Examples of piecemeal migrations include:

- Migration of a new Business objects with its options and algorithms.
- Migration of a new maintenance portal, its zones, and its application service.
- Migration of all outbound message types.

Administrative users can define a migration request with the specific types of objects that should be copied and use selection criteria to limit the records to copy as desired.

The migration plans included in a 'piecemeal' migration request should be reviewed carefully to verify what subordinate instructions are included. When copying a specific type of record, there may be related records that should be copied as well and other related records that should not be copied. For example, when copying a custom To Do Type, perhaps any algorithms it refers to should be copied (along with their algorithm types) and its message should be copied, but not its navigation option or its drill keys, which would probably refer to base values.

Characteristic Type Configuration

The CMA import process will attempt to create a log record for any administrative object that includes a log table. If your implementation has introduced any custom administrative tables that you plan to include in a migration request and it includes a log table, you must, to ensure that the log creation is successful, add your log table as a valid characteristic entity to the characteristic type **F1-MGO** (Migration Object).

Navigate using **Admin** > **Characteristic Type** > **Search** and select the characteristic type **F1-MGO**. Navigate to the **Characteristic Entity** tab and add a row to include the characteristic entity for your custom maintenance object's log table.

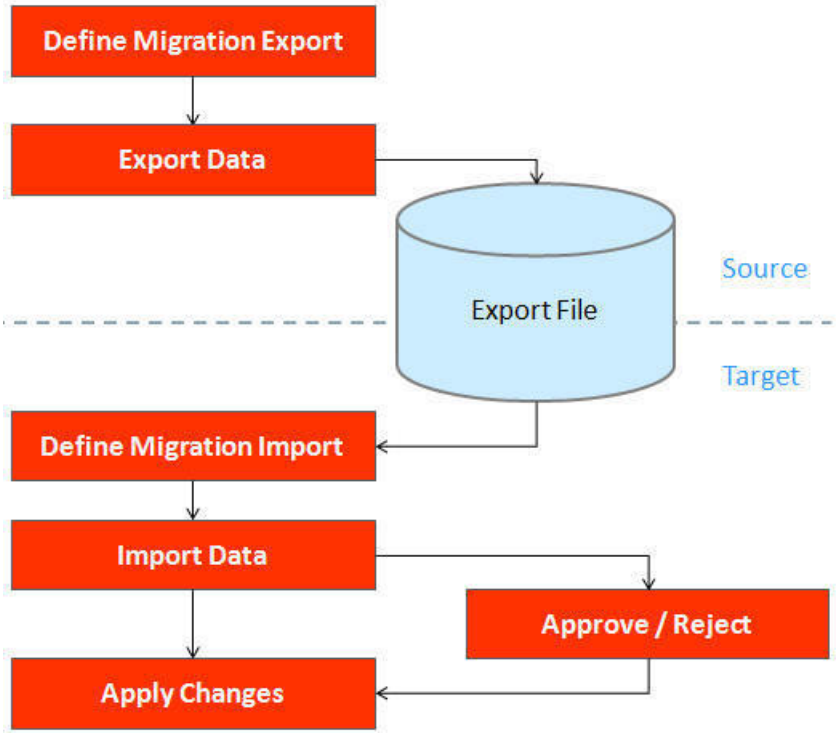
Identifying Tables to Exclude From Migrations

Some maintenance objects that are eligible to be migrated may include child tables that should not be included in the migration. For example, if an object includes log tables, the entries in the log should reflect the actions on the object in that system, and will be different between the source system and the target system. If you have a custom Maintenance Object that includes tables you don't wish to migrate (such as a log table), use the **Non-Migrated Table** option on the MO to specify this table. All child records for this table will also be ignored during migration.

Another use case to consider is a child "many-to-many" table that connects two administrative objects and exists in the maintenance object of both tables. The child table may be in both MOs for convenience sake, but it may be that one MO is considered more of a "driver" object and the other more of a subordinate. If you are doing a targeted (piecemeal) migration where you want to copy a subset of objects, you may want to only copy the driver object and its children and their data but not their children. For example, in Oracle Public Sector Revenue Management, a Form Type includes a collection of valid Form Change Reasons and in turn the Form Change Reason refers to its Form Types. If an implementation wants to do a targeted migration of a form type and include all its related information, including form rules and form change reasons, it does not want the migration of a form change reason to in turn copy all its form types (and their data).

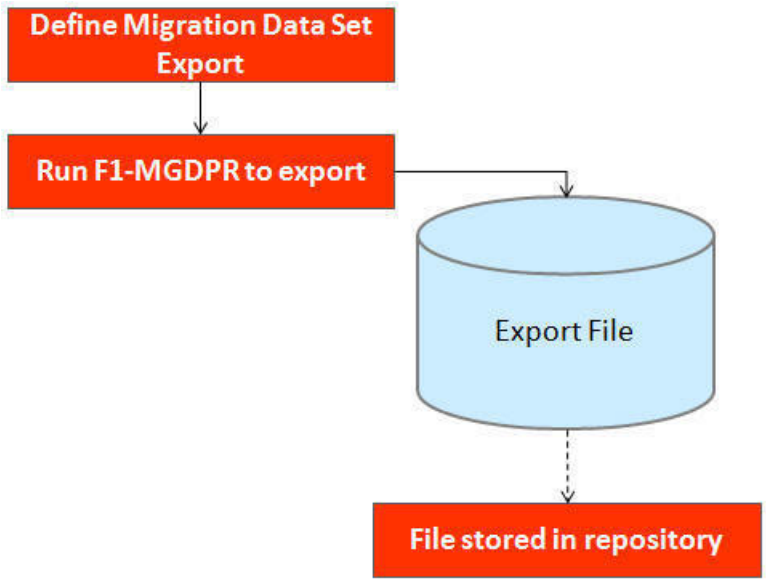
The CMA Execution Process

The following diagram illustrates a high-level view of the Configuration Migration Assistant execution process. The subprocesses illustrated here are described in more detail in the following sections.



Exporting a Migration

The migration export process begins in the source environment by defining a **Migration Data Set Export**, which specifies a defined **Migration Request** and provides a file name and description for the export file. After the data set is defined and saved, the **Migration Data Set Monitor** batch job can be submitted generate the export file.



The following topics provide more detail about this process.

Migration Data Set Export

To migrate data from one region to another, define a **Migration Data Set Export**. This establishes the export file name and identifies the migration request that includes the migration plans and selection criteria for the objects to include in the migration.

To view an existing migration data set export, navigate using **Admin** > **Migration Data Set Export** > **Search** . Use the query criteria to locate the desired data set.

To define a new migration data set export record, navigate using **Admin** > **Migration Data Set Export** > **Add** .

Note that you can also initiate the creation of an export data set from the *Migration Request* portal using the **Export** button.

The export requires the name of an existing **Migration Request**.

Enter a unique **File Name** for the export. Do not use spaces in the name, and do not enter the file extension or a path. The output location and file extension of the intended export file, which should appear in the **Export Directory** and **File Suffix** labels, are defined in the *Migration Assistant Configuration* master configuration record.

