Oracle® Fusion Functional Setup Manager Developer's Guide

11*g* Release 6 (11.1.6)

Part Number E23515-06

September 2012

Oracle® Fusion Functional Setup Manager Developer's Guide

Part Number E23515-06

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

Author: Chitra Mitra

Contributors: Nihar Barik, Maneesha Damle, Angayarkanni Dhanapal, Patricia Perdomo, Jui Prabhudesai, Vivek Ranjan, Yunjie Yu

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

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

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

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle and Java are registered trademarks of Oracle Corporation and its affiliates. Other names may be trademarks of their respective owners.

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

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface

This Preface introduces the guides, online help, and other information sources available to help you more effectively use Oracle Fusion Applications.

Oracle Fusion Applications Help

You can access Oracle Fusion Applications Help for the current page, section, activity, or task by clicking the help icon. The following figure depicts the help icon.

With a local installation of help, you can add custom help files to replace or supplement the provided content. Help content patches are regularly made available to ensure you have access to the latest information. Patching does not affect your custom content.

Oracle Fusion Applications Guides

Oracle Fusion Applications guides are a structured collection of the help topics, examples, and FAQs from the help system packaged for easy download and offline reference, and sequenced to facilitate learning. You can access the guides from the **Guides** menu in the global area at the top of Oracle Fusion Applications Help pages.

Note

The **Guides** menu also provides access to the business process models on which Oracle Fusion Applications is based.

Guides are designed for specific audiences:

- **User Guides** address the tasks in one or more business processes. They are intended for people who perform these tasks, and managers looking for an overview of the business processes. They are organized by the business process activities and tasks.
- **Implementation Guides** address the tasks required to set up an offering, or selected features of an offering. They are intended for implementers. They are organized to follow the task list sequence

of the offerings, as displayed in the Setup and Maintenance work area provided by Oracle Fusion Functional Setup Manager.

- **Concept Guides** explain the key concepts and decisions for a specific area of functionality. They are intended for decision makers, such as chief financial officers, financial analysts, and implementation consultants. They are organized by the logical flow of features and functions.
- **Security Reference Manuals** describe the predefined data that is included in the security reference implementation for one offering. They are intended for implementers, security administrators, and auditors. They are organized by role.

To supplement these guides, which cover specific business processes and offerings, the following guides address common areas:

Guide	Intended Audience	Purpose
Common User Guide	All users	Explains tasks performed by most users.
Common Implementation Guide	Implementers	Explains tasks in the Define Common Applications Configuration task list, which is included in all offerings.
Information Technology Management, Implement Applications Guide	Implementers	Explains how to use Oracle Fusion Functional Setup Manager to plan, manage, and track your implementation projects, migrate setup data, and validate implementations.
Technical Guides	System administrators, application developers, and technical members of implementation teams	Explain how to install, patch, administer, and customize Oracle Fusion Applications.

Other Information Sources

My Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Use the My Oracle Support Knowledge Browser to find documents for a product area. You can search for release-specific information, such as patches, alerts, white papers, and troubleshooting tips. Other services

include health checks, guided lifecycle advice, and direct contact with industry experts through the My Oracle Support Community.

Oracle Enterprise Repository for Oracle Fusion Applications

Oracle Enterprise Repository for Oracle Fusion Applications provides visibility into service-oriented architecture assets to help you manage the lifecycle of your software from planning through implementation, testing, production, and changes. In Oracle Fusion Applications, you can use the Oracle Enterprise Repository for Oracle Fusion Applications for:

- Technical information about integrating with other applications, including services, operations, composites, events, and integration tables. The classification scheme shows the scenarios in which you use the assets, and includes diagrams, schematics, and links to other technical documentation.
- Publishing other technical information such as reusable components, policies, architecture diagrams, and topology diagrams.

The Oracle Fusion Applications information is provided as a solution pack that you can upload to your own deployment of Oracle Enterprise Repository for Oracle Fusion Applications. You can document and govern integration interface assets provided by Oracle with other assets in your environment in a common repository.

Oracle Fusion Applications

For an introduction to Oracle Fusion Applications, see the Oracle Fusion Applications Concepts Guide.

Oracle JDeveloper and Application Development Framework

For information on Oracle JDeveloper and Application Development Framework, see the Oracle Fusion Middleware Oracle Fusion Developer's Guide for Oracle Application Development Framework.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/us/corporate/accessibility/index.html.

Comments and Suggestions

Your comments are important to us. We encourage you to send us feedback about Oracle Fusion Applications Help and guides. Please send your suggestions to oracle_fusion_applications_help_ww@oracle.com. You can use the **Send Feedback to Oracle** link in the footer of Oracle Fusion Applications Help.

Audience

This document is intended for business analysts, implementers, and application developers who will customize and extend the standard functionality provided by Oracle Fusion Applications, and who use Oracle Fusion Functional Setup Manager to configure the extended features.

1

Introduction

About Oracle Fusion Functional Setup Manager

Oracle Fusion Functional Setup Manager (FSM) enables rapid and efficient planning, implementation and deployment of Oracle Fusion Applications through self-service administration. Using FSM, you can:

- Learn and analyze implementation requirements of Oracle Fusion Applications
- Configure Oracle Fusion Applications to match your business needs
- Get complete visibility to setup requirements through guided, sequential task lists
- Enter setup data through user interfaces available directly from the task lists
- Export and import to rapid-start functional setup at different instances
- Validate setup by reviewing setup data reports

In addition, FSM allows application developers to manage application design objects, which are the core components of Oracle Fusion Applications, to add and modify Oracle Fusion Applications functionality.

Extending Oracle Fusion Applications Using Oracle Fusion Functional Setup Manager

Although Oracle Fusion Applications deliver a complete suite of enterprise applications that addresses all major business processes, customers and partners have the ability to extend the applications, if necessary, by adding new functionality to better satisfy their business needs.

Adding or modifying Oracle Fusion Applications functions require managing application design objects, which are the functional building blocks of Oracle Fusion Applications, through FSM. These objects are:

- **Business processes:** A collection of related, structured activities or tasks performed to achieve a particular business goal such as fulfilling orders, procuring raw material, closing accounting

period, and so on. To business users, enterprise applications are means to achieve the end goal of optimizing their business processes. Oracle Fusion Applications use business processes as a platform framework and deliver enterprise application functions in context of business processes.

- **Offerings**: Groups of application functions representing one or more typical business processes and subprocesses that are usually implemented as a unit. Offerings are the core drivers of provisioning and implementing Oracle Fusion Applications. They often include optional business processes and alternative business rules known as options and features respectively. A few examples of offerings are Financials, Procurement, Order Orchestration, Sales, Marketing, Workforce Deployment, and so on.
- **Tasks:** The logical representations of work performed to transact various business processes and subprocesses.
- **Setup tasks:** Represent the work necessary to set up initial configurations of offerings – and in turn, the business processes and sub processes that those offerings support – to make them ready for transactions. For example, Manage Reporting Currency, Assign Balancing Segments to Ledger, and Manage Tax Regime, are some of the setup tasks that are performed to make the Financials offering ready for transactions.

  Setup tasks are performed in FSM. They are associated with interface programs such as user interfaces (UI) or web services that are invoked by FSM when you perform those tasks.
- **Setup task lists:** Groupings of setup tasks that are often performed together to set up a specific business process or subprocess in an offering is called a setup task list.
- **Business objects:** Logical representation of real-world objects. They represent the data entered by performing the Oracle Fusion application tasks. Business objects used by setup tasks are also required to have web services to facilitate export and import of corresponding data. Some of the examples of business objects are Ledgers, Items, Orders, Opportunities, Campaign, and Employees.

Roles with Permission to Extend Oracle Fusion Applications

The following roles that are provided by Oracle Fusion Applications have appropriate permissions to create and modify application design objects that are needed to extend Oracle Fusion Applications functionality. People who are assigned to the roles inherit the privileges.

- Application Implementation Consultant

- Application Implementation Administrator
- Functional Setup Application Developer Duty
- Functional Setup Application Administrator Duty

Note

For detailed information on functional setup roles and permissions, see Oracle Fusion Functional Setup Manager: Roles and Permissions and Oracle Fusion Applications Security Reference Manuals.

2

Methodology for Extending Oracle Fusion Applications

Process Overview

Designing New Functionality

The typical design process follows a top-down approach and starts with identifying what new functions you need to add to the existing applications. The steps are:

- Identify gaps in existing offerings.
- Define all business requirements that should be addressed by the new functions. These are your basis for determining relevant business processes and deciding if you need to:
  - Create new business processes or subprocesses
  - Create new or modify existing offerings
- Evaluate whether any of the business requirements are optionally performed or support alternative business rules. These are your basis for defining options and features.
- Identify what initial setups are required for those business requirements before you can start to transact. These are the basis for defining setup tasks.
- Analyze what data should be captured for those setups. These will be your basis for defining the business objects for the setup tasks.
- Determine how that data will be captured. These will be your basis for defining interface programs (UI or web services) for the setup tasks.

- Identify in what sequence and in what logical grouping these setup tasks should be performed. These are the basis for defining setup task lists.

Once you complete defining the new functions you need, you are ready to define the application design objects required to support the new functions.

Tip

Define a new offering if the new functionality represents one or more typical business processes and subprocesses. For example, you can create Client Relationship Management for Law Firms as a new offering.

Define a new task if the new functionality represents a typical step in a business process and subprocess. For example, you can create Configure Knowledge Repository as a new task representing a step in a business process such as Manage Service Requests.

Define a new task list if the new functionality is represented by a group of tasks and the group of tasks will be reused in many different implementations. For example, Define Security Clearance can be a new task list comprised of multiple tasks if security clearance will be defined in many different implementations.

Reusing Existing Design Objects

The existing application design objects, including those delivered by Oracle, can also be reused if they meet your requirements. For example, you can:

- Create a new offering that includes new and existing options and features
- Create a new option or feature that uses new and existing setup tasks and task lists
- Create a new setup task list that groups new and existing tasks and task lists
- Create a new task that uses new or existing business objects

Tip

Reuse existing application design objects whenever possible instead of creating new objects. This will simplify maintenance of the objects.

See also:

Business Processes – Design Considerations

Offerings, Options, and Features – Design Considerations

Task Lists and Tasks Design Considerations

Business Objects – Design Considerations

Creating Application Design Objects

After design is complete, start creating necessary application design objects and relationships.

Common Attributes

A set of common attributes are always applicable to all design objects. Define these attributes based on the following standards.

- **Name:** The display name of each instance of the object that you will use to identify it. It is a required attribute for all objects and must be unique in the same object. Name can include spaces.
- **Code:** A short code to uniquely identify an instance of an object. A-Z, 0-9, and underscore (_) are the valid characters that can be used in defining a code. When a new object is created, the value of the code is defaulted to the name of the object with spaces replaced by underscores and converting lowercase characters to uppercase. You can change this default value, however, once an object is created (saved in FSM), you cannot modify the code.
- **Description:** A brief description of the object. If no description is entered when creating an object, the name of the object is defaulted as the description. You can change the description of an object even after the object is created and saved in FSM.

Note

In addition to these common attributes, each object has other object-specific attributes. See respective chapters on application design objects for more information on object-specific attributes.

Best Practices

- Reuse objects whenever possible. That is, before creating a new object, find out if a similar object already exists which meets your requirements. Follow the instructions for viewing list in chapters on respective objects to find existing instances of that object.
- Keep names of the objects to less than 40 characters.

- Use meaningful code for better identification when needed.
- Although not required, always provide a meaningful but brief description of the object.

Recommended Naming Standards

The following naming standards are recommended, although not enforced by FSM.

- **Setup tasks:** Start with an action verb followed by the business entity for which the set up is performed. However, exclude *Define, Manage*, and *Maintain* as the action verb. In a special case when the setup task is performed to search and manage a list of business entities, use *Manage* as the starting action verb. Setup task examples are:
  - Create Marketing Budget
  - Transfer Intercompany Transactions
  - Manage Sales Note Type
    The verb *Manage* is used because a list of sales note types is searched and new note types are added.
- **Setup task lists:** Start with the verb *Define* followed by the functionality it represents. Optionally, append the term *Configuration* to clarify that the task list is used to configure a transactional flow rather than perform the actual transactional task. Examples of setup task lists are:
  - Define Sales Stages
  - Define Budgetary Controls
  - Define Expenses Configuration: The term *Configuration* is added to clarify that the actual expenses are not defined with this task list, but rather the configuration for expenses is defined.
- **Options and features:** Use noun phrases for names. Do not begin the phrase with a verb. Option and feature names should be concise and appropriately communicate the functionality they represent. Examples of options and features are:
  - Sales Prediction Engine
  - Intercompany Transaction
  - Sub Ledger Accounting Rules

Tip

Use a bottom-up approach when creating new objects and defining relationships. Although FSM allows you to create objects in any order, creating the dependent associated objects before creating the parent objects (the objects that will consume the other objects) will save you time because you will not need to edit the parent objects multiple times.

See also:

Business Processes

Offerings, Options, and Features

Task Lists and Tasks

Business Objects

Developing Interface Programs for New Setup Tasks

If you create new setup tasks, then you must also build the programs that you will use to enter setup data for those new tasks and to export and import the setup data from one instance into another. FSM supports entering data through UI or web services. To export and import setup data, web services must be developed.

Special Case: Setup Tasks for Deployment Methods

New tasks for setting up deployment methods do not require building new interface programs. Typically Oracle Fusion Application core objects (such as profile options, lookups, and flexfields) fall under this category. Standard UI and web services are used to enter data and export and import their setup data. See Business Objects as Deployment Methods and Task Execution Programs for Setting up Deployment Methods for more information.

Note

If no interface programs are provided for entering or exporting and importing setup data for any new tasks, except if the tasks are for setting up deployment methods, then it is implied that data entry and export and import of those tasks is performed outside of FSM.

See also:

Task Lists and Tasks

Task Execution Programs – FSM Requirements

Business Objects

Web Service for Setup Export and Import

Warning

JDeveloper and ADF are typically used to develop new programs for new tasks. You should be knowledgeable in ADF and JDeveloper, and experienced in developing applications on the Oracle platform. If you are not familiar with them, consult the appropriate documentation and training material.

Defining Deployment Topology for New Programs

Every new program (UI or web service) used in performing setup tasks or setup export and import must have the applications server deployment topology specified.

Deployment Topology: The deployment topology indicates where the J2EE components of the setup programs are deployed on the application servers, such as WebLogic Server (WLS), and consists of:

- Enterprise (J2EE) applications
- Set of modules in each enterprise application where the J2EE components are deployed and executed
- Application sever domains to which the enterprise applications belong

You specify the deployment topology by associating the programs with the appropriate enterprise application modules. After the programs are actually deployed on the application servers for execution, enterprise topology definitions need to be updated with the endpoint URLs. When you perform a setup task or initiate export and import of setup data, FSM invokes the correct program by using the deployment topology information.

Reuse or create new: For new setup-related programs, evaluate if an existing enterprise application module can be reused or a new module is needed.

- **Reusing:** If you can reuse an existing enterprise applications module for a program, then proceed with associating the new program and the relevant deployment topology information (J2EE enterprise application and module) with the desired setup task or business object.
- **Creating new:** If you need a new enterprise application module for a program, then you must first define and register the new

module, enterprise application, and domain information. Once they are registered, you can associate the program and new enterprise application module with the desired setup task or business object.

Note

New tasks are operational for data entry and setup export and import only after the J2EE components are properly deployed to the appropriate enterprise application modules, and Topology Manager is updated with the correct location (endpoint URL).

See also:

Task Lists and Tasks: Task Execution Programs

Task Lists and Tasks: Entering Task Execution Programs

Business Objects: Role of Business Objects in Setup Export-Import

Business Objects: Defining Export and Import Services

Setup Program Deployment Topology

Warning

You are expected to be familiar with J2EE platform, WebLogic Server (WLS), and Oracle Fusion Applications standards and guidelines for defining deployment topology of enterprise (J2EE) applications. If you are not familiar with them, then consult the appropriate documentation and training material before defining deployment topology configurations.

Packaging and Delivering New Objects

If you plan to use Oracle Fusion Application design objects in the same instance where they are created, then they do not require packaging and delivery. Once created, they will automatically become available to use. If, however, the new application design objects are developed in an instance other than where they will be used, then you must package and deliver them using one of the following methods:

- **Offering export and import:** New offerings can be exported from the source instance (instance where they are developed) and then imported at the target instance (instance where they will be used).
- **Task list and task export and import:** Use export and import of a configuration package to deliver new task lists and tasks, which do not belong to an offering, to a different instance.

See also:

Offering Export and Importing Offering

Warning

Typically, application developers do not have the privilege to perform export and import of configuration packages. If you need to perform export and import, then either request the Oracle Fusion Applications security administrators to grant you the appropriate privileges, or work with someone who has the required privileges.

3

Business Processes

Design Considerations

About Oracle Fusion Business Process Model

The term Business Process Management (BPM) refers to a set of activities that organizations can perform to either optimize their business processes or adapt them to new organizational needs. Important elements include business process modeling and analysis (BPA), orchestration (BPEL), and business activity monitoring (BAM).

For business users, enterprise applications are not products or technologies, but rather a means to achieve the end goal of optimizing business process operations. Oracle Fusion Applications use business processes as a platform framework to deliver application functionality. Each Oracle Fusion offering is mapped to one or more business processes.

The business process model used by Oracle Fusion Applications is composed of four levels.

- **Business processes:** The highest level, conceptual representation of major functional groups that describe an enterprise process. Examples of business processes in Oracle Fusion Applications are Asset Lifecycle Management, Compensation Management, Financial Control and Reporting, Order Fulfillment, Procurement, Sales, and Workforce Deployment.
- **Detailed business processes:** The next level decomposition of the business processes to add more visibility to details of a business process. Examples of detailed business processes for the business process Order Fulfillment are Analyze Order Fulfillment, Bill Customers, and Manage Accounts Receivable Balances.
- **Activities:** Under the conceptual levels of business processes and detailed business processes, activities represent further decomposition to more specific business activities. Example of the activities for the detailed business process Bill Customers are Create and Process Bill, Create Receivables Transaction, and Process Billing Adjustments.
- **Tasks:** The final level of the Oracle Fusion Business Process Model contains a set of tasks required to perform a given activity. These tasks drive Oracle Fusion application flows and

functionality. Examples of tasks for the Create and Process Bill activity are Calculate Transaction Taxes, Correct Receivables Transaction Line Exceptions, and Create Receivables Invoice.

Offerings and Business Processes

To make a business process operational you must perform a set of setups, usually grouped as an offering, to implement the corresponding business process. An offering may support an entire business process or one or more detailed business processes. Therefore, when creating a new offering, you must associate the business processes or detailed business processes that will be supported by the offering.

See also:

Offerings, Options, and Features

Sharable and Unsharable Activities

A sharable activity is a list of tasks that can be part of more than one detailed business process. An unsharable activity typically represents a subgroup of an activity and therefore, can belong to only one parent activity.

Tip

By definition, activities are sharable. If you have an exception and need to divide an activity, then create the subactivities as unsharable activities that are only relevant in the context of the specified parent.

Viewing Lists of Business Processes

1. Go to the Setup and Maintenance work area.
2. Click Manage Business Processes from the Tasks pane to open a page with the same name. This page shows a list of business processes, detailed business processes, or activities based on the default search setting. If no personalization was performed, the default search setting for this page is to display all business processes.
3. If the page does not show a list of business processes, detailed business processes, or activities when it opens, then perform one of the following actions to return the desired list:
    - **Ad hoc search:** In the Search region, select Business processes, Detailed business processes, or Activities in the

Search field, enter the appropriate search criteria, and click Search.

- **Saved search:** Select the desired saved search from the Saved Search field.

4. From this page you can:
    - View and edit business process details
    - Create business processes
    - Delete business processes
    - View and edit detailed business processes
    - Create detailed business processes
    - Delete detailed business processes
    - View and edit activities
    - Create activities
    - Delete activities
    - View business processes hierarchy

See also:

Business Process – Design Considerations

Viewing and Editing Business Processes Details

1. From the list of business processes on the Manage Business Processes page, perform one of the following actions to open the Edit Business Process page for the selected business process:
    - Click the business process name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Action menu on the table task bar.
2. View or edit the business process.
3. Click Save or Save and Close to save changes.

Note

You can view Oracle-delivered business processes, but cannot edit or delete them.

See also:

Business Processes – Design Considerations

Viewing Lists of Business Processes

Creating Business Processes

1. From the list of business processes on the Manage Business Processes page, perform one of the following actions to open the Create Business Process page:
    - Click Create > Business Process on the table task bar.
    - Click Create > Business Process from the Action menu on the table task bar.
4. Enter business process details.
5. Click Save or Save and Close to save changes.

Business Process Details

Enter or modify following fields to create a new business process or editing an existing business process.

Basic Information

Field	Field Type	Required	Description
Name	Text	Yes	Use a unique name for the business process that you can easily identify.
Code	Text	Yes	Use a unique code to identify the business process. Once created, code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the business process.

Note

Review best practices and standards and guidelines for defining the core fields.

Detailed Business Processes

This region is applicable only in View and Edit mode, not in Create mode. It shows the detailed business processes that are associated with the given business process. If collapsed, expand the region.

Column	Description
Display Sequence	The value in this column determines the relative sequence in which the associated detailed business processes will appear in the application, and therefore should correspond to the sequence in which the detailed business processes are expected to be performed.
Name	Name of the detailed business process.
Description	Description of the detailed business process.

Changing Detailed Business Process Display Sequence

When a business process is specified as the parent of a detailed business process, the newly associated detailed business process gets a higher display sequence number than the existing associations. You will see the newly associated detailed business process at bottom of the list. You can change the display sequence number of the associated detailed business processes as follows.

1. If the Detailed Business Process region is collapsed, expand it.
2. Change the values in the Display Sequence column to the desired values.

Deleting Business Processes

1. Select the desired row from the list of business processes on the Manage Business Processes page.
2. Perform one of the following actions:
    - Click Delete on the table task bar.
    - Click Delete from the Action menu on the table task bar.

Viewing and Editing Detailed Business Processes

1. From the list of detailed business processes on the Manage Business Processes page, perform one of the following actions to open the Edit Detailed Business Process page:
    - Click the detailed business process name.

- Select the desired row and click Edit on the table task bar.
- Select the desired row and click Edit from the Action menu on the table task bar.

2. View or edit detailed business process details.
3. Click Save or Save and Close to save changes.

Note

You can view Oracle-delivered detailed business processes, but cannot edit or delete them.

Creating Detailed Business Processes

1. From the list of detailed business processes on the Manage Business Processes page, perform one of the following actions to open the Create Detailed Business Process page.
    - Click Create > Detailed Business Process on the table task bar.
    - Click Create > Detailed Business Process from the Action menu on the table task bar.
2. Enter detailed business process details.
3. Click Save or Save and Close to save changes.

Detailed Business Process Details

Basic Information

When creating new detailed business processes or editing an existing detailed business process, enter or modify following fields. If the Basic Information region is collapsed, expand it.

Field	Field Type	Required	Description
Name	Text	Yes	Use a unique name for the detailed business process that you can easily identify.
Code	Text	Yes	Use a unique code to identify the detailed business process. Once created, code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the detailed business process.
Business Process	List of values	Yes	Search and select the parent business process

Warning

A detailed business process can belong to only one business process. If a new business process is selected, then the existing association is replaced by the new association.

Note

Review best practices and standards and guidelines for defining the core fields.

Associating Activities

When creating new detailed business processes or editing an existing detailed business process, associate activities as follows.

1. Expand the Activities region, if it is collapsed.
2. Add or remove activities to detailed business processes associations by performing the following steps.

Adding Activity Associations

1. To associate one or more activities, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate activities from the popup window.
3. Apply and close the window when complete.

Removing Activity Associations

To remove an existing association:

1. Select appropriate rows in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Changing Activity Display Sequence

- By default, the newly added activities are added to the bottom of the table with sequential display sequence numbers that are higher than the existing highest number.
- To change the display sequence numbers, click the Display Sequence column in the desired row and edit the existing value.

Deleting Detailed Business Processes

1. Select the desired row from the list of detailed business processes on the Manage Business Processes page.
2. Perform one of the following actions:
    - Click Delete on the table task bar.
    - Click Delete from the Action menu on the table task bar.

Viewing and Editing Activities

1. From the list of activities on the Manage Business Processes page, perform one of the following actions to open the Activity page:
    - Click the activity name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Action menu on the table task bar.
2. View or edit activity details.
3. Click Save or Save and Close to save changes.

Note

You can view Oracle-delivered activities, but cannot edit or delete them.

Creating Activities

1. From the list of activities on the Manage Business Processes page, perform one of the following actions to open the Create Activity page.
    - Click Create > Activity on the table task bar.

- Click Create > Activity from the Action menu on the table task bar.

2. Enter activity details.
3. Click Save or Save and Close to save changes.

Activity Details

Basic Information

When creating new activities or editing an existing activity, enter or modify following fields. If the Basic Information region is collapsed, expand it.

Field	Field Type	Required	Description
Name	Text	Yes	Use a unique name for the activity which you can easily identify.
Code	Text	Yes	Use a unique code to identify the activity. Once created, code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the activity.
Unsharable	Checkbox	No	Select this checkbox if this activity is a sub activity of another activity and can be used only in the context of the parent activity. The **DEFAULT** setting of this field is **UNCHECKED**.
Parent Activity	List of Values	Yes (conditional)	Search and select the name of the parent activity.

Note

Parent activity is applicable only if the activity is designated as Unsharable. Therefore, the Parent Activity field becomes visible and required if the Unsharable checkbox is selected.

Also, review best practices and standards and guidelines for defining the core fields.

Associating Detailed Business Processes

When creating new activities or editing an existing activity, associate detailed business processes as follows:

1. Expand the Parent Detailed Business Processes region, if it is collapsed.
2. Add or remove detailed business processes associations by performing the following steps.

Adding Detailed Business Process Associations

1. To associate one or more detailed business processes, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate detailed business processes from the popup window.
3. Apply and close the window when complete.

Removing Detailed Business Process Associations

To remove an existing association:

1. Select appropriate rows in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Note

This region is visible if the Unsharable checkbox is not selected. Every unsharable activity, on the other hand, must be associated with one or more detailed business processes.

Associating Tasks

When creating new activities or editing an existing activity, associate tasks as follows.

1. Expand the Tasks region, if it is collapsed.
2. Add or remove task associations by performing the following steps.

Adding Task Associations

1. To associate one or more tasks, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate tasks from the popup window.
3. Apply and close the window when complete.

Removing Task Associations

To remove an existing association:

1. Select appropriate rows in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Deleting Activities

1. Select the desired row from the list of activities on the Manage Business Processes page.
2. Perform one of the following actions:
    - Click Delete on the table task bar.
    - Click Delete from the Action menu on the table task bar.

Viewing Business Process Hierarchy

You can also view the business processes in a hierarchical view.

1. From the list of business processes, detailed business processes, or activities on the Manage Business Processes page, select the row that contains child rows that you want to view.
2. Perform one of the following actions to open the View Hierarchy page.
    - Click View Hierarchy on the table task bar.
    - Click View Hierarchy from the Action menu on the table task bar.

3. The selected row that contains the children you want to view will be the top node and the name will appear in the page title.
4. All successive children levels will be shown in a hierarchical view. Expand and collapse as appropriate to see various levels in the hierarchy.
5. Click Done to close the hierarchical view and return to the Manage Business Processes page.

4

Offerings, Options, and Features

Design Considerations

About Offerings, Options, and Features

They are various modules of Oracle Fusions Applications representing one or more business processes and subprocesses.

Offerings: The highest level groupings of applications functions representing one or more typical business processes and subprocesses that are usually implemented as a unit. Offerings are the core drivers for provisioning (installing) and implementing Oracle Fusion Applications. Examples are Sales and Financials.

Options: Optional subprocesses in an offering. As the name indicates, options are optionally implemented and are not core to standard transaction of the offering. For example, Sale Catalog is an option of Sales offering, and Intercompany Transaction is an option of Financials offering.

Features

Features are alternative business methods or rules used to fine tune business processes and subprocesses supported by an offering or an option. For example:

- Customer Merge is a feature of the Sales offering since some businesses may choose to merge their customers from all channels while others may not.
- Similarly, Accounting Representation is a feature of the Oracle Fusion Accounting Hub offering with choices of Currency, Accounting Calendar, Chart of Accounts, and Accounting Method, where different combinations of these choices may apply to different businesses.

Feature Choices: While offerings and options allow only Yes or No selection, features can be defined with multiple choices. In addition, features with multiple choices can be defined to allow you to choose only one of the choices or to allow multiple selections.

Feature Choice Selection Types: One of the following three selection types is specified for each feature, which then determines how you are allowed to make selection of the choices of the feature.

- **Yes or No:** When a feature is defined with this selection type, it gets two default choices, Yes and No, and no other choices can be defined for it. This type of feature is presented with a single checkbox. When you select the checkbox, the Yes option for the feature is set. Otherwise, the No option is set.

  For example, when implementing Oracle Fusion Financials, if customers are allowed to choose whether they will use subledger, then a feature called Secondary Ledger can be defined as a Yes or No selection type. You will be presented with a checkbox, which you can either select to indicate that you will use Secondary Ledger in Oracle Fusion Applications, or do not select the checkbox to indicate otherwise.

- **Single:** A feature with this selection type will have multiple choices but you are allowed to select only one of those choices, and therefore, the choices are presented as radio buttons.

  For example, if Oracle Fusion Financials supports two-way, three-way and four-way matching methods in accounts payable, but only one of those choices is applicable for an organization, then purchase order matching methods can be defined as a feature with a selection type of Single with the choices of two-way, three-way and four-way. You will see the three choices presented as radio buttons allowing selection of only one of the three choices.

- **Multiple:** A feature with this selection type has multiple choices, and you are allowed to select any and all of those choices. The choices are presented as check boxes.

  For example, if Oracle Fusion Applications support accounting representation by currency, accounting calendar, chart of accounts, and accounting methods, and an organization can use any and all of the four representation types, then accounting representation can be defined as a feature with a selection type of Multiple. Each of the four choices will have a checkbox, and any and all checkboxes can be selected.

Role of Top Task List in Offerings

The top task list of an offering should contain the complete list of setup tasks needed to implement the offering, including its dependent options and features. This task list is used to auto-generate the implementation task list when the offering is selected for implementation. Every offering must be associated with only one top task list should be organized in the sequence and groupings that is optimal for implementing the offering.

Note

Get a copy of the Setup Task Lists and Tasks report from the FSM Getting Started page to find more examples of top task lists. For each offering, the report shows all task lists and tasks contained in the top task list of the offering.

Dependent Options and Features

Offerings, options, and features are related to each other in hierarchical manner, such as:

- Offerings can have dependent options and features
- Options can have dependent options and features
- Features can have dependent features

Assists in Progressive Decision Making: The dependent options and features are presented to application implementers in the context of the parent offering, option, and feature. This helps application implementers progressively decide whether the corresponding application functions are applicable to their business and make selections about implementation accordingly.

For example, if you are implementing the Sales offering, you are presented with the options and features belonging to Sales, such as whether to use the Sales Catalog option or the Merge Sales Customers feature. On the other hand, if you are implementing the Financials offering instead, then Sales Catalog and Merge Sales Customers are irrelevant.