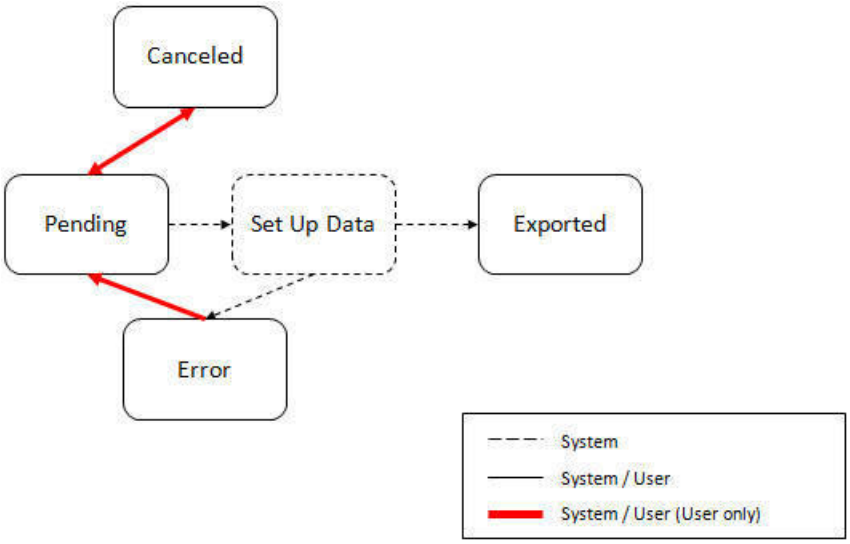
Enter an **Export Description** for the data set.

The **Source Environment Reference** is for information purposes. It should be populated with text that provides a meaningful description of the source environment. The default value is the URL of the source environment.

When the data set is complete, click **Save**.

Export Lifecycle

The following diagram describes the lifecycle of a Migration Data Set Export (data set).



The following points describe the lifecycle.

- The data set is created in **Pending** status.
- A user may choose to temporarily **Cancel** a pending data set to prevent it from being processed. The user can later return it to the Pending state when desired.
- The record remains in Pending until its monitor batch job is submitted. The **Migration Data Set Export Monitor** (F1–MGDPR) selects pending records and transitions them to **Set up data**. Refer to *Running Batch Jobs* for more information.
- Set up data is a transitory state that includes the algorithm that does the work of determining the objects to include in the export and group related objects together into a transaction. If everything is successful, the export file is written to the appropriate file location and the record transitions to **Exported**. If an error is detected, the process stops and the record transitions to **Error**.
- If a record is in error and it is possible to correct the error, the record may be transitioned back to Pending to try again.

When the process is marked as **Exported**, the export file can be imported into the target system.

NOTE: The export process creates a file, providing the benefits of having a standalone file. It can be stored in a version control system for audit purposes or provided to others for import purposes.

CAUTION: Under no circumstances should exported data files be edited manually. Doing so could cause data corruption when the file is applied to the target environment.

NOTE: The export functionality is supported using the business object **Migration Data Set Export** (F1-MigrDataSetExport). The expectation is that implementations will use the delivered base business object and its logic and will have no reason to implement a custom business object for the CMA export process.

Importing and Applying a Migration

The import process is broken down into four general steps: Import, Compare, Approve, Apply. The following points provide an overview of the steps.

- **Import**. The first step covers importing the file and creating appropriate Migration Import records in the target environment to facilitate the subsequent steps.

- **Compare**. The compare step reviews each object that is in the import file and compares the object in the import against the equivalent record in the target environment. The comparison step results in noting which objects are unchanged, which are new (and the appropriate SQL to insert them) and which objects are changed (and the appropriate SQL to update them). Based on user configuration at import time, the objects that qualify for the import may be in a state that requires review or may be pre-approved.
- **Approve**. Once the comparison is complete, the user should review the results. There may be records marked for review. All of these records must be approved or rejected before the import can proceed. When the user is satisfied with the results of the comparison and has completed the review, the import is marked to proceed to the Apply step.
- **Apply**. This is the final step and is the step where the records in the target environment are added or updated. Because of potential high volumes of data and because of possible dependencies between records, this step supports two levels of attempting to apply the records. There is an apply step at the object level and an apply step at the transaction level. This will be described in more detail below.

Import Step

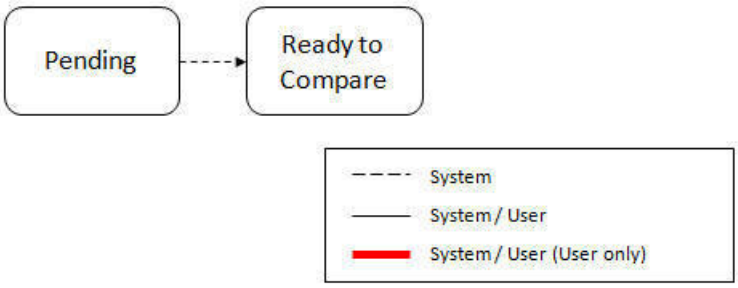
The import process starts with verifying the import directory configured in the *Master Configuration — Migration Assistant* for the target environment and ensuring that the exported file is located in that directory. Then, in the target environment, a **Migration Data Set Import** record should be created. The user indicates the file name.

In addition, the user decides what the default status should be for resulting objects.

- The **Default Status for Add** sets the default status for objects that are determined to be ***new*** during the import comparison process. The default is to automatically set new objects to **Approved** status. Other options are to set any new objects to **Rejected** or **Needs Review** status.
- The **Default Status for Change** sets the default status for objects that are determined to be ***changed*** during the import comparison process. As with new objects, the default for changed objects is **Approved**, with **Rejected** or **Needs Review** options available.

The file to import contains a list of all the objects included in the export. Any objects that the export step determined to be related have been grouped into “transactions”. Once the Migration Data Set Import (data set) is created, the next step is to read in the file and create Migration Transactions and Migration Objects.

The following is a portion of the Migration Data Set Import lifecycle as it pertains to the import step.



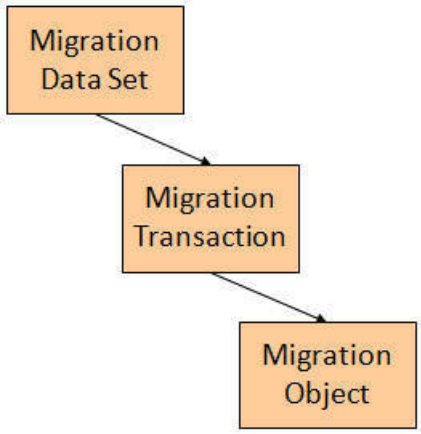
The following points describe the lifecycle.

- The data set is created in **Pending** status.
- The record remains in Pending until its monitor batch job is submitted. The **Migration Data Set Import Monitor** (F1–MGDIM) selects pending records and transitions them to **Ready to Compare**. Refer to *Import Process Summary* for more information.

  The Ready to Compare state has an algorithm that is responsible for reading the related import file and creating the migration transactions and migration objects. The data set remains in this state until the comparison step is complete.