Note

Get a copy of the Associated Features report available from the FSM Getting Started page to find more examples of hierarchical definitions of Oracle-delivered offerings, options, and features.

Shared Options

Options can be shared, meaning they can have more than one parent. For shared options, selections are maintained at the global context and not in the context of a parent. This means if the option is selected as part of selection of any of its parents, the option will be considered selected for all its parents. For example, Absence Management is an option for both Compensation Management and Workforce Deployment offerings. When configuring the offerings, if you select Absence Management for Compensation Management, then it is also selected for Workforce Deployment offering, and vice versa.

Tip

If you need to make different selections of the same option in the context of each parent, then define different options instead of sharing the same option. For example, in the case of Absence Management, define two options, Absence Management for Compensation and Absence Management for Workforce Deployment, and associate each of them to the appropriate parent.

Setup Tasks for Options and Features

Tasks and task lists in the top task list of an offering that are used to set up the dependent options and features of the offering should be identified and associated with the appropriate options and features accordingly.

If setup tasks and task lists are associated with options and features, they become optional when the corresponding offering is implemented. That is, when an implementer chooses to implement the parent offering, the implementation task list, which is auto-generated by FSM based on the top task list, will include the optional setup tasks and task lists only if their associated options and features are also selected by the implementer. In the case of features with multiple choices, the selection of the choices will determine which setup tasks are included or excluded from the implementation task list.

Example

Consider the following example:

- The Oracle Fusion Accounting Hub offering has a dependent option called Intercompany Transaction.
- The same offering also has a dependent feature called Accounting Representation, which has multiple choices of Currency, Accounting Calendar, Chart of Accounts, and Accounting Methods.
- The top task list of the offering, also called Oracle Fusion Accounting Hub, contains the following list of tasks along with many others:
    - Define And Maintain Intercompany Processing Rules
    - Manage Reporting Currencies
    - Define Secondary Ledgers
- The option and feature choices are associated as follows:
    - The Intercompany Transaction option is associated with Define and Maintain Intercompany Processing Rules task list.

- The Currency choice of the Accounting Representation feature is associated with the Manage Reporting Currency task list.
- The Accounting Calendar, Chart of Accounts, and Accounting Methods choices of the Accounting Representation feature are associated with the Define Secondary Ledgers task list.

In this case, when the Oracle Fusion Accounting Hub offering is implemented, the auto-generated implementation task list will include the setup tasks only if the implementer has selected the corresponding option or feature choice. These optional tasks will otherwise be excluded, such as:

- The Define and Maintain Intercompany Processing Rules task list will be included in the implementation task list if you select the Intercompany Transaction option.
- Similarly, the Manage Reporting Currency task list will be included if the Currency choice of the Accounting Representation feature is selected by you, while the Define Secondary Ledgers task list will be included if any of the other choices are selected.

Note

Not all options and feature choices require association with setup tasks. If no setup tasks are associated, then the selection of that option or feature choice will not have any impact on the auto-generated implementation task list.

Options and Feature Choices Without Associated Setup Tasks

The selection of option and feature choices can also be used to manipulate UI and menu components such as expose or hide buttons or menu items, and so on. Therefore, an option or feature choice could have no impact on setup tasks but used only to control a UI or menu component. In such cases, the option and feature choice should still be defined but will not have any associated setup tasks.

FSM provides public web services which can be used to find the selection status of option and feature choices and then use them to manipulate other interface components. See Offering, Option, and Feature Choice Integration for more information.

Note

An option or feature choice can be used to control both setup tasks and UI menu components. The two outcomes are not mutually exclusive. That is, if an option and feature choice is associated with setup tasks, then the

tasks will become optional for the corresponding offering. On the other hand, any option and feature choices can be used in controlling interface components irrespective of whether the same option and feature choice is also used in defining one or more setup tasks as option for an offering.

Associating Related Documents with Offerings

When documents are associated with offerings, they become available to you through Getting Started page. These documents are intended to help you in planning successful implementation of the offerings.

For every new offering a default set of reports are automatically provide, whose content or format cannot be modified. They are:

- **Offering Content Guide**: Shows detailed information on the business processes and subprocesses supported by implementation of the Offering.
- **Setup Task Lists and Tasks**: Shows the task lists and tasks that should be performed to successfully implement the offering.
- **Related Features**: Shows the list of options and features associated with the offering.
- **Associated Business Objects**: Gives visibility to all setup data needed to implement the Offering by providing a list of all business objects that are associated with the setup tasks belonging to the offering.
- **Related Enterprise Applications**: Shows the list of enterprise applications used by the programs (UI and web services) for the offering.

In addition to these reports, you may choose to add other custom reports and related documents that help you during planning and implementation or when performing the setup tasks.

Viewing Lists of Offerings or Options

1. Go to the Setup and Maintenance work area.
2. Click Manage Offerings and Options from the task pane, which will display a page with the same name. This page will show a list of offerings or options based on your default search setting. If no personalization was performed, the default search setting of this page is to display all offerings.
3. If the page does not show a list of offerings or options when first displayed, then perform one of the following actions to return an appropriate list:

- **Ad hoc search:** In the Search region, select Offering or Option in the Search field, enter the appropriate search criteria, and click Search.
- **Saved search:** Select the desired saved search from the Saved Search field.

4. From this page you can:
    - View and edit offering
    - Create offering
    - Delete offering
    - Export offering
    - Import offering
    - View and edit option
    - Create option
    - Delete option

See also:

Offering, Options, and Features – Design Considerations

Viewing and Editing Offering Details

Note

You should review design considerations to understand various design implications before you start creating or editing offering, option, and feature details, especially if you are doing so for the first time.

1. From the list of offerings on the Manage Offerings and Options page, perform one of the following actions to open the Edit Offering page:
    - Click the offering name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Actions menu on the table task bar.
2. View or edit offering details.
3. Click Save or Save and Close to save changes.

Note

You can view Oracle-delivered offerings, but cannot edit or delete them.

See also:

Offering, Options, and Features – Design Considerations

Viewing Lists of Offerings and Options

Creating Offerings

1. From the list of offerings on the Manage Offerings and Options page, perform one of the following actions to open the Create Offering page:
    - Click Create Offering on the table task bar.
    - Click Create Offering from the Actions menu on the table task bar.
2. Enter offering details.
3. Click Save or Save and Close to save changes.

Offering Details

For a typical offering, you will define basic information, related business processes, associated options, associated features, and associated documents.

Missing Associations

Although associations are not required to define an offering, each affects how you experience the offering usage.

- If business processes are not associated, then the Offering Content Guide available through Getting Started page will be empty.
- If options and features are not associated, then the offering will not allow you to choose optional feature functionality.
- If documents are not associated, then the list of related documents for the offering available from the Getting Started page will be empty. These documents help you plan offering implementation.

Entering Basic Information

When creating a new offering or editing an existing offering, enter or modify the following fields. If the Basic Information region is collapsed, expand it first.

Field	Field Type	Required	Description
Name	Text	Yes	Use a unique name for the offering which you can easily identify.
Code	Text	Yes	Use a unique code to identify the offering. Once created, code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the offering.
Offering Icon	Text	Yes	Browse or enter fully qualified location of an image file. Supported formats are: GIF, PNG, JPEG, and TIFF. This image will be used as the icon representing the offering on Getting Started page. For best result, see Tip.
Top Task List	List of Values	Yes	Select the task list which represents the complete list of tasks needed to implement the entire functionality of the offering, including its dependent options and features.

Tip

Use an offering icon that is 64x64 in size. If the icon is larger than the recommended size, it will be truncated when displayed on the Getting Started page. On the other hand, if the icon is smaller than the recommended size then, it will be expanded to the recommended size using white as the background color.

Note

Also review best practices and standards and guidelines for defining these core fields.

Associating Business Processes and Detailed Business Processes

When creating a new offering or editing an existing offering, associate business processes and detailed business processes as follows.

1. Expand the Business Processes region on the Enter Basic Information page of an offering, if it is collapsed.
2. Add or remove offering to business processes and detailed business processes associations by performing the following steps.

Adding Business Process and Detailed Business Process Associations

1. To associate one or more business processes and detailed business processes, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate business processes and detailed business processes from the popup window.
3. Apply and close the window when complete.

Removing Business Process and Detailed Business Process Associations

To remove an existing association:

1. Select the appropriate row in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Associating Dependent Options

When creating a new offering or editing an existing offering, associate dependent options as follows.

1. Click Associate Options, the second step of the guided process for creating and editing an offering. This will display a page with the same name.
2. Add or remove offering to option associations by performing the following steps.

Adding Option Associations

1. To associate one or more options, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate options from the popup window.

3. Apply and close the window when complete.

Removing Option Associations

To remove an existing association:

1. Select appropriate rows in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Changing Option Display Sequence

- By default, the newly added options are added to the bottom of the table with sequential display sequence numbers that are higher than the existing highest number.
- To change the display sequence numbers, click the Display Sequence column in the desired row and edit the existing value.

Associating Dependent Features

When creating a new offering or editing an existing offering, associate dependent features as follows.

1. Click Associate Features, the third step of the guided process for creating and editing an offering. This will display a page with the same name.
2. Add or remove offering to feature associations by performing the following steps.

Adding Feature Associations

1. To associate one or more features, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate features from the popup window.
3. Apply and close the window when complete.

Removing Feature Associations

To remove an existing association:

1. Select appropriate rows in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Changing Feature Display Sequence

- By default, the newly added features are added to the bottom of the table with sequential display sequence numbers that are higher than the existing highest number.
- To change the display sequence numbers, click the Display Sequence column in the desired row and edit the existing value.

Associating Documents

When creating a new offering or editing an existing offering, associate documents as follows.

1. Click Associate Documents, the fourth step of the guided process for creating and editing an offering. This will display a page with the same name.
2. Add or remove offering to document associations by performing the following steps.

Adding Document Associations

1. To associate one or more documents, perform one of the following two actions:
    - Click Add Row on the table task bar.
    - Click Add Row from the Action menu on the table task bar.
2. A new editable row will open up in the table. Enter data for the new row.

Column	Column Type	Required	Description
Type	List of Values	Yes	Specifies what type of document is associated. The documents could be any one of the Oracle-delivered standard reports for Offerings or URL for one of your

			own document.
Title	Text	Yes	Use a brief descriptive title for the document, which is how it will appear in through **Getting Started** page. The title of the standard reports cannot be modified.
URL	Text	No	Enter URL of the document. This field is enabled only if you have selected the document type as URL.
Parameters	Test	No	If the document accepts a parameter to generate dynamic content, then specify the parameters in this field. When passing multiple parameters, use & as delimiter, such as param1='X'&param2='Y').
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the Offering. As with the title, the description of the standard reports cannot be modified.
Display Seq	Number	Yes	Specify the sequencing in which the related documents will appear in the Getting Started page.

Removing Document Associations

To remove an existing association:

1. Select appropriate rows in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Deleting Offerings

1. From the list of offerings on the Manage Offerings and Options page, select the desired row.
2. Perform one of the following actions:
    - Click Delete on the table task bar.
    - Click Delete from the Action menu on the table task bar.

See also:

Viewing Lists of Offerings and Options

Exporting Offerings

You can export an offering to transfer it from one instance to another, or to save a version of it. When exported, the current version of the offering details, which includes the entire top task list, associated business processes and detailed business processes, dependent options and features, and related documents, are exported to a ZIP file. However, related setup data is NOT exported. Download the export ZIP file and use it to import the offering to another instance.

1. Select an offering from the list of offerings on the Manage Offerings and Options page
2. Perform one of the following actions:
    - Click Export on the table task bar.
    - Click Export from the Action menu on the table task bar.
3. When the Offering export completes successfully, a date appears as a link in the Download Published Version column, indicating when the export file was published. You may have to refresh the page to see the latest version in the column.
4. Download or delete the export file.

Downloading Export Files

To download an export file:

1. Review the Download Published Version column of the desired offering and determine if the displayed version is the one you want. Refresh the page if necessary to see the latest published version.
2. Click the date link in the column to download the export file.

Deleting Export Files

To delete a published version of an export file, select the desired offering and click Delete Published Version from the Action menu on the table task bar.

Note

You cannot export Oracle-delivered offerings.

Importing Offerings

As an alternative to creating an offering from the start, you can import an offering created at another instance. After the import completes successfully, the imported offering will behave the same as an offering that was created at the target instance.

1. Go to the Setup and Maintenance work area.
2. Click Manage Offerings and Options from the task pane, which will display a page with the same name.
3. Perform one of the following actions:
    - Click Import on the table task bar.
    - Click Import from the Action menu on the table task bar.
4. In the popup window, browse or enter fully qualified location of the ZIP file that you want to import. This is the file that you downloaded after exporting an offering.
5. Click Import.

Note

The imported offering version will replace an offering with the same code if one exists at the target instance.

Viewing and Editing Option Details

1. From the list of options on the Manage Offerings and Options page, perform one of the following actions to open the Edit Option page:
    - Click the option name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Actions menu on the table task bar.
2. View or edit option details.
3. Click Save or Save and Close to save changes.

Note

You can view Oracle-delivered options, but cannot edit or delete them.

Creating Options

1. From the list of options on the Manage Offerings and Options page, perform one of the following actions to open the Create Option page:
    - Click Create Option on the table task bar.
    - Click Create Option from the Actions menu on the table task bar.
2. Enter option details.
3. Click Save or Save and Close to save changes.

Option Details

At a minimum you will define basic information and associated setup tasks for every option. You may also define dependent options and dependent features.

Entering Basic Information

When creating a new option or editing an existing option, enter or modify following fields. If the Basic Information region is collapsed, expand it.

Field	Field Type	Required	Description
Name	Text	Yes	Use a unique name for the option which you can easily identify.
Code	Text	Yes	Use a unique code to identify the option. Once created, code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the option.

Note

Review best practices, standards, and guidelines for defining these core fields.

Associating Setup Tasks and Task Lists

When creating a new option or editing an existing option, associate setup task and task lists as follows.

1. Click Associate Setup Tasks, the second step of the guided process for creating and editing an option. This will display a page with the same name.

2. Add or remove setup task and task list associations by performing the following steps.

Adding Setup Task and Task List Associations

1. To associate one or more setup tasks and task lists, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate tasks and task lists from the popup window.
3. Apply and close the window when complete.

Removing Setup Task and Task List Associations

To remove an existing association:

1. Select the appropriate row in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Note

Not all options require an association with a setup task. See Option and Feature Choices Without Associated Setup Tasks for more information.

Associating Dependent Options

When creating a new option or editing an existing option, associate dependent options as follows.

1. Click Associate Options, the third step of the guided process for creating and editing an option. This will display a page with the same name.
2. Add or remove option to option associations by performing the following steps.

Adding Option Associations

1. To associate one or more options, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate options from the popup window.
3. Apply and close the window when complete.

Removing Option Associations

To remove an existing association:

1. Select appropriate rows in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Changing Option Display Sequence

- By default, the newly added options are added to the bottom of the table with sequential display sequence numbers that are higher than the existing highest number.
- To change the display sequence numbers, click the Display Sequence column in the desired row and edit the existing value.

Associating Dependent Features

When creating a new option or editing an existing option, associate dependent features as follows.

1. Click Associate Features, the fourth step of the guided process for creating and editing an option. This will display a page with the same name.
2. Add or remove option to feature associations by performing the following steps.

Adding Feature Associations

1. To associate one or more features, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate features from the popup window.
3. Apply and close the window when complete.

Removing Feature Associations

To remove an existing association:

1. Select appropriate rows in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Changing Feature Display Sequence

- By default, the newly added features are added to the bottom of the table with sequential display sequence numbers that are higher than the existing highest number.
- To change the display sequence numbers, click the Display Sequence column in the desired row and edit the existing value.