NOTE: A user may choose to **Cancel** a data set. Refer to *Cancelling a Data Set* for more information.

The following diagram highlights the relationships of the resulting migration import records.



The migration transaction and migration object each have their own lifecycle that will help manage the subsequent compare and apply steps. At the end of the import step, the status values of the three types of records are as follows:

- Migration Data Set Import is in the **Ready to Compare** state.
- Migration Transaction is in the **Pending** state.
- Migration Object is in the **Pending** state.

NOTE: The import functionality is supported using business objects supplied by the base product. The expectation is that implementations will use the delivered base business objects and their logic and will have no reason to implement a custom business objects for the CMA import process. The base business objects are **Migration Data Set Import** (F1-MigrObjectImport), **Migration Transaction** (F1-MigrTransactionImport) and **Migration Object** (F1-MigrObjectImport).

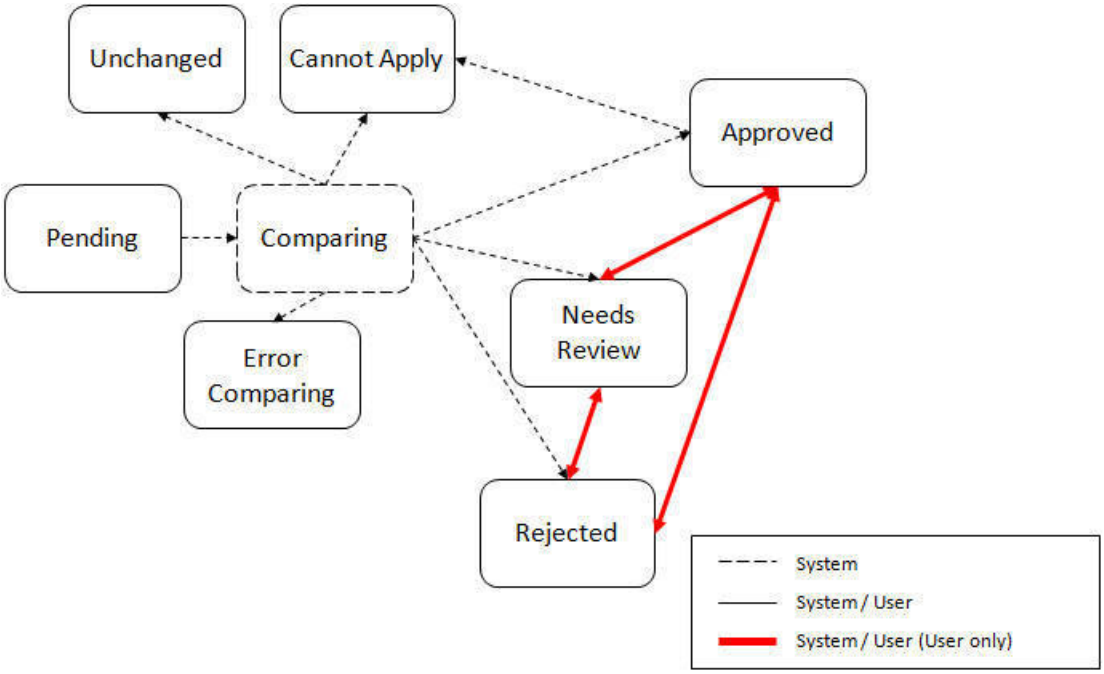
Compare Step

The import step results in the creation of one or more migration objects, one for each record selected in the export based on the export's migration request and its configuration. Related objects are grouped together in migration transactions. The next step in the import process is the Comparison step. In this step, the data captured by the import file for each object is compared to the view of that object in the target environment.

To cater for a possible large volume of objects, the comparison is done via a batch monitor. To aide in performance of the process, the monitor is performed on the migration objects so that it can be run multi-threaded. Once the objects are finished with the comparison, the migration transactions and the migration data set should be updated with an appropriate overall status before continuing to the next step. As a result, the comparison actually requires three steps: Migration Object Comparison, Migration Transaction Status Update and Migration Data Set Export Status Update. The steps are explained in detail in the following sections.

Migration Object Compare

This is the main step of the comparison. The **Migration Object Monitor** (F1–MGOPR) selects pending migration object records and transitions them to **Comparing**. This is a transitory state that includes the algorithm that does the work of comparing. There are various possible outcomes that could occur based on the logic in the algorithm. The following diagram illustrates a portion of the migration object lifecycle that pertains to comparison.



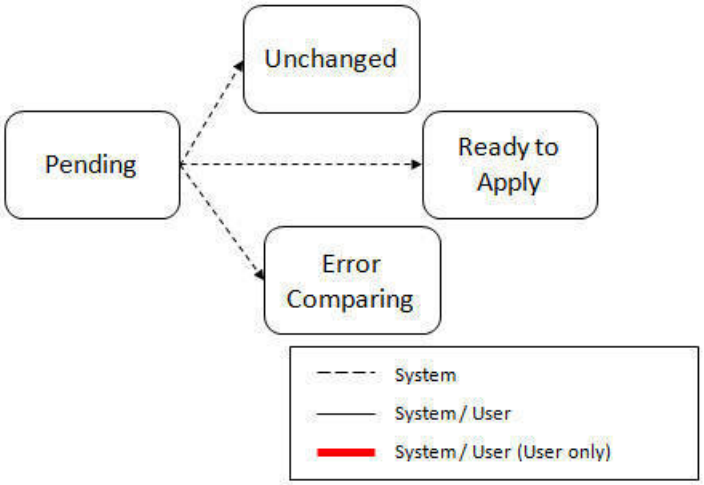
The following points describe the lifecycle.

- When **Pending** records are selected by the monitor batch job, it transitions to **Comparing** where the algorithm will determine the appropriate next state.
- If the record in the migration object is found in the target environment and the data is exactly the same, the record transitions to **Unchanged**.
- If the record in the migration object is found in the target environment and the data is different, the algorithm generates the appropriate SQL to be used later in the Apply step to update the record and then transitions to **Approved**, **Needs Review** or **Rejected** based on the Default Status For Change setting captured on the Data Set.
- If the record in the migration object is not found in the target environment, the algorithm generates the appropriate SQL to be used later in the Apply step to insert the record and then transitions to **Approved**, **Needs Review** or **Rejected** based on the Default Status For Add setting captured on the Data Set.
- If there is any issue with attempting to parse the object data from the import, the record transitions to **Error Comparing**.
- If there is any reason that the imported object is not valid for import, the record transitions to **Cannot Apply**. The log will be updated with the error that caused the record to transition to this state. An example is that perhaps the record was exported in a different version of the product and has additional elements that are not recognized in this version.

NOTE: Refer to *Cancelling a Data Set* for information about cancelling a data set and its impact on its related objects.

Migration Transaction Status Update

After the import step, the migration transaction remains in the Pending state until all its objects have completed the comparison step. At that point, the status of the transactions should be updated based on the results of their objects. The **Migration Transaction Monitor** (F1–MGTPR) selects pending migration transaction records and runs its monitor algorithms. There are various possible outcomes that could occur based on the logic in the algorithms. The following diagram illustrates a portion of the migration transaction lifecycle that pertains to comparison.



The following points describe the lifecycle possible next states after Pending.

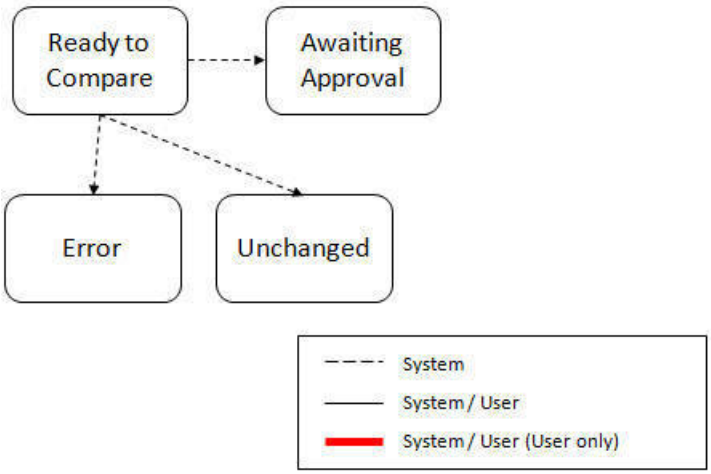
- If any related migration object is in the Error Comparing state, the transaction transitions to **Error Comparing**.
- If all related migration objects are in the Unchanged state, the transaction transitions to **Unchanged**.
- Otherwise, the transaction transitions to **Ready to Apply**. This means that at least one object is in an “apply-able” state.

The transaction remains in the **Ready to Apply** state until a user has approved the data set to move to the Apply step and the transaction’s related objects have attempted to apply themselves. This is described in more detail below.

NOTE: Refer to *Cancelling a Data Set* for information about cancelling a data set and its impact on its related objects.

Migration Data Set Import Status Update

Once all the objects and all transactions have been updated via the previous two steps, the migration data set export must be updated based on the results of their transactions. The **Migration Data Set Import Monitor** (F1-MGDIM) selects Ready to Compare data sets and runs its monitor algorithms. Note that this is the same monitor process that is used to select Pending data sets. There are various possible outcomes that could occur based on the logic in the algorithms. The following diagram illustrates the portion of the migration transaction lifecycle that pertains to comparison.



The following points describe the lifecycle possible next states after Ready to Compare.

- If any related migration transactions is in the Error Comparing state, the data set transitions to **Error**.
- If all related migration transactions are in the Unchanged state, the data set transitions to **Unchanged**.

- Otherwise, the transaction transitions to **Awaiting Approval**. This means that there are no errors and at least one object is in an “apply-able” state.

The data set remains in the **Awaiting Approval** state until a user decides that the data set and all its records are ready to progress to the Apply step.

NOTE: A user can choose to cancel a data set at any time while it is in progress. Refer to *Cancelling a Data Set* for more information.

Approval Step

Once the comparison is complete and the data set transitions to the Awaiting Approval state, a user needs to progress the data set to **Apply Objects** to trigger the Apply step. The following points describe steps a user may take during the approval step.

- If the data set configuration for the Default State for Add and Change was set to **Approved**, then any migration object that is determined to be eligible for the Apply step will be in the Approved state. In this situation, a user may want to review the data set and its transactions and objects to see verify that the results make sense. At that time, the user is able to move an object to Needs Review or Rejected as appropriate.
- If the data set configuration for the Default State for Add and Change was set to **Needs Review** for either option, then each migration object in the Needs Review state must be reviewed and the user must either Reject or Approve each object before moving to the Apply step.
- If the data set configuration for the Default State for Add and Change was set to **Rejected** for either option, the assumption is that the rejected records don’t need to be reviewed. But if a user finds a rejected record that shouldn’t be rejected, it may be transitioned to Approved (or Needs Review) as appropriate.

Once the user is comfortable with the data set’s results and no more objects are in the Needs Review state, the user should transition the record to **Apply Objects**. This will initiate the Apply step.

NOTE: Refer to *Maintaining Import Data* for details about the pages provided to help the user review a data set and its transactions and objects to help in the approval step.

NOTE: A user can choose to cancel a data set at any time while it is in progress.

Apply Step

The apply step is the step where records in the target environment are added or updated. Like the comparison step, the apply step is actually multiple steps to optimally handle high volume and dependencies between records as smoothly as possible. Before explaining the apply steps in detail, the following points highlight the type of data that may be included in a given data set.

1. Records that have no foreign keys and therefore no dependencies on other records. Examples: Message, Display Profile.
2. Records that have foreign keys that may already be in the target. Examples: Algorithms for existing algorithm types, To Do Roles for existing To Do Types.
3. Records that have foreign keys that are new records but also part of the migration. CMA detected the relationship at export time and grouped the objects in the same transaction. Example: Script-based Algorithm Type where the script is also in the migration.
4. Records that have foreign keys that are new records but also part of the migration. CMA did not detect the relationship. This may occur if the reference the foreign key is in a CLOB or parameter column and the migration plan did not include an instruction to explicitly define the relationship. Example, a Zone that references a visibility script.

5. Records that have circular references where both records are new and are part of the migration. CMA detected the relationship at export time and grouped the objects in the same transaction. Example: plug-in Script for a BO plug-in spot. The script references the BO and the BO references an algorithm for the script's algorithm type.

To handle high volume data, the first step in the apply process is to perform the apply logic at the migration object level via a multi-threaded batch job. This should result in all records in categories 1 and 2 above being applied successfully.

For records in categories 3 and 4 above, if a record with a foreign key is added or updated before its related record, the validation will fail. However, if the related record is added first and then the record referring to it is added, validation will pass. Because the migration objects are not ordered, the multi-threaded batch process may not process records in the desired order. To overcome this potential issue, the Apply step has special functionality, described in detail below.

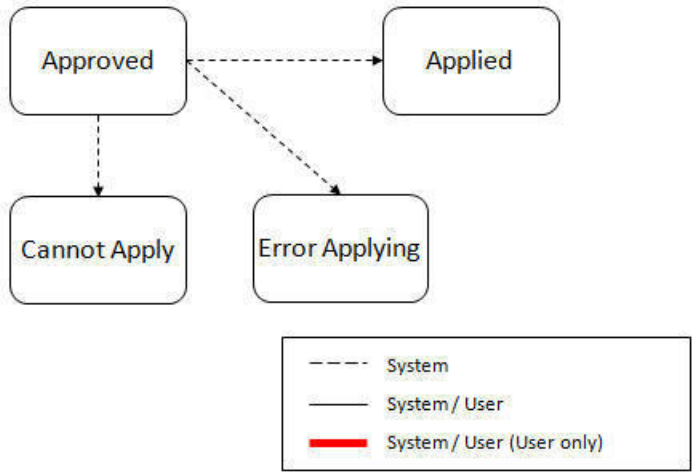
For records in category 5 above, the circular reference will mean that the apply process at the object level will not successfully add or update these records. The apply process at the transaction level will cover these records. This is described in detail below.