Deleting Options

1. From the list of options on the Manage Offerings and Options page, select the desired row.
2. Perform one of the following actions:
    - Click Delete on the table task bar.
    - Click Delete from the Action menu on the table task bar.

See also:

Viewing Lists of Offerings and Options

Viewing Lists of Features

1. Go to the Setup and Maintenance work area.
2. Click Manage Features from the task pane, which will display a page with the same name. This page will show a list of features based on your default search setting. If no personalization was performed, the default search setting of this page is to display all features.
3. Modify search results if desired.
    - **Ad hoc search:** In the Search region, enter the appropriate search criteria and click Search.
    - **Saved search:** Select the desired saved search from the Saved Search field.
4. From this page you can:
    - View and edit feature
    - Create feature
    - Delete feature

Viewing and Editing Feature Details

1. From the list of features on the Manage Features page, perform one of the following actions to open the Edit Feature page:
    - Click the feature name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Actions menu on the table task bar.
2. View or edit feature details.
3. Click Save or Save and Close to save changes.

Note

You can view Oracle-delivered features, but cannot edit or delete them.

See also:

Offering, Options, and Features – Design Considerations

Viewing Lists of Features

Creating Features

1. From the list of features on the Manage Features page, perform one of the following actions to open the Create Feature page:
    - Click Create on the table task bar.
    - Click Create from the Actions menu on the table task bar.
2. Enter offering details.
3. Click Save or Save and Close to save changes.

Feature Details

At minimum, primary details (basic information and feature choices) and associated setup tasks should be defined for every feature. See Setup Tasks for Options and Features for an exception case when setup tasks may not be associated with a feature.

You can also define dependent features.

Entering Primary Details

When creating a new feature or editing an existing feature, enter or modify the primary details of the feature as follows. If the Basic Information region is collapsed, expand it.

Field	Field Type	Required	Description
Name	Text	Yes	Use a unique name for the feature that you can easily identify.
Code	Text	Yes	Use a unique code to identify the feature. Once created, code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the feature.
Selection Type	List of Values	Yes	This field indicates what type of selection you will be able to make for this feature.

Note

Review best practices, standards, and guidelines for defining the core fields, and About Offerings, Options, and Features for more details on the Selection Type field.

Define the choices to be presented for this feature in the Feature Choices region.

Column	Column Type	Required	Description
Display Sequence	Text	Yes	Enter sequence numbers for the feature choices in which they will be presented to you. This column in not editable if the selection type of the feature is set to Yes or No, in which case Yes is always listed with display sequence as 1 and No is listed as 2.
Name	Text	Yes	Use a brief, unique name that can be easily recognized to understand the choice. This name appears in the Configure Offering > Select Feature Choices page.
Code	Text	Yes	Use a unique code to identify the feature choice. Once created, the code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description.
Default	Checkbox or Radio Button	No	Specify which of the feature choices will be selected by default. For Yes or No and Single selection types, you can set only one choice as the default. For the Multiple selection type**,** you can set more than one choice as default.

Associating Setup Tasks

When creating a new feature or editing an existing feature, associate setup task and task lists as follows.

1. Click Associate Setup Tasks, the second step of the guided process for creating and editing a feature. This will display a page with the same name, which will show two tables in a master-details format.
    - **Feature Choices:** This is the master table. The defined feature choices are in the order of display sequence.
    - **<Choice Name>: Associated Setup Tasks:** This is the detail table that shows the task and task lists that are associated with selected feature choice in the master table.
2. Select a feature choice in the master table.
    - The detail table shows tasks and task lists that are associated with the feature choice selected in the master table.
3. Add or remove feature choices to setup task and task list associations by performing the following steps.

Adding Setup Task and Task List Associations

1. To associate one or more setup tasks and task lists, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate tasks and task lists from the popup window.
3. Apply and close the window when complete.

Removing Setup Task and Task List Associations

To remove an existing association:

1. Select the appropriate row in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Note

Not all features require an association with a setup task. See Option and Feature Choices Without Associated Setup Tasks for more information.

See also:

Setup Tasks for Options and Features – Design Considerations

Associating Dependent Features

Dependent features of a parent feature are defined through the feature choices of the parent. When creating a new feature or editing an existing feature, associate dependent features as follows.

1. Click Associate Dependent Features, the third step of the guided process for creating and editing a feature. This will display a page with the same name, which will show two tables in a master-details format.
    - **Feature Choices:** This is the master table. The defined feature choices are in the order of display sequence.
    - **<Choice Name>: Dependent Features:** This is the detail table that shows the features that are associated with selected feature choice in the master table.

2. Select a feature choice in the master table.
    - The detail table shows dependent features that are associated with the feature choice selected in the master table.
3. Add or remove feature to feature associations by performing the following steps.

Adding Feature Associations

1. To associate one or more features, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate features from the popup window.
3. Apply and close the window when complete.

Removing Feature Associations

To remove an existing association:

1. Select the appropriate row in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Changing Feature Display Sequence

- By default, the newly added features are added to the bottom of the table with sequential display sequence numbers that are higher than the existing highest number.
- To change the display sequence numbers, click the Display Sequence column in the desired row and edit the existing value.

Deleting Features

1. From the list of features on the Manage Features page, select the desired row.
2. Perform one of the following actions:

- Click Delete on the table task bar.
- Click Delete from the Action menu on the table task bar.

5

Task Lists and Tasks

Design Considerations

About Tasks

A task is the logical representation of work performed for business processes and subprocesses supported by Oracle Fusion Applications. In FSM, setup tasks represent the lowest units for work that needs to be performed when implementing an Oracle Fusion offering. Tasks are often related to entering offering-specific setup data, but may be other actions needed to make the offering ready for a transaction. An example of a task is installing a secured printer to print sales orders when implementing the Sales offering.

About Task Lists

A setup task list is a logical grouping of setup tasks that are related to the same business processes or subprocesses, and are often performed together. Task lists are hierarchical, which means that a parent task list can include children that are either tasks or other task lists.

For example, Define Sales Quotas is a setup task list that supports setting up sales quotas when implementing the Sales offering. This task list includes the following task lists and tasks as children:

- Quota Management Lookup task list
- Manage Sales Quota Spread Formulas task
- Manage Sales Territory Quota Formulas task
- Manage Sales Quota Plan task
- Manage Sales Quota Seasonality Groups task

Note

You can add Oracle-delivered task lists and Tasks to your custom-created task lists; however, you cannot modify Oracle-delivered task lists.

Tip

Consider defining a task list when a group of tasks are expected to be performed together to complete implementation of any specific applications functionality. The task lists will not only give better visibility to full implementation requirements, but also help to easily reuse the same set of tasks in many implementations.

Task Execution Programs

When you create a setup task, you typically associate a program to the task that is executed when you perform the task. This program can be either a UI or a web service.

User Interfaces (UI)

In the most common cases, setup tasks are associated with UIs, which you use to enter setup data. In Oracle Fusion, typically ADF task flows are used as UI programs.

Best Practices

For optimal user experience, setup task UIs should follow certain standards and guidelines. These are:

- If the UIs are invoked from FSM, then the control should return to FSM when you close the page.
- The UIs should be able to accept setup scope and task parameters when applicable.

Special Case: Setting Up Deployment Methods

In FSM, certain business objects can be specified as deployment methods. Tasks that are used to enter setup data for deployment methods do not require developing and associating any programs with them. Instead, in each task using the deployment method, you must specify the appropriate value for the qualifying attribute of the deployment method. A common UI and a common export and import service are used by all tasks using the same deployment method for entering and exporting and importing corresponding setup data. See Business Objects as Deployment Methods for more information.

Application Core Objects

Oracle-delivered deployment methods are typically the applications core objects using Applications Core Setup UI. They are:

- Attachment category
- Attachment entity
- Common lookup
- Descriptive flexfield
- Document sequence
- Document sequence category
- Extensible flexfield
- Flexfield value set
- Key flexfield
- Message
- Profile category
- Profile option
- Profile value
- Set-enabled lookup
- Standard lookup
- Tree
- Tree label
- Tree structure

Warning

Explanation of various types of applications core setups, and standards and guidelines for defining those setup tasks, are beyond the scope of this document. If you are not familiar with them, consult the Oracle® Fusion Applications Developer's Guide and related training material.

Web Services

Setup tasks can also be executed using web services. When you initiate such tasks from FSM, the associated web services are invoked and the data entry is processed in the background, without direct user interaction. The web services can run a program immediately or may submit a background process.

Request ID: If the web service is submitting a background process, it should return the Request Process ID of the background process.

Task Completion Status: The web service must return the task completion status once the Task execution is finished.

Tasks Without Execution Programs (External Tasks)

If no program (UI or web service) is provided for a setup task and it is not used for setting up any deployment method, then it is implied that those are external tasks, meaning they will be performed outside of FSM. Although you cannot perform the tasks directly from FSM, all other functionality of setup tasks, such as assigning users and tracking progress during implementation, are available for those tasks in FSM.

Business Object Associations with Setup Tasks

The business objects associated with a setup task represent the setup data corresponding to the task. A setup task may be associated with multiple business objects. In FSM, these associations are specifically used during export and import to identify the setup data that will be migrated.

See also:

Role of Business Objects in Setup Export and Import

Required Tasks in a Task List

A required task in a task list indicates that the task is required to successfully complete the implementation of the task list. Required tasks can be defined only in the context of a task list. That is, if the same task belongs to more than one task list, it can be required in one task list but not in the others.

When a task is marked as required for a task list, an asterisk (*) appears in front of the task name when viewing the same task list.

Note

The Required flag is only an indicator of the importance of the task in the task list. FSM cannot enforce that the task is actually performed.

Task List Scope

If the tasks in a setup task list need to be performed iteratively to enter context-sensitive setup data, then a scope may be defined for the task list. As a result, you will have the ability to choose a context, a scope value,

before performing the tasks in the task list. When you perform the tasks in the task list, FSM will pass the selected scope value to the tasks.

For example, Define Accounting Configuration is a setup task list with multiple tasks used for such setups as specifying primary ledger options, assigning balancing segments, and managing reporting currencies for a primary ledger. However, setups of primary ledger options, balancing segments, or reporting currencies are different for different primary ledgers. Therefore, primary ledger is used as the scope for the Define Accounting Configuration task list. This allows you to first select a primary ledger and then enter setup data for the tasks in the task list specific to the selected primary ledger.

In technical terms, a scope is a business object, which is part of the unique identifier of setup data related to the tasks in a task list. Each instance of the business object is known as scope value in FSM. For example, if legal entity is the scope, then each of its values, such as InFusion USA, InFusion Canada, InFusion UK, and InFusion China, are scope values.

Exception Case: Predefined Scope Values

If there is a business need to pass certain predefined business object values to the tasks in a task list, similar to what scope enabled task list allows you to achieve by selecting scope values, but not to allow you to select the scope values, then you can use a special task parameter to specify predefined scope values at design time. When you use this functionality, you cannot select any scope values when performing any of the tasks in the given task list. Only the predefined values are passed by FSM.

Search and Manage Tasks

When you specify a business object as the scope of a task list, you must also specify a Search task, which is a task that allows you to search and select the scope values, and a Manage task, which is a task that allows you to create a scope value, for the business object. See Significance of Search and Manage Tasks for more information.

Note

When you create a new instance of scope using Manage task, you first create the value and return to FSM. Then you use Search task to search and select the newly created value. Also, you must have the appropriate permissions to create scope instances. Otherwise, you will not see the option to create new scope instances.

Task List Scope Dependencies

In some cases, a scope of a task list may have dependency on a scope defined on one of its parent task lists. In such cases, the dependency must be explicit by defining a parent-child relationship between the respective business objects. FSM will not implicitly derive the dependencies based on task list hierarchy. See Business Object Dependencies for more information.

Task List Scope and Setup Data Export

All task list scopes are always available as filters for setup data during setup export (setup data export scope). The scope values selected when you perform the tasks in the task list are available as filter values that can be used to filter exported setup data. However, setup data export scopes (filters) can also be defined without defining task list scopes. See Business Objects as Scopes for more information. Also see Implementing Export API – Identifying Filter Criteria for more information on how setup export and import web services should be programmed to use scope values for setup data filter during export.

Note

The predefined scope values (the exception case), which are defined as task list parameters, are not available as filters during setup data export.

See also:

Business Objects

Task Execution Programs – Scope Selection Enabled

Web Services for Setup Export and Import – Identifying Filter Criteria

Task Parameters

FSM supports defining parameters for tasks at two levels.

Parameter for a Task

In this case, the parameters are defined for a specific task and whenever the task is performed, irrespective of the task list that is used, FSM will pass the defined parameters to the corresponding task program.

In addition to the parameters you define, the following predefined parameters are always available.

- **Task Name:** FSM passes the task name as a parameter by default. You do not need to explicitly define it. The parameter is called pageTitle.
- **ID of a Task in an Implementation Project:** A task can belong in (meaning it can be an item of) many task lists in many implementation projects. FSM passes the ID that uniquely identifies a specific instance of a task in a specific task list through a parameter called iPTLItemKey. This parameter is passed by default. There is no need to specifically define it.
- **ID of Parent Task List in an Implementation Project:** FSM also passes the ID of the parent task list in an implementation project by default. The parameter is called iPTLParentItemKey.
- **Feature Selection:** Although referred generically as a feature, FSM supports passing current selection of offerings, options, and features as parameters. You can pass the current setting of the features to setup task programs by using the parameter called FS_FEATURE_SHORT_NAME.
- **Parent Feature Selection:** FSM also supports the case of deriving the selection of the dependent features by identifying the parent feature. In this case, a combination of two parameters called FS_PARENT_FEATURE and FS_FEATURE_LEVEL must be defined. FS_FEATURE_LEVEL indicates how many levels of dependent feature selections are passed to the task execution program.

Parameters for a Task in the Context of a Specific Task List

In these cases, if a task belongs to more than one task list, then FSM passes these parameters to the corresponding task program only when the task is performed from the given task list.

- **Predefined Task List Scope Values:** In this special case, application designers and developers may decide not to let you select a scope value, but instead define a specific scope value at design time and pass it to the ADF task flows for any or all of the tasks in a setup task list. In such cases, they will define a combination of parameters called BOShortName, BOAttributeCode, and BOAttrbuteValue. When these parameters are used, you do not get the option to selection scope values when performing those tasks.

Task Parameters and Setup Export and Import

FSM will always pass all parameters defined at the task levels to setup export and import web services. If and how the parameters are used by the web services, however, is up to the programming of the web services. On

the other hand, FSM will not pass the parameters defined for a task within the context of a specific task list to setup export and import web services. See Web Services for Setup Export and Import for more information on these web services.

Note

If parameters are defined, FSM will pass them to corresponding task programs; however, the task programs must be able to accept the parameters and take appropriate actions as desired.

If both levels of parameters are applicable to any given task, then parameters from both levels are passed.

See also:

Business Objects

Task Execution Programs – Parameter Enabled

Task Dependencies (Predecessor Tasks)

Setup tasks may have dependencies on one another. That is, performing a task successfully may depend on data entered via one or more other tasks. In such cases, task dependencies can be defined using predecessor tasks to make you aware of the dependency so that you can plan to perform the tasks in the correct order. A task can have one or more predecessor tasks.

Warning Event: For each predecessor of a given dependent task, you can define one of the following two conditions for performing the dependent task:

- **Start Status**: This condition indicates that setup of the predecessor task should at least be started but not necessarily completed before the dependent task is performed.
- **Completed Status:** This condition indicates that setup of predecessor task should be completed before performing the dependent task.

If the status of the predecessor task does not meet that of the specified condition, a warning appears in the application. In case a task has multiple predecessors, if condition of any one of the predecessors is not met, the warning will appear.