Apply Objects

Once the Data Set is in the state of **Apply Objects**, the **Migration Object Monitor — Apply** process (F1–MGOAP) runs to attempt to apply the objects. The background process in conjunction with the Apply algorithm have special functionality to ensure records in categories 3 and 4 (above) successfully apply during this step:

- The **Migration Object Monitor — Apply** process is a special one that continually re-selects records in the **Approved** state until there are no more eligible records to process.
- When an error is received in the Apply Object algorithm, the algorithm increments an "iteration count" on the migration object record. If the iteration count does not exceed a maximum count (noted in the algorithm), the object remains in the **Approved** state and is eligible to be picked up for processing again. If the iteration count exceeds the maximum defined in the algorithm, the record transitions to the **Error Applying** state.

The following diagram is the portion of the migration object lifecycle that pertains to the Apply step. Note that this diagram is not complete. A subsequent section provides more information about resolving errors.



At the completion of the Apply monitor process, typically the objects will be in the **Applied** state or the **Error Applying** state. The records in the Error Applying state are in that state for one of two reasons.

- They are in category 5 described above where the records have a circular reference with another record. For this scenario, the Apply Transactions step described below should successfully apply the records.
- There is some other error that is unrelated to the records in the current migration. In this case, manual intervention is required. Refer to the Resolving Errors section below for more information.

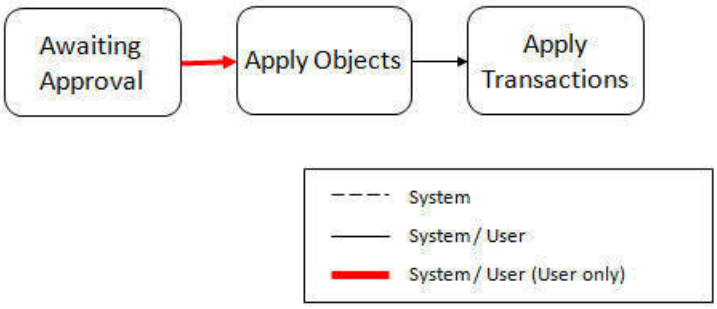
As shown in the diagram, the Apply Objects algorithm may also detect a reason that the object cannot be applied. This may occur if the object in the target environment has been updated since the comparison step, making the SQL captured at that

point no longer applicable. If this occurs, after the current migration is fully applied, the original file may imported again, and new comparisons can be generated and applied.

Apply Transactions

Ideally, after the Apply Objects step, all the objects are **Applied** or are in **Error Applying** due to the "circular reference" situation. The typical next step is to turn over responsibility to the transactions. The migration transactions can then attempt to apply their objects in bulk.

In order to ensure that multiple background processes are not trying to select migration objects to run the Apply step, the transactions are only eligible to attempt to "apply my objects" if the Data Set is in the **Apply Transactions** state.



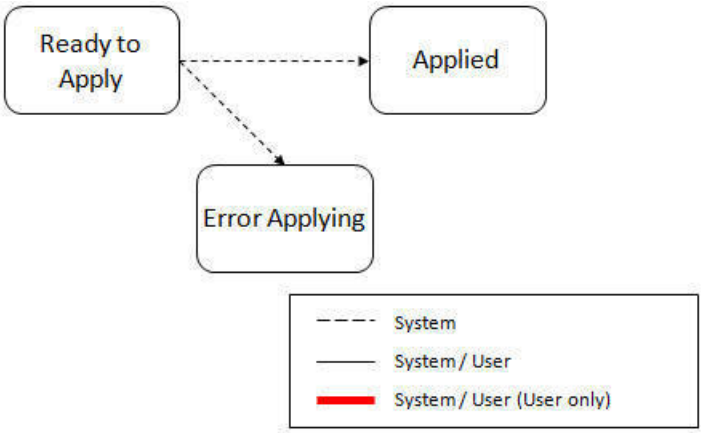
A monitor algorithm (executed by the data set monitor batch process) on the Apply Objects state checks to see if all migration objects are no longer **Approved** and do not have any records in **Error Applying**. If so, it automatically transitions the record to the **Apply Transactions** state.

If any objects are in **Error Applying**, the monitor algorithm does not automatically transition the record. In that case, a user must decide if the record should transition to **Apply Transactions**. The Resolving Errors section below describes alternative steps that the user may take if there are errors.

Once the Data Set is in the state of **Apply Transactions**, the **Migration Transaction Monitor — Apply** process (F1–MGTAP) runs. It attempts to apply the transaction's objects. If no migration objects are in error, the migration transaction simply transitions to **Applied**. If any of the migration objects are in **Error Applying**, the background process and the Apply algorithm have special functionality to try to overcome dependencies in migrated objects:

- The Apply algorithm selects all migration objects in error and performs all their SQL, then validates all the records. If there are objects in the transaction with circular references, they should pass validation at this point.
- Because there may still be some dependencies across transactions, similar error handling described in the Apply Objects step occurs here. When an error is received in the Apply Transaction's Object algorithm for any of the objects in the transaction, the algorithm increments an "iteration count" on the migration transaction record. If the iteration count does not exceed a maximum count (noted in the algorithm), the transaction remains in the **Ready to Apply** state and is eligible to be picked up for processing again. If the iteration count exceeds the maximum, the record transitions to the **Error Applying** state. Note that if any objects in the transaction are in error, none of the objects are applied. They all remain in error.
- The **Migration Transaction Monitor — Apply** process is a special one that continually re-selects records in the **Ready to Apply** state until there are no more eligible records to process.

The following diagram is the portion of the migration transaction lifecycle that pertains to the Apply step illustrating the points above. Note that this diagram is not complete. A subsequent section provides more information about resolving errors.



If at the end of the transaction level Apply process there are transactions in error (and therefore there are still objects in error), a user must review the errors and determine how to fix them. Refer to the Resolving Errors section below for more information.

Resolving Errors

As mentioned in the previous sections, errors may be received after the Apply Objects process runs and the system will not automatically transition the data set to the **Apply Transactions**. Rather, a user must make the decision. This allows for the user to review the errors and make a decision:

- If the errors appear to be dependency related, the user can decides to let the "transactions apply their objects" and transition the data set to **Apply Transactions**, described above.
- If the errors appear to be related to an outside issue that can be manually resolved, the user may choose to fix the issue and redo the Apply Objects step.
- The user may also decide to reject one or more objects to remove them from the migration.

After the Apply Transactions step, if there are still errors, a user must review the records and determine how to proceed

- The user may decide to reject one or more objects to remove them from the migration.
- The user may manually resolve an issue external to the migration and then decide to do one of the following:
  - Redo the **Apply Objects** step. This is recommended if there are a large number of Objects still in error and not a large number of dependencies expected. The benefits of running the Apply Objects multi-threaded will ensure that the process runs efficiently.
  - Redo the **Apply Transactions** step.

Because the objects and transactions are in Error Applying, in order to “retry” the Apply step after manually fixing an error, the system needs to move the records back to the state that allows them to be picked up by the appropriate Apply process. For migration objects, records need to be moved back to **Approved**. For migration transactions, records need to be moved back to **Ready to Apply**. This is accomplished with a Retry Count. This is different from the Iteration count that is used during the apply steps to handle the iterative selection logic. The following points describe the Retry logic for migration objects.

- When the Migration Data Set Import record first enters the **Apply Objects** state, its Object Retry Count is set to 1.
- When Migration Objects first enter the **Approved** state, their Object Retry Count is set to 1.
- When migration objects transition to **Error Applying**, it remains in this state as long as the object’s Retry Count matches the data set’s Object Retry Count (or it is successfully applied via the Apply Transactions step).
- If a user decides to **Retry Objects** (using an action button on the Migration Data Set Import page), the data set’s Object Retry Count is incremented and the Data set returns to the **Apply Objects** state.

- At this point, the monitor on the **Error Applying** state for the objects detects that the object's Retry Count doesn't match the data set's Object Retry Count and that triggers the transition back to **Approved**. This increments the object's retry count to match the data set's. Now the objects are eligible to be picked up by the object level Apply process.

Analogous logic exists for the migration transactions.

- When the Migration Data Set Import record first enters the **Apply Transactions** state, its Transaction Retry Count is set to 1.
- When Migration Transactions first enter the **Ready to Apply** state, their Transaction Retry Count is set to 1.
- When migration transactions transition to **Error Applying**, it remains in this state as long as the transaction's Retry Count matches the data set's Transaction Retry Count.
- If a user decides to **Retry Transactions** (using an action button on the Migration Data Set Import page), the data set's Transaction Retry Count is incremented and the Data Set returns to the **Apply Transactions** state.
- At this point, the monitor on the **Error Applying** state for the transactions detects that the transaction's Retry Count doesn't match the data set's Transaction Retry Count and that triggers the transition back to **Ready to Apply**. This increments the transaction's retry count to match the data set's. Now the transactions are eligible to be picked up by the transaction level Apply process.

Finalize Apply Step

Once all the migration objects for a migration transaction are in a final state (**Applied**, **Rejected** or **Cannot Apply**), the migration transaction transitions to the **Applied** state. Once all the migration transactions are in the **Applied** state, the Migration Data Set record transitions to the **Applied** and the import is complete.

> **NOTE:** To review the full lifecycle for each record, refer to the Business Object — Summary tab in the application metadata for the base business objects **Migration Data Set Import** (F1-MigrObjectImport), **Migration Transaction** (F1-MigrTransactionImport) and **Migration Object** (F1-MigrObjectImport).

Adjusting Data While Importing

Some records may have data that is specific to the environment it is in and won't apply in the target environment. In such cases, an algorithm plugged into the migration plan instruction may be used to adjust the data when importing. This algorithm is executed in the apply step after the record is inserted but before the validation is executed.

Some examples of records that may require import algorithms.

- Batch Control references its next batch sequence number. This number is only relevant with respect to its environment. The instruction for a batch control can include an algorithm to not overwrite the batch sequence number when copying a batch control.
- Some products include administrative objects that reference a master data object. Master data objects are not copied as part of CMA. An import algorithm may be used to adjust the referenced master data foreign key when importing, for example to reset it (or not overwrite when updating). If the algorithm knows how to find the appropriate master data record to link, that may also be included.

Import Process Summary

The following table summarizes the steps required to complete the import process from start to finish. It highlights what steps are manual and what steps are performed by a batch monitor process. For the Apply steps, there are two parts where multiple next actions are possible. The possible next actions have the same sequence with a letter following the sequence. For each step, the table highlights the Next Action sequence that would occur.

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
Import	**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 or Error Comparing	21
Compare	**21**	**Migration Transaction Status Update**	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply, Unchanged or Error Comparing	22
Compare	**22**	**Migration Data Set Export Status Update**	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Awaiting Approval, Unchanged or Error	30
Approval	**30**	**Review comparison results, approve / reject as needed**	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
Apply	41a	Data Set Transition - auto transition to Apply Transactions. Only applicable if no migration objects are in Error Applying	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Transactions	42
Apply	**41b**	**Data Set Transition - manual transition to Apply Transactions.**	Manual	Migration Data Set Import - click Apply Transactions		Migration Data Set Import - Apply Transactions	42

Step	Seq	Action	Manual / Batch	Portal — Action	Batch Control	Record Impacted — Resulting Status	Next Action Sequence
		Occurs if user reviews errors and determines that they may be resolved in the Apply Transaction step.					
Apply	41c	Data Set Transition - manual transition to Retry Objects. Occurs if user decides to fix an external error and wants to try the batch level Apply again.	Batch	Migration Data Set Import - click Retry Objects		Migration Data Set Import - Apply Objects	44
Apply	**42**	**Apply Transactions**	Batch		F1-MGTAP - Migration Transaction Monitor - Apply	Migration Transaction - Applied or Error Applying Migration Object - Applied or Error Applying	43
Apply	**43a**	**Data Set Transition - auto transition to Applied** Applicable if all transactions are Applied	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Applied.	N/a
Apply	43b	Data Set Transition - manual transition to Retry Objects. Occurs if user decides to fix an external error and wants to try the batch level Apply again at the Object level.	Manual	Migration Data Set Import - click Retry Objects		Migration Data Set Import - Apply Objects	44
Apply	43c	Data Set Transition - manual transition to Retry Transactions. Occurs if user decides to fix an external error and wants to try the batch level Apply again at the Transaction level.	Manual	Migration Data Set Import - click Retry Transactions		Migration Data Set Import - Apply Transactions	45

Step	Seq	Action	Manual / Batch	Portal — Action	Batch Control	Record Impacted — Resulting Status	Next Action Sequence
Apply	44	Move Objects from Error Applying back to Approved. Occurs if user chose to Retry Objects.	Batch		F1-MGOPR - Migration Object Monitor	Migration Object - Approved	40
Apply	45	Move Transactions from Error Applying back to Ready to Apply. Occurs if user chose to Retry Transactions.	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply	42

The following table summarizes the batch monitor jobs that are used in the import process. You can see that there are special monitor processes for the Apply step for both the Object and Transaction. However, for all other states that have monitor logic, the standard monitor process for that MO is used.

Batch Control	Description	Comments
F1-MGDIM	Migration Data Set Import Monitor	Processes data set records in the following states: • Pending • Ready to Compare • Apply Objects • Apply Transactions
F1-MGTPR	Migration Transaction Monitor (Deferred)	Processes transaction records in the following states: • 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.
F1-MGOPR	Migration Object Monitor	Processes object records in the following states: • 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.

Refer to *Running Batch Jobs* for more information about managing the batch jobs.

Cancelling a Data Set

A user may choose to **Cancel** a data set to prevent it from being processed at any point during the process.

If related migration transactions or migration objects have already been created, they will not be canceled as part of the data set getting canceled (due to possible high volumes of related records). They will be canceled the next time an appropriate monitor batch process runs. The child records checks to see if the data set has been canceled prior to any state transition.

Additional Note Regarding Imports

The following points describe miscellaneous comments related to Migration Import.