For example, if defining tax rates will have dependency on setting up tax regimes, then Mange Tax Regimes could be defined as the predecessor task of Manage Tax Rates. In addition, if all tax regimes must be defined first before you attempt to create tax rates, then warning condition can be defined as Completed Status. In this case, when you attempt to perform

Manage Tax Rates, if the status of Mange Tax Regimes is set at anything other than Completed, a warning appears. If, on the other hand, the warning status was defined as Start Status, then a warning appears if Mange Tax Regimes status is Not Started. For all other statuses of Mange Tax Regimes, no warning appears.

Note

Although a warning appears, you are not prevented from performing the dependent task if the predecessor condition is not met. The programs used for performing the dependent task is expected to ensure and prevent any data integrity issues as a result of performing dependent task ahead of the predecessor task.

Tip

Consider defining predecessors for any tasks that may have prerequisite tasks, which you should be aware of before you attempt to perform those tasks.

Viewing Lists of Task Lists and Tasks

1. Go to the Setup and Maintenance work area.
2. Click Manage Task Lists and Tasks from the task pane, which will display a page with the same name. This page will show a list of task lists or tasks based on your default search setting. If no personalization was performed, the default search setting of this page is to display all task lists.
3. If the page does not show a list of tasks or task lists when first displayed, then perform one of the following actions to return an appropriate list:
    - **Ad hoc search:** In the Search region, select Task Lists or Tasks in the Search field, enter the appropriate search criteria, and click Search.
    - **Saved search:** Select the desired saved search from the Saved Search field.
4. From this page you can:
    - View and edit task list
    - Create task list
    - Delete task list

- View and edit task
- Create task
- Delete task
- Define predecessor tasks

See also:

Task Lists and Tasks – Design Considerations

Viewing and Editing Task List Details

Note

You should review design considerations to understand various design implications before you start creating or editing task list or task details, especially if you are doing so for the first time.

1. From the list of task lists on the Manage Task Lists and Tasks page, perform one of the following actions to open the Edit Task List page:
   - Click the task list name.
   - Select the desired row and click Edit on the table task bar.
   - Select the desired row and click Edit from the Actions menu on the table task bar.
2. View or edit task list details.
3. Click Save or Save and Close to save changes.

Note

You can view Oracle-delivered task lists, but cannot edit or delete them.

See also:

Task Lists and Tasks – Design Considerations

Viewing Lists of Task Lists and Tasks

Creating Task Lists

1. From the list of task lists on the Manage Task Lists and Tasks page, perform one of the following actions to open the Create Task List page.
   - Click Create Task List on the table task bar.

- Click Create Task List from the Actions menu on the table task bar.

2. Enter task list details.
3. Click Save or Save and Close to save changes.

See also:

Task Lists and Tasks – Design Considerations

Viewing Lists of Task Lists and Tasks

Task List Details

All task lists must have basic information and associated tasks. You may also specify required tasks, task parameters, and task list scopes.

Entering Basic Information

When creating a new task list or editing an existing task list, enter or modify following fields. If the Basic Information region is collapsed, expand it.

Field	Field Type	Required	Description
Name	Text	Yes	Use a unique name for the task list which you can easily identify.
Code	Text	Yes	Use a unique code to identify the task list. Once created, code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the task list.
Use Only In Task List	List of Values	No	**Defining Non-Sharable Task Lists:** Use this field to identify that a task list cannot be shared. Search and select the task list which can be the only parent of the current task list. By definition, all task lists are sharable, meaning they can be included in multiple parent task lists. However, if a task list must be included in only one parent, then specify the only parent in this field.
Help Topic ID	Text	No	Leave this field blank.

Note

Also review best practices and standards and guidelines for defining these core fields.

Associating Tasks

When creating a new task list or editing an existing task list, associate tasks to task lists using the following steps.

Adding Task Associations

1. To associate one or more tasks, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate tasks or task lists from the popup window.
3. Apply and close the window when complete.

Removing Setup Task and Task List Associations

To remove an existing association:

1. Select the appropriate row in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Changing Task Display Sequence

To change the display sequence, select the row with the sequence that you want to change, and drag and drop the row onto the row after which you want it to appear. For example, if you select T1, and drag and drop it onto T2, then T1 will appear after T2.

Note

The items (or members) of a task list may be referred to as tasks, although they may be tasks or other task lists.

Specifying Required Tasks

When creating a new task list or editing an existing task list, specify tasks as required using the following steps.

1. If the Tasks region is collapsed, expand it.

2. Select the row for the desired task or task list.
3. Select the corresponding checkbox in the Required Task column.
4. If the Required Task column is not visible:
    - Select View > Columns from the table task bar.
    - Enable the Required Task column.

Note

You can set the Required Task option only for the immediate children.

Defining Task Parameters

When creating a new task list or editing an existing task list, specify parameter for an item in the task list using the following steps.

1. If the Tasks region is collapsed, expand it.
2. Select the row for the desired task or task list.
3. Enter parameter and value pairs in the corresponding Parameters column. If defining multiple parameters, then use an ampersand (&) as a separator.
    - For example: param1=value1&param2=value2
4. If the Parameter column is not visible:
    - Select View > Columns from the table task bar.
    - Enable the Parameter column.

Special Case: Predefined Scope Values

In this case, a scope value for specific tasks in a setup task list can be defined at design time. A combination of three parameters needs to be used in the following format:

```
BOShortName=<value>&BOAttributeCode=<value>
&BOAttributeValue=<value>
```

The values specified for these parameters should be as follows:

- **BOShortName**: The short name (code) of the business object used as the scope of the task list. See Defining Task List Scope for more information.
- **BOAttributeCode**: The code of the scope value to be passed.
- **BOAttributeValue**: The scope value to be passed.

For example, subledger accounting rules are used by many different Oracle Fusion Applications. The setups of the rules are different depending on which application they apply to. Therefore, the Subledger Application setup business object is defined as the scope for the Define Subledger Accounting Rules task list and the scope attribute of the business object is APPLICATION_ID. However, you are not permitted to select the scope values because those are defined at the design time based on various parent task lists of the Define Subledger Accounting Rules task list. Therefore, the predefined scope values can be passed as task list parameters as follows:

```
BOShortName=XLA_SUBLEDGER&BOAttributeCode=APPLICATION_ID
&BOAttributeValue=PAYROLL
```

Note

When these parameters are defined, if the same task list has a scope defined with the same code as the specified value of BOShortName parameter, you cannot select a scope value.

Defining Task List Scope

When creating a new task list or editing an existing task list, specify task list scope by using the following steps.

1. If the Task List Scope region is collapsed, expand it.
2. Select a business object.
3. The Search Task and Manage Task fields are not editable. They are populated based on the information defined on the selected business object.
    - If the selected business object does not have a Search Task and Manage Task defined, a warning message appears.

See also:

Business Objects

Note

Ensure that Search Task and Manage Task are defined for business objects that are selected as scope of a task list. Otherwise, scope will be unusable because you will not be able to search and select or create any instances of scope value.

Tip

If you want to define the scope value of a task list at design time and pass it to ADF task flows of specific tasks belonging to the task list, then use the task parameter. In this case, you cannot select scope values.

Defining Scope Dependencies

Dependency or parent-child relationships between two scopes are defined via parent-child relationships between their respective business objects. See Defining Parent-Child Dependencies for Business Objects for more information.

Finding Options and Features Associated with Task Lists

1. From the list of task lists on the Manage Task Lists and Tasks page, select the desired row.
2. Click the icon in the Features column to open a page displaying the associated options and features.
3. If the Features column is not visible:
    - Select View > Columns from the table task bar.
    - Enable the Features column.

Finding Parents of Task Lists

1. From the list of task lists on the Manage Task Lists and Tasks page, select the desired row.
2. Click the number link in the Occurrences column to open a window displaying the parent task lists.
3. If the Occurrences column is not visible:
    - Select View > Columns from the table task bar.
    - Enable the Occurrences column.

Note

If the Occurrences column contains a zero (0), it indicates that the task list does not have any parents, and therefore there is no link to open the window.

Deleting Task Lists

1. From the list of task lists on the Manage Task Lists and Tasks page, select the desired row.
2. Perform one of the following actions:
    - Click Delete on the table task bar.
    - Click Delete from the Action menu on the table task bar.

See also:

Viewing Lists of Task Lists and Tasks

Viewing and Editing Task Details

Note

You should review design considerations to understand various design implications before you start creating or editing task list or task details, especially if you are doing so for the first time.

---

1. From the list of tasks on the Manage Task Lists and Tasks page, perform one of the following actions to open the Edit Task page:
    - Click the task name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Actions menu on the table task bar.
2. View or edit task details.
3. Click Save or Save and Close to save changes.

Note

You can view Oracle-delivered tasks, but cannot edit or delete them.

See also:

Task Lists and Tasks – Design Considerations

Viewing Lists of Task Lists and Tasks

Creating Tasks

1. From the list of tasks on the Manage Task Lists and Tasks page, perform one of the following actions to open the Create Task page.
    - Click Create Task on the table task bar.
    - Click Create Task from the Actions menu on the table task bar.
2. Enter task details.
3. Click Save or Save and Close to save changes.

See also:

Task Lists and Tasks – Design Considerations

Viewing Lists of Task Lists and Tasks

Task Details

Typically all tasks should have basic information, task program information, and associated business objects. The only exceptions are the tasks using deployment methods, which will not have associated business objects, and will have deployment method attributes.

Entering Basic Information

When creating a new task or editing an existing task, enter or modify following fields. If the Basic Information region is collapsed, expand it.

Field	Field Type	Required	Description
Name	Text	Yes	Use a unique name for the task list which you can easily identify.
Code	Text	Yes	Use a unique code to identify the task list. Once created, code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the task list.
Deployment Method	List of Values	No	Default setting is none, which indicates that the task uses business objects for its setup data. Pick a value other than none only if the task is for setting up data for one of the deployment methods. See Defining Tasks for Deployment Methods for more information.
Uses user interface	Checkbox	No	Indicates whether this task is performed via UI or web service. Checked: Associated task program is expected to be UI.

			This is the default setting. Not Checked: Associated task program is expected to be a web service.
Open In	Radio Button	No	Standard View: The UI is displayed with standard Oracle Fusion UI shell. This is the default setting Full Screen View: The UI is displayed without standard UI shell.
Help Topic ID	Text	No	

Note

Review best practices and standards and guidelines for defining these core fields.

If selected deployment method is none, then business objects should be associated with the task. Otherwise, deployment method attributes may need to be entered.

Associating Business Objects

When creating a new task or editing an existing task, associate business objects using the following steps.

Adding Business Object Associations

1. To associate one or more business objects, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select appropriate the business objects from the window.
3. Apply and close the window when complete.

Removing Business Object Associations

To remove an existing association:

1. Select the appropriate row in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Note

Business object associations are only applicable to those tasks with deployment methods that are selected as none in the Basic Information region. If a business object is not defined for such tasks, then FSM is not able to identify the setup data during export and import process.

Entering Task Program Information

While the basic information fields are defined during design process, the task program-specific information is typically added later after the programs are developed. Once the programs are developed and related information is available, edit an existing task, and enter the task program specific information.

If the Basic Information region is collapsed, expand it.

Field	Field Type	Required	Description
Program	Text	No	Enter the program name (ADF task flow or web service name) that will be used to perform the task. For example: /WEB-INF/oracle/apps/financials/tax/configuration/regime/ui/flow/TransTaxSearchFlow.xml#TransTaxSearchFlow
Enterprise Application	List of Values	No	Pick the enterprise application that will be used to deploy the UI or web service program.
Module	List of Values	No	Pick the module in the selected enterprise application that will be used by the UI or web service program.

Warning

If any tasks are missing program information, then it is implied that you will not perform those tasks from FSM.

See also:

Task Execution Program Requirements - Design Considerations

Task Execution Program – FSM Requirements

Testing Task Programs

Testing Task Programs

When application developer finishes developing task execution programs, they often want to test their execution from FSM before deploying them to the live environment. Edit an existing task to test the task program by using the following steps.

1. If the Basic Information region is collapsed, expand it.
2. Enter full path to access the task program including the host and port information in Test URL text field.
    - The path can point to any instance where the program is available.
3. If the program is not secured, then append the following string at the end of the URL: &secureFSProducer=N; that is:
    - **Secured Program:** http://rws65042fwks.us.oracle.com:8988/regime-ui-context-root/
    - **Unsecured Program**: http://rws65042fwks.us.oracle.com:8988/regime-ui-context-root/&secureFSProducer=N
4. Save changes.
5. Click Test Go to Task to test program execution.

Warning

If you enter a test URL, then enterprise application and module information entered for the same task are ignored by FSM. It also adds overhead when the task is executed. Make sure to remove value from the Test URL field before making the task available for use.

Defining Web Services for Export and Import of Related Setup Data

The web services necessary to export and import setup data related to a setup task, must be defined for each of the business objects associated with the task.

1. View task details to find associated business objects.
2. Define export and import services for each of the associated business objects.

Defining Task Parameters

When creating a new task or editing an existing task, use the following steps to define task parameters.

1. If the Basic Information region is collapsed, expand it.
2. In the Parameters field, and enter any parameter that you want FSM to pass to the corresponding task program.

3. Enter parameter and value pairs. If defining multiple parameters, then use an ampersand (&) as a separator.
    - For example: param1=value1&param2=value2

Special Cases of Task Parameters

Offering, Option, and Feature Selection: If you want to pass the offering, option, and feature selections made to the ADF task flow of a setup task, then use name-value pairs as: FS_FEATURE_SHORT_NAME=<value>. The value of FS_FEATURE_SHORT_NAME should be the short name (or code) of the offering, option, and feature whose selection you want to pass.

For example, FS_FEATURE_SHORT_NAME = ZX_REGIME_REG_STATUS, where ZX_REGIME_REG_STATUS is the short name (code) of a feature called Tax Regime Registration.

Dependent Option and Feature Selection: In this case, the parent offering, option, or feature is specified as parameter but the ultimate goal is to derive the selected choices of the dependents. Use name-value pairs as follows:

```
FS_PARENT_FEATURE=<value>&FS_FEATURE_LEVEL=< 1, 2, 3, and
so on>
```

The value of FS_PARENT_FEATURE should be the short name (or code) of the parent offering, option, or feature, and FS_FEATURE_LEVEL should indicate how many levels of children should be returned. For example, 1 indicates immediate children, 2 indicates immediate children and their children, and so on.

Note

If parameters are specified, FSM will pass them to corresponding task programs. However, the task programs must be able to accept the parameters and take appropriate actions.

Defining Tasks for Deployment Methods

When creating a new task or editing an existing task, if the task is for setting up data for one of the deployment methods, then use the following steps.

1. If the Basic Information region is collapsed, expand it.
2. Select the deployment method.
3. If the Deployment Method region is collapsed, expand it.
4. Enter appropriate value for the deployment method attribute.

Note

The Deployment Method attributes section is only visible and applicable if deployment method of the task (in the Basic Information region) is a value other than None.

See Special Case: Setting Up Deployment Methods under Task Execution Programs - Design Considerations for more information.

Warning

Explanation of various types of Oracle-delivered deployment methods, meaning applications core setups, and standards and guidelines for defining those setup tasks, are beyond the scope of this document. If you are not familiar with them, consult the Oracle® Fusion Applications Developer's Guide and related training material.

Defining Predecessor Tasks

1. From the list of tasks on the Manage Task Lists and Tasks page, select the desired row.
2. Click the number link in the Predecessor Tasks column to open the Edit Predecessor Tasks page.
3. If the Predecessor Tasks column is not visible:
    - Select View > Columns from the table task bar.
    - Enable the Predecessor Tasks column.
4. Add or remove predecessor tasks by performing the following steps.

Adding Predecessor Task Associations

1. To associate one or more predecessor tasks, perform one of the following two actions:
    - Click Select and Add on the table task bar.
    - Click Select and Add from the Action menu on the table task bar.
2. Search and select the appropriate tasks from the window.
3. Apply and close the window when complete.