- CMA relies on the fact that database referential integrity constraints are not in place, and that the SQL statements can be run in any order within the transaction. Any archiving solution that requires referential integrity constraints (such as Information Lifecycle Management) would not be possible on this data. Given that CMA migrations comprise administrative data and not transactional data, this should be a reasonable exception.
- The validation that is performed is only via the **Page Validate** service. BO validation algorithms are not executed. Page validation does not include validation of the business object against the schema (for example, for required fields, field sizes, etc.).
- All migration requests can be exported at the same time. On the import side, however, you should consider importing, reviewing, and applying an entire file/data set before moving on to the next one. The reason is that if objects are included in more than one file, two sets of "inserts" will be generated, but only the first will succeed. The second will cause an insert error on the object, and the transaction would be marked with status "Error Applying". If instead you wait until the first file is applied before importing the second, the second data set will not generate any SQL for the object, since it has already been inserted. It's a matter of efficiency: If you first import all files and then try to apply all, you'll have to identify the duplicated object as an error and then mark the object as rejected before applying the transaction.

Caching Considerations

Because CMA updates administrative data that is usually read from a cache, a thread pool with the L2 cache disabled is required during the execution stages of migration. Contact your system administrator and ask for the name of a thread pool with the L2 cache disabled. This thread pool name must be entered in the batch controls used for CMA processing in the **Thread Pool Name** parameter.

Refer to *Running Batch Jobs* for more information about running CMA batch jobs.

The *Batch Server Administration Guide* also contains information on preparing and implementing thread pools and turning off the L2 cache.

Also note that after a successful migration, the target region now has new administrative data which needs to be part of the L2 cache. The existing thread pool workers should be restarted by your system administrators.

Maintaining Import Data

This section describes the portals provided to add, view and maintain migration import data.

Migration Data Set Import

Use the Migration Data Set Import portal to view and maintain migration data set import records. Navigate using **Admin** > **Migration Data Set Import** > **Search** .

Migration Data Set Import Query

Use the query portal to search for an existing Migration Data Set Import record. Once a record is selected, you are brought to the maintenance portal to view and maintain the selected record.

In addition, the query provides an option to specifically search for data sets that have either objects in error or transactions in error.

Migration Data Set Import Portal

This page appears when viewing the detail of a specific migration data set import record.

The topics in this section describe the tabs that appear on the portal.

Migration Data Set Import - Main

This portal appears when a migration data set import record has been selected from the query portal. The topics below describe the base-package zones that appear on the portal.

Migration Data Set Import

The Migration Data Set Import zone contains display-only information about the selected record.

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

Migration Data Set Transactions

This zone is visible once the *Import Step* has occurred and lists all the transactions that are related to the data set. To see more information about a specific migration transaction, click the hypertext for its ID. This brings you to the *Migration Transaction* portal.

Migration Data Set Impacted Object Summary

This zone is visible once the *Import Step* has occurred and lists the objects that are related to the data set. To see more information about a specific migration object, click the hypertext for its ID. This brings you to the *Migration Object* portal.

Migration Data Set Objects in Error

This zone is only visible if the data set's status is **Apply Objects**, and there are objects in the status **Error Applying**. It indicates the error for each object. A user may use this zone to review errors after the monitor batch job to apply objects completes. Using the error information shown, the user can choose to transition the record to **Error Applying** or manually fix the cause of the error and click **Retry Objects**.

NOTE: Refer to *Apply Step* for more information about resolving errors.

Migration Data Set Transactions in Error

This zone is only visible if the data set's status is **Apply Transactions**, and there are transactions in the status **Error Applying**. It indicates the error for each object. A user may use this zone to review errors after the monitor batch job to apply transactions completes. The errors received when attempting to apply objects at the transaction level may differ from those received when attempting to apply objects at the object level. A transaction log is created for each object error received and these exceptions are shown in this zone.

NOTE: Refer to *Apply Step* for more information about resolving errors.

Migration Data Set Import - Log

Navigate to the Log tab to view the *logs* for the migration data set import record.

Migration Transaction Portal

This page appears after drilling into a specific migration transaction from the migration data set portal or from the migration object portal.

The topics in this section describe the tabs that appear on the portal.

Migration Transaction - Main

The topics below describe the base-package zones that appear on the portal.

Migration Transaction

The Migration Transaction zone contains display-only information about the selected record.

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

Migration Transaction Objects

This zone lists the objects that are related to the data set. To see more information about a specific migration object, click the hypertext for its ID. This brings you to the *Migration Object* portal.

Migration Transaction - Log

Navigate to the Log tab to view the *logs* for the migration transaction.

Migration Object Portal

This page appears after drilling into a specific migration object from the migration data set portal or from the migration transaction portal.

The topics in this section describe the tabs that appear on the portal.

Migration Object - Main

The topics below describe the base-package zones that appear on the portal.

Migration Object

The Migration Object zone contains display-only information about the selected record.

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

Migration Object - Log

Navigate to the Log tab to view the *logs* for the migration object.

Running Batch Jobs

There are several batch jobs that are part of the CMA process, especially the import step. And in some cases, a single batch jobs may process multiple states in the same business object lifecycle. Implementations must decide the best way to manage the batch job submission depending on how they plan to work.

- **Batch scheduler**. If an implementation wishes to put these batch jobs in the batch scheduler, a given job may need to be included several times to manage progressing the records to completion.
- **Timed Batches**. The batch controls can be configured as timed batches so that they run every N minutes based on the setting. This allows for the batch jobs to run periodically and process whatever is ready. A user doesn't have to manually submit a batch request. Navigate to **Admin** > **Batch Control** > **Search** and select the appropriate batch controls. For each one, change the Batch Control Type to **Timed**. Fill in the additional information that appears for timed batches.
- **Manual submission**. The user managing the CMA import process submits the appropriate batch jobs on demand when a particular step is ready. Navigate using **Menu** > **Batch** > **Batch Job Submission** > **Add** , select the appropriate batch control and fill in the parameters as needed.

Note that for the CMA batch processes, especially the "apply" batch processes, the setting of the **Thread Pool Name** parameter is important. This parameter specifies the thread pool that will be executed by the batch. Since the CMA apply processes update administrative data that is usually read from a cache during batch processing, the thread pool named here must be one with the L2 cache disabled. In addition, after successfully applying the migrated data, the L2 cache of all thread pools must be refreshed. For more information, see *Caching Considerations*.

Refer to the parameter descriptions in the batch control metadata for more information about filling in the parameters.

For additional details on submission controls, refer to the topic *Batch Job Submission - Main* in the Batch Jobs section.

CMA Reference

This section provides additional reference information.

Framework-Provided Migration Configuration

This topic describes special information relating to migration objects provided for use by CMA in the product. Additional objects may be provided by your specific product. Any special information for objects is provided separately in each product's documentation.

You can see all currently available objects and their descriptions by performing a search in the **Migration Plan Query** or **Migration Request Query** pages.

The following points highlight some information about the Framework-provided migration plans and migration requests:

- Fields and characteristic types are not migrated with the object unless specifically indicated.
- The **Application Service** used by an object is migrated only if it is CM-owned.
- The **F1-Language** migration plan should **never** be used in a migration request with other objects. Defined constraints will cause CMA to link all objects with a language table—which is almost all of the system admin data—into one large transaction. Attempting to update and commit all the changes for this one large transaction may cause performance issues and/or out of memory errors. For this reason, this MO has its own migration request, **F1-Languages**.
- The migration request **F1-SchemaAdmin** defines migration plans related to data that is commonly used in schema-based objects.

- The **Batch Control** object optionally references a User. If this user does not exist on the target system, CMA cannot apply the requested changes.
- The base migration plans for MO and BO include instructions to copy option types that use foreign key references to refer to other objects. Note that the data stored in the options are not validated, so defining these instructions is not required when doing wholesale migrations. However, including subordinate instructions for foreign key references is useful for piecemeal migrations to ensure that the related data is included in the migration. If you add additional MO or BO option types that use foreign keys and you want to support piecemeal migrations, you must create custom migration plans and requests for MO and BO, respectively to include these referenced objects in the migration plan. Note that you do not need to duplicate the instructions in the base migration plans. You may define the additional migration plans to only have the additional custom option types. When submitting a migration request for MO or BO you must include both the base migration plans and the custom migration plans in the request.
- The migration plans provided by CMA migrate scripts, schema-based objects and zones, and, through constraints, some of the associated data associated with them. However, data specified through alternate formats (such as through **Edit Data** steps in scripts, referenced in schemas for schema-based objects, or data from mnemonics in zone parameters, etc.) are not identified and combined in the same transaction. It's important to note that this could cause validation errors during **Apply**, and may require retrying or migrating the additional data separately. See *Apply Step* for a description of how to reapply the import should an **Applied With Errors** result occur.
- There are two migration plans for **Scripts**. The migration plan **F1-ScriptOnly** migrates just the script and its **Application Service** (provided the Application Service is CM-owned). This is used in the **F1-FrameworkAdmin** migration request, which migrates all admin objects. The migration plan **F1-Script** includes most related objects, but does not migrate any objects referenced in the edit data area steps. It does not move the **Function** maintenance object (which has been deprecated). This migration plan is not included in any base migration requests. It may be included in any appropriate custom piecemeal migration request where scripts and related data should be migrated.
- If your implementation includes a **Feature Configuration** setting for the **F1_DBCONINFO** entry that will be included in a migration request, be sure that the import user on the target region has the appropriate security rights to this entry (**Super User** access mode for the Feature Configuration application service (**CILTWSDP**).
- All of the provided Framework-owned migration requests include the SQL statement "1=1" so that all records in the plan are migrated. If an implementation wants to limit a migration to a subset of records, the prepackaged migration request may be duplicated and then customized to modify the key selection criteria.
- Many of the XAI-related maintenance objects include references to environment-specific data, and data should be migrated with extreme care. Migration plans are not provided for MOs that contain mostly environment-specific data. Migration plans are provided for the following XAI-related maintenance objects, but these plans are not included in the **F1-XAI-Admin** migration request: **External System**, **XAI Receiver**, **Message Sender**, and **XAI Route Type**.

Migration Requests for Wholesale Migrations

The following framework provided migration requests may be used in a wholesale migration. Refer to your specific product's CMA documentation for its recommendation on which migration requests to use for a full migration of framework and product administrative tables.

- The **F1-SchemaAdmin**(Framework Schema Admin) migration request. This request contains migration plans for objects needed by schema based objects, such as Field and Lookup. It has few dependencies on its data and many other objects include dependencies to it.
- The **F1-FrameworkAdmin** (Framework Admin) migration request. This includes migration plans for most widely used maintenance objects Your product may have chosen to incorporate migration plans from this migration request directly into one of its owned migration requests.
- The **F1-GeneralSystemOptions** (General System Objects) migration request. This includes migration plans for business related objects that may or may not be applicable to all products. Your product may have chosen to incorporate a subset of the migration plans from this migration request directly into one of its owned migration requests.

For each migration request determined to be needed for a wholesale migration, create a migration data set export record and process it as documented in **Exporting a Migration**.

Executing Piecemeal Migrations

Piecemeal migrations involve copying a targeted set of data. It may be that it includes a subset of maintenance objects (for example all Activity Types or all Asset Types) or it may further limit the records within one or more maintenance objects (for example, all ‘electric’ rates). Many piecemeal migrations are ad-hoc, depending on the specific requirements at the time.

If your product has provided migration requests for piecemeal migrations out of the box, it would they would be defined to copy the appropriate type of data. However, the selection criteria would most likely not be limited to a type of record. As such, it could be used as is for a “copy all data in these maintenance objects” migration. For example, the framework provides a migration request for Security Configuration. This may be used to copy all the data in the various security tables. Oracle Public Sector Revenue Management provides a migration request to copy form types and its related data.

If your implementation wants to copy only a subset of data for maintenance objects covered by a base provided migration request, do the following:

1. Duplicate the base migration request.
2. Find the migration plan for the object or objects where you want to limit the data. Enter appropriate Key Selection criteria to select the desired data.
3. Use that migration request to request a migration data set export request.

If your piecemeal migration requirement is not satisfied by a base migration request that may be used as a starting point, review the base migration plans for the records that should be copied. Determine if the migration plans have the desired subordinate instructions to copy (or not copy) the related data as desired.

- If existing migration plans satisfy your requirements, create a migration request that includes those plans (and desired key selection as needed). Use that migration request to create a migration data set export request.
- If existing migration plans do not satisfy your requirements, create migration plans as needed. Create a migration request that includes those plans (and desired key selection as needed). Use that migration request to create a migration data set export request.

Chapter 18

Configuring Facts

Fact is an optional configuration tool for simple workflow-type business messages or tasks. The base package does not provide a dedicated Fact user interface because fact is generic by design. Implementations configure their own user interface to visualize the desired custom business process. The topics in this section describe the generic Fact entity and how it can be customized.

Fact Is A Generic Entity

The Fact maintenance object is a generic entity that can be configured to represent custom entities and support automated workflows for a variety of applications. Each fact references a business object to describe the type of entity it is. A status column on the fact may be used to capture its current state in the processing lifecycle controlled by its business object.

The maintenance object also supports a standard characteristic collection as well as a CLOB element to capture additional information.

Where Used

Follow this link to open the data dictionary where you can view the tables that reference *F1_FACT*

Fact's Business Object Controls Everything

A fact's business object controls its contents, lifecycle and various other business rules:

- Its schema defines where each piece of information resides on the physical Fact maintenance object.
- If may define a lifecycle for all fact instances of this type to follow. Each fact must exist in a valid state as per its business object's lifecycle definition.
- It may define validation and other business rules to control the behavior of facts of this type.

FASTPATH: For more information about business objects, refer to *The Big Picture of Business Objects*.

Fact Supports A Log

The Fact maintenance object supports a log. Any significant event related to a Fact may be recorded on its log. The system automatically records a log record when the fact is created and when it transitions into a new state. In addition, any custom process or manual user activity can add log entries.

FASTPATH:

Refer to *State Transitions Are Audited* for more information on logging. For more information about the various configuration tools available, refer to *Configuration Tools*. For more information about user interfaces, refer to *Configurable User Interface Features*.