Removing Predecessor Task Associations

To remove an existing association:

1. Select the appropriate row in the table.
2. Perform one of the following two actions:
    - Click Remove on the table task bar.
    - Click Remove from the Action menu on the table task bar.

Defining Warning Message Event

To apply the same event to all predecessor tasks:

1. Select the Apply Same Warning Message Event to All Tasks option at the top of the page.
2. Select one of the two available options in Warning Message Event.

To apply different events to different predecessor tasks:

1. Deselect the Apply Same Warning Message Event to All Tasks option if it is already selected.
2. Select the desired row in the table.
3. Select the value from the list of values in the corresponding Warning Message Event column.
4. Click Save or Save and Close to save changes.

Note

Review Effect of Defining Task Dependencies - Design Considerations for more information.

Finding Options and Features Associated with Tasks

1. From the list of tasks on the Manage Task Lists and Tasks page, select the desired row.
2. Click the icon in the Features column to open a page displaying the associated options and features.
3. If the Features column is not visible:
    - Select View > Columns from the table task bar.
    - Enable the Features column.

Finding Parent Task Lists of Tasks

1. From the list of tasks on the Manage Task Lists and Tasks page, select the desired row.
2. Click the number link in the Occurrences column to open a window displaying the parent task lists.
3. If the Occurrences column is not visible:
    - Select View > Columns from the table task bar.
    - Enable the Occurrences column.

Note

If the Occurrences column contains a zero (0), it indicates that the task list does not have any parents, and therefore there is no link to open the window.

Deleting Tasks

1. From the list of tasks on the Manage Task Lists and Tasks page, select the desired row.
2. Perform one of the following actions:
    - Click Delete on the table task bar.
    - Click Delete from the Action menu on the table task bar.

See also:

Viewing Lists of Task Lists and Tasks

6

Business Objects

Design Considerations

About Business Objects

Business objects are logical representation of real-world objects. They represent Oracle Fusion Applications data. Some of the examples of business objects are Ledger, Business Units, Taxes, Items, Orders, Opportunities, Campaign, and Employees.

In ADF terminology, a business object involves one or more View Objects (VO) mapped to underlying entity objects (EO), which are related through composition (meaning that child entities cannot exist without their parents).

Role of Business Objects in FSM Export and Import

FSM provides export and import functionality to move Oracle Fusion Applications setup from one instance into another. Business objects are used to read and write the correct set of setup data at the source and the target instances respectively.

To export and import setup, you will first identify, at the source instance, the implementation they want to move and then export it. FSM uses the list of tasks used in the identified implementation as a template to determine what setup data will be exported. It then finds the setup data based on the associated business objects of those tasks. Similarly, during import, FSM writes back the data according to the business objects.

Since web services are used to read and write setup data during export and import, business objects associated with setup tasks must also have such web services available.

Warning

Provide web services to read from and write to all setup business objects. If web services are not registered for any business objects, it indicates that the corresponding setup data will not be exported and imported by FSM.

See also:

Business Object associations with Setup Tasks

Web Services for Setup Export and Import

Business Objects as Scope

A business object can be a scope of a setup task list or of exported setup data.

- **Setup Task List Scope:** If a business object is defined as the scope of a setup task list, you have the option to enter setup data for the tasks contained in the task list in the context of a scope value. For more information see Task List Scope.
- **Setup Data Export Scope:** A business object can also be a scope (setup data filter) during setup data export. For example, if Financials is set up for multiple legal entities, different legal entities can be exported at separate times by picking different scopes (different values of legal entities) during setup data export.

  While all task list scopes are automatically designated as setup data export scopes as well, a business object can be setup data export scope without being a task list scope.

Any business object intended to be used as a scope for either setup task lists or setup data export must have a Search task specified for it. In case of designated task list scope, the business object must also have a Manage task specified for it. See Significance of Search and Manage Tasks for more information.

Tip

Consider defining a business object as scope if the business object is often used as a qualifier to uniquely identify data of other business objects. That is, the business objects whose values often act as the foreign keys to data of other business objects in the physical data model are good candidates for defining as scope, especially if business processes and functions are often segmented by them as well.

See also:

Task Execution Programs – Design Considerations for Scope Selection Enabled

Task Execution Programs – Making Scope Selection Enabled

Web Services for Setup Export and Import – Identifying Filter Criteria

Business Objects as Deployment Methods

In Oracle Fusion Applications, certain business object may be designated as deployment methods. If a business object is defined as deployment method, then it implies that the business object represents a common entity which is set up using a common interface. One or more attributes of the business object must then be specified as the qualifying identifiers whose values will typically identify specific instances of the object.

For example, Applications Profile Option is a deployment method where the attribute, Profile Option Name, identifies a specific profile option for which you may be setting profile values.

Many different setup tasks may be using a deployment method for their setup data. However, for each of those setup tasks, the appropriate value for the qualifying attribute must be specified, which will be used to identify the specific instances of the deployment method as applicable to a given setup task.

For example, Manage Calendar Profile Option setup task uses a deployment method called Application Profile Option, as do many other setup tasks such as Manage Currency Profile Option and Manage General Ledger Profile Option. Since qualifying attribute of Application Profile Option is defined as Profile Option Name, when Manage Calendar Profile Option task is created, it requires that the value of Application Profile Option be set to the appropriate code that identifies the Calendar Profile Option (such as ZCA_COMMON_CALENDAR).

When setup tasks for deployment methods are performed, or the data is exported or imported, a standard task execution program and export and import web service, specified as the Manage task for deployment method, are invoked by FSM. The qualifying attribute value specified for a given task is passed as a parameter to the standard programs for the deployment method to identify the applicable instance of the common entity. See Significance of Search and Manage Tasks for more information.

See also:

Task Execution Program - Design Considerations

Web Services for Setup Export and Import – Identifying Filter Criteria

Significance of Search and Manage Tasks

A business object should be associated with a Search task and a Manage task if it will be used in any of the following cases:

- As a task list scope.
- As a scope for setup data export.

- As a deployment method.

Search Task

The Search task should allow you to find a list of existing values of the business object. The Manage task should let you create, edit, or delete business object values.

When you select a scope value, either to perform an iterative task in a task list or to filter setup data during export process, FSM uses the task execution program associated with the designated search task to enable you to find and select an appropriate value.

Manage Task

When you create a new scope value before making a scope selection, FSM will present the corresponding Manage task program. The Manage task is also important if the business object is designated as a deployment method. Its associated program is used as the standard interfaces for entering and exporting and importing data for any task using the deployment method.

Tip

Task Execution Program and Export and Import web services for Search and Manage tasks must be designed such that they provide you with the ability to perform all expected data search and data management actions, such as create, edit, and delete.

Note

The task execution program of a task designated as the search task of a business object must not only support searching and selecting a value, but also must support returning the value to FSM.

Business Object Dependencies

If there is data dependency between two business objects then parent-child relationship can be defined between them. Typically this signifies that parent business object is needed to fully qualify the child business object. For example, Tax Regime is the parent business object while Tax Rate is its child business object. This means, that to fully identify a tax rate one needs to know the tax regime as well.

In FSM, parent-child relationships between business objects are especially important if you need to define hierarchical task list scopes. Those relationships between the scopes must be defined through the parent-child relationships of the respective business objects.

For example, Tax Regime is the scope for Define Tax Rate and if Legal Entity is the scope for Define Tax Regime, then if you want to select both scopes, Legal Entity and Tax Regime, when defining Tax Rate, Legal Entity must be defined as the parent business object of Tax Regime.

A child or dependent business object can have only one parent, while a parent may have multiple children.

Viewing Lists of Business Objects

1. Go to the Setup and Maintenance work area.
2. Click Manage Business Objects from the Tasks pane to open a page with the same name. This page will show a list of business objects based on your default search setting. If no personalization was performed, the default search setting of this page is to display all business objects.
3. If the page does not show a list of business objects when first displayed, then perform one of the following actions to return an appropriate list:
    - **Ad hoc search:** In the Search region, select Business Objects in the Search field, enter the appropriate search criteria, and click Search.
    - **Saved search:** Select the desired saved search from the Saved Search field.
4. From this page you can:
    - View and edit business objects
    - Create business objects
    - Delete business objects
    - View related tasks

Viewing and Editing Business Objects

Note

You should review design considerations to understand various design implications before you start creating or editing business object details, especially if you are doing so for the first time.

1. From the list of business objects on Manage Business Objects page perform one of the following actions to open the Edit Business Object page:
    - Click the business object name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Actions menu on the table task bar.
2. View or edit business object details.
3. Click Save or Save and Close to save changes.

Note

You can view Oracle-delivered business objects, but cannot edit or delete them.

See also:

Business Object – Design Considerations

Viewing Lists of Business Objects

Creating Business Objects

1. From the list of business objects on the Manage Business Objects page, perform one of the following actions to open the Create Business Object page.
    - Click Create Business Object on the table task bar.
    - Click Create Business Object from the Actions menu on the table task bar.
2. Enter business object details.
3. Click Save or Save and Close to save changes.

See also:

Business Objects – Design Considerations

Viewing Lists of Business Objects

Business Object Details

At minimum, all business objects should have basic information and associated web services for setup export and import. You may also specify the parent-child relationship, supertype-subtype relationship, or define the business object as deployment method.

Entering Basic Information

When creating a new business object or editing an existing business object, enter or modify following fields. If the Basic Information region is collapsed, expand it.

Field	Field Type	Required	Description
Name	Text	Yes	Use a unique name for the business object which you can easily identify.
Code	Text	Yes	Use a unique code to identify the business object. Once created, code cannot be changed.
Description	Text	No	Although not required, always provide a brief, meaningful description that will help you understand the purpose of the business object.
Search Task	Text	No	If the business object is defined as scope for either a setup task list or for filtering setup data during export, then this field must include the task name which can be performed to find an appropriate value.
Manage Task	Text	No	If the business object is defined as scope for a setup task list or as a deployment method, then this field must include the task name which can be performed to create, edit, and delete appropriate values.
Subtype Discriminator	Text	No	This is only applicable if the business object is a super type Business object. Enter the discriminating attribute that identifies the subtypes in the super type business object.

Note

Also review best practices and standards and guidelines for defining these core fields.

Defining Export and Import Services

While the basic information fields are defined during design process, the export and import service specific information will be typically added later after the web services are developed. Once the web services are developed and related information is available, edit an existing business object, and enter the web service specific information.

If the Service region is collapsed, expand it.

Field	Field Type	Required	Description
Service	Text	No	Enter full path name to the web service that will be used to perform export and import of the business object. For example: oracle.apps.financials.tax.configuration.regime.regimeService.TaxRegimeService
Enterprise Application	List of Values	No	Pick the enterprise application that is used to deploy the export and import web service.
Module	List of Values	No	Pick the module in the selected enterprise application that is used by the web service.

Warning

If any business objects are missing service information, then it is implied that you will not export and import those setup data using FSM.

Note

You may want to test your web service before deploying it to the appropriate applications server.

Testing Export and Import Services

When application developers finish developing setup export and import services, they often want to test the web services from FSM before deploying them to the live environment. Edit an existing business object to test the web services by using the following steps.

1. If the Service region is collapsed, expand it.
2. Enter the full path to access the web service including the host and port information in Test URL text field. The path can point to any instance where the web service is available.
    - For example: http://ap6008fems.us.oracle.com:6033/finCommon/finSer

oracle.apps.financials.tax.configuration.regime.regimeService.TaxRegimeService2

Warning

If you enter a test URL, then enterprise application and module information entered for the same business object are ignored by FSM. It also adds overhead when the task is executed. Make sure to remove value from the Test URL field before making the task available for use.

Defining Parent-Child Dependencies

The parent-child relationship between two business objects is defined at the child or dependent business object.

When creating a new business object or editing an existing business object, enter or modify the associated parent business object using the following steps.

1. If the Dependencies region is collapsed, expand it.
2. Select the parent business object from the Parent Business Object field.

Defining Deployment Methods

When creating a new business object or editing an existing business object, specify the business object as a deployment method using the following steps.

1. If the Basic Information region is collapsed, expand it.
2. Select the Create as Deployment Method option.
3. If the Attributes region is collapsed, expand it.
4. Add or remove attribute names that uniquely identify each instance of this business object, meaning the alternate key for the business object.
    - For example, if the Application Profile Options business object is specified as a deployment method, then Profile Option Name is the attribute entered in this region because Profile Option Name identifies each profile option, meaning it is the alternate key.

Defining Supertype-Subtype Business Objects

These are special type of business object dependencies. Super type business objects are the primary business objects, and the sub types

identify specific segments within them. For example, Person may be a super type business object with subtypes of Employees, Contractors, Partners, and Customers.

In technical terms, a super type business object often becomes the database table in the physical data model, while the sub types are identified using a discriminating attribute in the same database table.

When creating a new business object or editing an existing business object, specify the business object as a super type or subtype business object using the following steps.

Super Type Business Objects

1. If the Basic Information region is collapsed, expand it.
2. Enter the attribute that is used as subtype discriminator of its sub types in the Subtype Discriminator text field.
    - For example, when defining the Person business object, enter Type, which is the attribute that will identify the subtypes.

Subtype Business Objects

1. If the Dependencies region is collapsed, expand it.
2. Select the super type business object in the Supertype Business Object field.
3. Enter the subtype discriminator value of the supertype business object.

Deleting Business Objects

1. From the list of tasks on the Manage Business Objects page, select the desired row.
2. Perform one of the following actions:
    - Click Delete on the table task bar.
    - Click Delete from the Action menu on the table task bar.

Finding Related Tasks

1. From the list of tasks on the Manage Business Objects page, select the desired row.
2. Observe the corresponding Related Tasks column.
3. Find related tasks as follows:

- If only one task is associated with the business object, then it will appear in the column.
- If multiple tasks are associated with the business object, then the first task appears in the column followed by (x more), where x represents the additional number of associated tasks.
- Scroll the mouse cursor over (x more) to view up to eight additional associated tasks.
- If more than eight additional associated tasks exist, then click (x more) to view the entire list of tasks associated with the business object.

4. If the Related Tasks column is not visible:
    - Select View > Columns from the table task bar.
    - Enable the Related Tasks column.

7

Task Execution Programs - FSM Requirements

Design Considerations

Warning

Application developers building task execution programs should be knowledgeable and experienced in ADF and JDeveloper technologies and their standards and guidelines.

About Setup Task Execution Programs

All setup tasks are expected to use either UI or web services to allow you to enter and maintain corresponding setup data. Therefore, either an ADF bounded task flow (in case of UI) or a web service should be associated with each setup task, which, in turn, will be used to perform the task.

ADF Task Flows

FSM uses Portal to invoke ADF bounded task flows as a portlet. FSM renders the ADF task flow as a dynamic region within an FSM page fragment. Because of the current limitation of ADF, you must define the task flows at design time and cannot call at runtime using Expression Language (EL) to resolve document and ID information.

To provide consistent user experience, the following standards are strongly recommended for setup task UIs.

- Must use UI Shell Main Area template
- Must include FSM Shared Library
- Must return control to FSM when executed from FSM
- Should be Task Parameter enabled, if applicable
- Should be Scope Selection enabled, if applicable
- Recommended to be Go To Next enabled
- Must be Security enabled

Web Services

Setup tasks can be performed using web services instead of a UI if the task is performed using a programmatic process rather than requiring user interaction. When a setup task execution program is a web service, it is up to the developer of the service to decide the approach and technology to use to build the web service. FSM only requires:

- A service API called lanuchProcess() is included in the web service.
- If the service submits a background process to execute setup of the task, then the task status should be updated by the program when the background process completes. FSM provides public web services to update task status that can be used to set the completion status of the background process.

See also:

Task Lists and Tasks: Design Considerations - Task Execution Programs

Warning

If a setup task does not have an associated ADF task flow or a web service, then you cannot perform the task from FSM and must enter and maintain corresponding setup data outside of FSM.

ADF Task Flow Template

You must use the UI Shell Main Area template to create the ADF bounded task flows for the setup tasks. When a setup task is performed from FSM, the FSM main area is replaced with the ADF task flow of the setup task. Therefore, you must use the correct template to render the local and the contextual areas of ADF task flows correctly in FSM main area.

FSM Shared Library

All ADF task flows that are used to perform setup tasks must include a shared library called AdfFunctionalSetupPublicUI.jar provided by FSM. The shared library will enable ADF task flows to:

- Be displayed as a portlet in the FSM main UI area. FSM uses Portal and renders ADF task flows for setup tasks as portlets.
- Return scope selections to FSM.
- Get the next tasks from FSM when implementing Go to Next feature.

Returning Control to FSM Task List

If an ADF task flow is invoked from FSM, then navigational control must be returned to FSM once the ADF task flow is closed. This is required to allow you to continue with performing the setup tasks (meaning to invoke the corresponding ADF task flows) in the correct, logical sequence as specified via a setup task list.

You can use one of the following two options provided by FSM to achieve this functionality.

- Declarative component called **setupButtonMenu**
- API called setupBean.returnControlToSetup()

Parameter Enabled

FSM allows setup task designers and developers to define parameters. See Task Parameters for more information. If you define parameters, FSM will pass them to corresponding ADF task flow at runtime when task flows are invoked. For those ADF task flows, you should ensure that the ADF task flows are able to accept the passed parameters and use them to achieve the intended outcome. Parameters can be passed from FSM to control a wide range of functionality in setup task UIs such as setting default values for certain fields, filter setup data, or use for validation, and so on.

Default Task Parameters

FSM always passes the following parameters without requiring them to be defined explicitly.

- Task name: Since it is typically used to comply with the Oracle Fusion page title standards, the parameter is often referred as page title as well.
- ID of a task in a specific implementation project.
- ID of the parent of a task in a specific implementation project.

Offering, Option, and Feature Selections as Task Parameters

If you want to control how ADF task flow used by a setup task behave based either selecting offerings, options, and features, or selecting the parent of an option or feature, then you can use FSM parameters to achieve it as well.

Predefined Scope Value

Task parameters can also be used to predefine the Scope values during design time. This functionality requires a specific combination of three parameters to be defined. When defined, you cannot select a scope value when you perform the task.

See also:

Defining Task Parameters

Entering Task Program Information

Scope Selection Enabled

This is applicable to ADF task flows of the setup tasks that are specified as Search task of a business object. In case of scope enabled setup task lists, this is also applicable to the consumer tasks in the task list, meaning the tasks that will accept the selected scope value.

Search Task UI

The ADF task flow used by the Search task of a scope is used to search and select a scope value either for a scope enabled task list or for filtering data during setup export. Therefore, these ADF task flows must be able to return the following to FSM:

- Scope (business object) attribute name
- The alternate key that identifies the scope instance
- The selected scope value

Target (Consumer) Task UI

In a scope enabled task list, when you select a scope and then invoke an ADF task flow of a task in that task list, FSM passes the selected scope value as a task flow parameter using key-value pair. The consumer (the target ADF task flow), however, should be able to accept the parameter and take desired actions as intended by the designer of the ADF task flow. The target ADF task flow can either use an EL expression or HashMap.

See also:

Defining Task List Scope

Business Objects as Scope

Significance of Search and Manage Tasks

Business Object Details

Save and Go to Task Enabled

Although not required, we recommend that setup UIs implement the ability to invoke another ADF task flow used by another setup task when the first UI is closed. This feature enhances usability by providing navigational shortcut to setup tasks without needing to return to FSM task list to perform subsequent tasks.

If implemented, the Save and Close button on an ADF task flow will include an option called Save and Go to Task when the ADF task flow is invoked from FSM. The option will have a sub menu showing the next three applicable setup tasks in the task list from where the ADF task flow was invoked.

For example, Task1, Task2, Task7, and Task9 are four tasks in Tasklist1 assigned to Enduser1. If the declarative component is added to the ADF task flows of the tasks, then when Enduser1 invokes Task1 from FSM, the Save and Close button on ADF task flow for Task1 will have an addition option called Save and Go to Task with a sub menu showing Task2, Task7, and Task9.

Tip

Do not implement this functionality in the ADF task flows that are used as the Search tasks for scope selection. You should be returned to FSM after searching and selecting a scope value.

Security Enabled

ADF task flows used by setup tasks must be properly secured using Oracle Fusion Security model. You should use the appropriate privileges, which should roll up to a proper job roles that are expected to perform the setup tasks. The same job roles will also require access to the Setup and Maintenance work area to use FSM, and therefore, must inherit ASM_FUNCTIONAL_SETUPS_ABSTRACT role.

FSM uses Portlet to invoke these ADF task flows. In Oracle Fusion, portlets are implemented as web services for remote portlets (WSRP) and are essentially web services. Therefore, they should be secured just like any other web services. Since the ADF task flows for setup tasks are considered the producer portlets, standards and guidelines for securing producer portlets should be followed.

API for Web Services Used in Executing Setup Tasks

If a setup task uses a web service for its execution, a method called launchProcess() must be included in the Service. The method must be able to:

- Accept the parameters passed by FSM and able to achieve the intended outcome. These parameters are defined by the task and task list designers and developers.
- Submit a background process (if applicable) or execute a program.
- Return a request process ID (for the submitted background process) and task status.
- Return status details (optional).

Developing ADF Task Flows for Setup Tasks

Warning

You should be experienced in developing applications using Oracle platforms and knowledgeable in ADF and JDeveloper. You should also be familiar with Oracle Fusion Applications standard and guidelines.

Note

All instructions assume that you are using JDeveloper as your development platform.

You must use the UI Shell Main Area template to build the ADF task flows.

Follow all ADF and Oracle Fusion Applications standards and guidelines.

Adding the FSM Shared Library

Add the shared library provided by FSM by using the following steps. In addition to adding the shared library to your project, which is required to make the ADF task flows for setup tasks compliant with FSM requirements, these steps will also create an entry in portlet.xml for FSM portlet called FSGeneicTaskFlow.

1. Select the relevant UI project.

2. Manually review web.xml, portlet.xml, and oracle-portlet.xml files, if they exist.
3. Click the Project Properties for this project.
4. Navigate to Libraries and Classpath > ADF Library > Edit and then delete the entry: adflibs/fs/AdfFunctionalSetupPublicUI.jar
    - Click Yes if warning message is posted.
5. Highlight the project to which you want to add the library.
6. Right click on the project name and select Functional Setup > Add Portlet Entries of FS Task Flow from the context menu.
7. If successful, a success confirmation message is posted confirming that FSGenericTaskFlow has been added successfully. Click OK on the dialog box.
8. If not successful, and you get a message that the process failed to add FSGenericTaskFlow, it typically means that FSM portlet entry already exists in portlet.xml. Follow the instructions in the failure message to verify.
9. Verify successful completion of adding FSM portlet entry:
    - Ensure that all jar file references are removed.
    - After all jar files references are removed and corresponding library references are introduced, build all of your projects to ensure they are building without errors.
    - Merge in your changes.
10. Validate that FSGenericTaskFlow is part of your deployed enterprise application.
    - Copy the URL of your deployed enterprise application and paste it as a browser URL.
    - Use up to the context root in the URL of your deployment enterprise application; for example: http://adc60153fems.us.oracle.com:6129/commonComponents-SuperWeb-context-root/
    - You should see a page titled WSRP Producer Test Page and at the top of the page it should show something similar to: CommonComponents_FSGenericTaskFlow (2.0)
    - The xx_FSGenericTaskFlow indicates that FS Shared Library has been included and FSGenericTaskFlow portlet is available.

Warning

If you do not add the shared library following these steps, your ADF task flow cannot be invoked from FSM and the following error message appears when you perform the task: OracleJSP error: java.io.FileNotFoundException

Limitations of Using requestScope() Method

Many of the recommended procedures for integrating ADF task flows used by setup tasks with FSM use Expression Language (EL), which in turn, use requestScope method. However, information retrieve using requestScope is available only in the scope of the current request. Therefore, you can use the information on the landing task flow when coming from FSM, but it will not be available to be used on other task flows.

If you intend to use the information passed by requestScope() on other task flows, retrieve the values using recommended EL and then store them in your own pageFlowScope variables.

Example

1. Invoke the following bean method on the landing page:

   ```
   SetupDynamicFlowBean setupDynamicFlowBean =
   (SetupDynamicFlowBean)

   this.evaluateEL(#{requestScope.setupDynamicFlowBean});
   ```

2. Store setupDynamicFlowBean object into the pageFlowScope using the same name as setupDynamicFlowBean
3. Access the information on other pages of the task flow by using one of the following:

   ```
   EL #{pageFlowScope.setupDynamicFlowBean}

   #{pageFlowScope.setupDynamicFlowBean.setupParametersBean
   .source}
   ```

Determining if an ADF Task Flow is Invoked from FSM

The following EL can be used to determine whether an ADF task flow is called from FSM:

```
#{requestScope.setupDynamicFlowBean.setupParameterBean.sour
ce}
```

If you call it from FSM, the returned value will be FS.

Note

This EL uses requestScope method. See Limitations of requestScope Method for more information.

Returning Control to FSM Task List

If an ADF task flow for a setup task is invoked from FSM, then the control must return to FSM task list once the ADF task flow is closed. Use the following steps to include this capability in your ADF task flow.

1. Determine if the ADF task flow was invoked from FSM.
2. Identify the controls (buttons, action menu items, and so on) that close the ADF task flow used by a setup task.
3. Add the following APIs as part of your code for the control:

```
SetupDynamicFlowBean setupBean =
SetupDynamicFlowBean.getInstance();

setupBean.returnControlToSetup();
```

Sample Code

```
SetupDynamicFlowBean setupDynamicFlowBean =
      SetupDynamicFlowBean.getInstance();

      String fsSource = null;

      if (setupDynamicFlowBean != null){
        fsSource  =
      setupDynamicFlowBean.getSetupParameterBean().getSource();

           if (fsSource == null || !FS.equals(fsSource)){
               return;
           }
           setupDynamicFlowBean.returnControlToSetup();
      }
```

Note

These APIs are available to you through FSM Shared Library. See Adding the FSM Shared Library for more information.

Reading Parameters Passed by FSM

Use the following steps to read parameters passed by FSM

1. Define input parameter names in your ADF task flows matching exact names of the task parameters specified in FSM.

2. Define the values of these input parameters according to ADF task flow standards.

Example

- Assume parameters for a setup task, T1, are defined as name1=value1&name2=value2
- In ADF task flow for T1, define two input parameters called name1 and name2
- Use #{pageFlowScope.name1} and #{pageFlowScope.name2} to read the values of name1 and name2 as passed by FSM

Note

When UI is invoked in full screen view, FSM passes all parameters – defined implicitly or explicitly through the URL. See Task Details – Entering Basic Information for more information on standard and full screen views of an UI.

Special Cases

In each of the following cases, reading the parameter values requires the same steps. However, each case passed the name-value pairs in a specific format.

- **Task Name**: The parameter is always passed and does not require to be explicitly defined on the tasks. The format used is: pageTitle=<task name>. For example, For Manage Tax Regime task the parameter will be, pageTitle=Manage Tax Regime.
- **ID of Task in an Implementation Project:** This is also a default parameter which is always passed by FSM without requiring task designers and developers to explicitly define it. The name-value pair used is: iPTLItemKey=<taskListItemID>
- **ID of Parent Task List in an Implementation Project:** Another default parameter passed by FSM without requiring task designers and developers to explicitly define it. The name-value pair used is: iPTLParentItemKey=<taskListItemID of the immediate parent>
- **Predefined Scope Value:** A combination of three parameters are passed using the following format: BOShortName=<value>&BOAttributeCode=<value> &BOAttributeValue=<value>

  The values are as follows:

- BOShortName is the short name (code) or code of scope (business object)
- BOAttributeCode is the short name (code) of the scope value
- BOAttributeValue is the actual scope value

- **Feature Selection:** Referred generically as feature, this functionality applies to offerings, options, and features and works the same way for all of them.

  Although the task parameter is defined as FS_FEATURE_SHORT_NAME=<Feature Short Name>, when the task is performed, FSM passes name-value pair as: <Feature Short Name>=<Selected Feature Choice Short Name>

  If more than one value is selected for the given feature, then multiple feature choice short names (codes) are passed using comma (,) as separator.

  For example, if FS_FEATURE_SHORT_NAME= ZX_REGIME_REG_STATUS is defined as task parameter, then ZX_REGIME_REG_STATUS = ZX_REGIME_REG_STATUS_YES will be passed by FSM where ZX_REGIME_REG_STATUS_YES is the actual short name (code) of the selected choice of ZX_REGIME_REG_STATUS.

- **Parent Feature Selection:** Feature is referred as a generic term for offering, option and features in this case as well and the functionality applies to all of them in exactly the same way.

  Although the task parameter is defined as FS_PARENT_FEATURE=<Parent Feature Short Name>&FS_FEATURE_LEVEL=<1, 2, 3, and so on>, when you perform the task FSM converts the parameter to:

  FS_PARENT_FEATURE=~@<Selected Child Feature Short Name>@<Selected Child Feature Choice Short Name>

  For example, if FS_PARENT_FEATURE=PER_PAYROLL&FS_FEATURE_LEVEL=1 is specified as task parameter, then FSM converts the parameter as follows when you perform the task:

  FS_PARENT_FEATURE=~@PER_PAYROLL_US@PER_PAYROLL_US_YES~PER_PAYROLL_UK@PER_PAYROLL_UK_NO

  Here, PER_PAYROLL_US and PER_PAYROLL_UK are first level children of PER_PAYROLL while PER_PAYROLL_US_YES is the selected choice of PER_PAYROLL_US and PER_PAYROLL_UK_NO is the selected choices of PER_PAYROLL_UK

If the level specified in the parameter was two, then the children of PER_PAYROLL_US and PER_PAYROLL_UK would have been included as well.

Note

The characters ~ and @ at the beginning of the returned value of FS_PARENT_FEATURE are important to notice. These two characters are used as separators for feature short name (code) and feature choice short name (code). Application developers must be careful in parsing the string to get the right information on feature and its selected choices.

Sample Code

This is an example of how name-value pairs passed for FS_PARENT_FEATURE can be parsed.

```
private void printFeatureChoices(String parameters) {
 System.out.println(parameters: + parameters);
 if (parameters != null && parameters.length() > 2) {
    String ampSep = parameters.substring(0, 1);
    System.out.println(ampSep: + ampSep);
    String eqSep = parameters.substring(1, 2);
    System.out.println(eqSep: + eqSep);
    String params = parameters.substring(2);
    System.out.println(params: + params);
    StringTokenizer st = new StringTokenizer(params, ampSep);
    String featureShortName = null, featureChoiceShortName =
      null;
    while (st.hasMoreTokens()) {
      String param = st.nextToken();
      int eqPos = param.indexOf(eqSep);
          if (eqPos >= 0) {
          featureShortName = param.substring(0, eqPos);
          featureChoiceShortName = param.substring(eqPos + 1);
          System.out.println(featureShortName: +
      featureShortName+ , featureChoiceShortName:  +
      featureChoiceShortName);
          }
    }
 }
}
```

See also:

Design Considerations – Parameter Enabled

Making Scope Selection Enabled

There are three types of ADF task flows that are impacted by FSM Scope. Each type requires certain steps to make it scope enabled.

Search Tasks

These ADF task flows are used by tasks designated as the Search task of an FSM scope.

1. Determine if the ADF task flow was invoked from FSM.
2. Make sure that the ADF task flow is invoked in MODE=SEARCH.
3. Read scope value passed from FSM.
    - This is optional and is only needed if the Search task is intended to search based on a qualifying attribute such as a parent scope value. The task designer needs to provide that information to the developer.
4. Append the scope value to the title of the page.
    - For example, if the scope value selected is US Excise Tax when you perform Manage Tax Statuses, then the page title should be Manage Tax Statuses: US Excise Tax.
5. Disable all buttons except the Done button.
6. Allow selection of the scope value.
7. Return the selected value to FSM.

Manage Tasks

These ADF task flows are used by tasks designated as the Manage task of an FSM Scope. Typically, they are for creating new instance of scope.

1. Determine if the ADF task flow was invoked from FSM.
2. Make sure that the ADF task flow is invoked in Manage mode.
3. Read the scope value passed from FSM.
    - This is optional and is only needed if the Manage task is intended to create a new value based on a qualifying attribute such as a parent scope value. The task designer needs to provide that information to the developer.
4. Append the scope value to the title of the page.
5. Allow the creation of new values.

Target (Consumer) Tasks

These ADF task flows are used by tasks that belong to a task list with Scope.

1. Determine if the ADF task flow was invoked from FSM.
2. Read scope value passed from FSM.
3. Return to FSM when task flow is closed.

Scope Enablement Related Functions

The following functions are used by one or more types of tasks that require scope enablement.

Determining Invocation Method

To determine which mode is used in FSM to invoke your ADF task flow, check the following parameter:

```
#{requestScope.setupDynamicFlowBean.setupParameterBean.mode
}
```

- If MODE = SEARCH, then Search task of a FSM scope has invoked this ADF task flow
- If MODE = MANAGE, then Manage task of a FSM scope has invoked this ADF task flow
- If MODE = NULL, then this ADF task flow is not invoked by scope selection process in FSM

Note

This EL uses requestScope method. See Limitations of Using requestScope Method for more information.

Reading Scope Values Passed By FSM

There are two ways to derive scope values passed by FSM.

Using EL Expression

Retrieve a combination of short name (code) of the scope business object and the primary key attribute code of the scope value by using the following expression. Note that it is one string but comma separated.

```
#{requestScope.setupDynamicFlowBean.scopeInfoParam['boShort
Name,attributeCode']}
```

For example, the following expression is used to retrieve scope value of Tax Regime (Scope business object) passed by FSM.

```
#{requestScope.setupDynamicFlowBean.scopeInfoParam['ZX_TAX_
REGIME,CN-TaxAttribute1']}
```

If the returned value is null or an empty string, then no match was found.

Note

This EL uses requestScope method. See Limitations of Using requestScope Method for more information.

Using HashMap

The following sample code illustrates how to read scope values from a HashMap.

```
SetupDynamicFlowBean setupDynamicFlowBean =
      SetupDynamicFlowBean.getInstance();
 String fsSource =
      setupDynamicFlowBean.getSetupParameterBean().getSource();
 if (fsSource == null || !FS.equals(fsSource)){
    return;
 }
HashMap fsScopeInfoMap = null;
fsScopeInfoMap = setupDynamicFlowBean.getScopeInfoMap();

if (fsScopeInfoMap == null){
 return;
}
else if (fsScopeInfoMap.keySet() == null){
 return;
}
Iterator iter = fsScopeInfoMap.keySet().iterator();
 if (iter == null){
    return;
 }
String boShortName = ;
String attributeCode=;
String scopeValueCode=;

while (iter.hasNext()){
 boShortName = (String) iter.next();
 System.out.println(boShortName= + boShortName);

 ArrayList scopeValueParamsArray =
      (ArrayList)fsScopeInfoMap.get(boShortName);

 int size = scopeValueParamsArray.size();

 if (size == 0){
    continue;
 }
```

```
 for (int i = 0; i < size; i++){
    ScopeValueParams scopeValueParams =
      ScopeValueParams)scopeValueParamsArray.get(i);

    if (scopeValueParams == null){
      continue;
    }
    attributeCode = scopeValueParams.getAttributeCode();
    scopeValueCode = scopeValueParams.getScopeValueCode();
    System.out.println(attributeCode= + attributeCode +
      scopeValueCode= + ScopeValueCode);
 }
}
```

Tip

EL expression is simpler to use. Use it if you need to retrieve only one scope value. If you expect to retrieve more than one scope value, use HashMap.

Returning Scope Values to FSM

The following API, available through the FSM Shared Library, can be used to return a value selected on the ADF task flow for the Search task of a scope to FSM:

```
setupDynamicFlowBean.setSelectedScopeInfo (String
selectedAttributeCode, String selectedScopeValueCode,
String selectedScopeValue);
```

It uses three parameters and their values can be up to 180 characters.

- **selectedAttributeCode:** Scope business object attribute name.
- **scopeValueCode:** Internal identifier of the scope value, which can be an alternate key or system-generated surrogate key.
- **selectedScopeValue:** Scope value name in the language of the login user performing the selection of the scope.

Note

Due to ADF limitations only one scope value can be returned to FSM at a time.

Sample Code

```
public void rowSelectedAction(SelectionEvent selectionEvent) {
        // Add event code here...

      ADFUtil.invokeEL(#{bindings.TaxRegimeVO.collectionModel.makeC
```

```
      urrent}, new Class[]{SelectionEvent.class}, new
      Object[]{selectionEvent});
//Get Scope values
       String taxRegimeCode = (String)
      ADFUtil.evaluateEL(#{bindings.TaxRegimeCode.inputValue});
        String taxRegimeName = (String)
      ADFUtil.evaluateEL(#{bindings.TaxRegimeName.inputValue});
//System.out.println(taxRegimeCode=+taxRegimeCode +   taxRegimeName
      = +taxRegimeName);
        SetupDynamicFlowBean setupDynamicFlowBean =
      SetupDynamicFlowBean.getInstance();
//Pass Scope values to ASM
       //The following method can be called several times if the
      selected rows contains
       //composite key for a scope value
       //setupDynamicFlowBean.setSelectedScopeInfo (String
      selectedAttributeCode, String selectedScopeValueCode, String
      selectedScopeValue);
        setupDynamicFlowBean.setSelectedScopeInfo(RegimeCode,
      taxRegimeCode, taxRegimeName );
```

See also:

Design Considerations – Scope Selection Enabled

Save and Go to Task Enabled

FSM provides a declarative component to add the Save and Go To Task option to the Save and Close button of an ADF task flow. The Save and Close action on ADF task flows of any of the tasks in the Save and Go To Task sub menu will return to the original task list in FSM from where the first task was invoked.

Add the declarative component as follows.

1. Create an FND application panel with Save and Close button enabled
2. Select FunctionalSetup from Component Palette
3. Drag and drop SetupButtonMenu from Component Palette into Save and Close commandToolbarButton of the fnd applicationsPanel
4. Bind a backing bean method to the ActionListener of SetupButtonMenu
5. The backing bean method will save and commit the current transaction and automatically invoke the same API as described in Returning Control to FSM Task List to return to FSM.

See also:

Design Considerations – Save and Go To Task Enabled

Note

The declarative component and APIs needed to implement this features are available through FSM shared library. See Adding the FSM Shared Library for more information.

Although this declarative component can be added to any buttons, the additional option shown on the button menu will always be called Save and Go To Task, irrespective of what the main button is called.

Enabling Security for Setup ADF Task flows

Use the following steps to properly secure ADF task flows used by setup tasks.

1. Add FSM Shared Library if not already done.
2. Secure ADF task flow with proper privilege as per Oracle Fusion Security standards and guidelines.
    - Ensure this privilege rolls up to the appropriate job roles that will be assigned to the users who will perform the setup task.
    - Ensure those job roles inherit ASM_FUNCTIONAL_SETUPS_ABSTRACT role. Without ASM_FUNCTIONAL_SETUPS_ABSTRACT role users will not have access to FSM and will not be able to perform setup tasks from FSM.

Warning

Understanding standards and guidelines of Oracle Fusion Security model and how to secure portlets are beyond the scope of this document. If you are not familiar with them, consult appropriate documentation and training material.

Developing Web Services to Execute Setup Tasks

While most setup tasks will be performed using ADF task flows (UIs), a setup task may also be executed by using web services. Typically, these tasks will launch a background process to manage setup data.

If you are using a web service to execute setup tasks, FSM requires that your web service includes an API called LAUNCHPROCESS() using the following specification. The rest of the design and technical details are up to the discretion of the web service designer and developer.

When the web service is invoking a background process to execute a task, it must also update the task status in FSM after the background process completes. FSM provides public web services to set task status, which can be used for this purpose.

launchProcess() Details

Java Specification

Interface.java

```
public interface LaunchProcessInterface {
public List<PropertyValue> launchProcess(List<PropertyValue>
      processControl);
}
```

PropertyValue.java

```
public class PropertyValue implements Serializable{
 String propertyName = null;
 String propertyValue = null;
 public PropertyValue() {
    super();
 }

 public void setPropertyName(String propertyName) {
    this.propertyName = propertyName;
 }

 public String getPropertyName() {
    return propertyName;
 }

 public void setPropertyValue(String propertyValue) {
    this.propertyValue = propertyValue;
 }

 public String getPropertyValue() {
    return propertyValue;
 }
}
```

XSD Specification

```
<xs:schema version=1.0 targetNamespace=…>
 <xs:element name=launchProcess type=tns:launchProcess/>
 <xs:element name=launchProcessResponse
      type=tns:launchProcessResponse/>
 <xs:complexType name=launchProcess>
```

```
      <xs:sequence>
        <xs:element name=arg0 type=tns:propertyValue minOccurs=0
         maxOccurs=unbounded/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name=propertyValue>
      <xs:sequence>
        <xs:element name=propertyName type=xs:string
         minOccurs=0/>
        <xs:element name=propertyValue type=xs:string
         minOccurs=0/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name=launchProcessResponse>
      <xs:sequence>
        <xs:element name=return type=tns:propertyValue minOccurs=0
         maxOccurs=unbounded/>
    </xs:sequence>
    </xs:complexType>
  </xs:schema>
```

Properties Passed by FSM

Property Name	Property Value
iPTLItemKey	Unique identifier of the task in a given task list in an implementation project
iPTLParentItemKey	Unique identifier of the immediate parent of the task in an implementation project. If the task is a direct child of an implementation project, then the ID of the implementation project.
FS_SOURCE	FS
FS_SCOPEINFOS	Selected scope value
FS_MODE	Search, Manage, or null See Making Scope Selection Enabled for more details.
Any explicitly defined Task Parameters	Specified values of the explicitly defined parameters. See Task Parameters for more details.

Properties Returned to FSM

Property Name	Property Value
processId	If the service submits a background process to execute the task, then process (job) ID of the submitted process should be returned. On the other hand, if the task execution process is completed as part of this service, then no value is returned (meaning the process ID may be empty).
taskStatus	Status of the task executing the service. If a background process is submitted, then the status returned must be Submitted (along with a process ID as specified). Otherwise, one of the valid statuses is returned.

statusDetails	Use this property to return errors if desired. FSM will show these errors in log files.

Task Status Description

Status Code	UI Equivalent	Description
NOT_STARTED	Not Started	User has not started performing the task
IN_PROGRESS	In Progress	User has started performing the task but the activity has not completed yet
DONE	Completed	User has completed performing the task
SUBMITTED	Submitted	A request has been submitted to a background process to perform the task.
COMPLETED_WITH_WARNINGS	Completed with Warning	Task has been performed but completed with Warnings.
COMPLETED_WITH_ERRORS	Completed with Error	Task has been performed but completed with Errors
EXECUTE_ONLY_ONCE	Execution Frozen	The task cannot be performed anymore

Tip

Use the Submitted status only if a background process has been submitted to execute a task. When a task is in Submitted mode, you cannot perform the task from FSM until the status is set to another value.

On the other hand, if status of a task is not set to Submitted when a background process has been submitted to perform the task, you may perform the task multiple times. This will result in submitting multiple background processes for the same task overlapping each other.

Coding Recommendation for Including launchProcess()

While it is ultimately the decision of the web service designer and developer on how the Service to execute a setup task is built, here are some guidelines that may be considered.

ADF BC Service Approach

These are the suggested steps if you are using ADF BC services.

1. Create a programmatic View Object (VO) called PropertyValueVO

2. Create two transient attributes of type String: PropertyName and PropertyValue
3. Create an Application Module (AM)
4. Add the PropertyValueVO to the AM
    - The View Object instance must be called PropertyValue
5. Go to the Java tab in your AM and select the option to generate Application Module Class
6. Implement launchProcess() API

Sample Code

```
public List<PropertyValueVORowImpl> launchProcess(
 List<PropertyValueVORowImpl> processControl){

 // Get the ProcessControl properties
 String taskListItemId = null;
 String parentTaskListItemId = null;
 String propertyName = null
 String propertyValue = null;

 for (int i = 0; i < processControl.size(); i++)
    PropertyValueVORowImpl prop = processControl.get(i);
    if (iPTLItemKey.equals(prop.getPropertyName()))
      taskListItemId = prop.getPropertyValue();
    else if (iPTLParentItemKey.equals(prop.getPropertyName()))
      parentTaskListItemId = prop.getPropertyValue();
    else{
      propertyName =  prop.getPropertyName();
      propertyValue = prop.getPropertyValue();
    }
 }

 /*****************************************/
 /*   Write Code to Submit Process               */
 /*****************************************/

 // Return the status + process Id + status Details
 String status = SUBMITTED;
 String statusDetails = null;
 String processId = 9999;

 PropertyValueVORowImpl statusProp =
      (PropertyValueVORowImpl)this.getPropertyValue().createRow();
 statusProp.setPropertyName(status);
 statusProp.setPropertyValue(status);

 PropertyValueVORowImpl statusDetailsProp =
      (PropertyValueVORowImpl)this.getPropertyValue().createRow();
 statusDetailsProp.setPropertyName(statusDetails);
 statusDetailsProp.setPropertyValue(statusDetails);
```

```
 PropertyValueVORowImpl processIdProp =
      (PropertyValueVORowImpl)this.getPropertyValue().createRow();
 processIdProp.setPropertyName(processId);
 processIdProp.setPropertyValue(processId);

 List<PropertyValueVORowImpl> processReturn = new
      ArrayList<PropertyValueVORowImpl>(2);
 processReturn.add(statusProp);
 processReturn.add(statusDetailsProp);
 processReturn.add(processIdProp);

 return processReturn;
}
```

Updating Task Status when Background Process Completes

If the web service for executing a setup task is submitting a background process to perform the task, then the service must also update the task status once the background process completes.

You can use FSM Public Service called setTaskStatus for this purpose. See Task Status Integration for more details.

Tip

Although FSM does not prevent the web service from setting the task status to any of the valid task statuses, when a background process completes the task status should be set to one of the following statuses: DONE, COMPLETED_WITH_WARNINGS, COMPLETED_WITH_ERRORS, and EXECUTE_ONLY_ONCE.

8

Web Services for Setup Export and Import

Design Considerations

Warning

Application developers building setup export and import web services should be knowledgeable and experienced in ADF, JDeveloper, and web service technologies and their standards and guidelines.

About Setup Export and Import Services

FSM allows you to export and import implementation projects to move setup data from one instance to another.

When an implementation project is selected for export and import process, FSM will identify business objects associated with each setup task in the implementation project. It will then use web services to read data from (at the source instance) and write data to (at the target instance) those business objects.

Therefore, you must provide setup export and import web service for each setup business object if their data is expected to be exported and imported using FSM. Each service must be atomic and self-sufficient, which means it must manage dependencies, handle surrogate keys, order processing, manage transaction, and so on.

Typically ADF Business Component (BC) View Objects are used to create these services; however, FSM also supports services based on non ADF BC objects.

If you are expecting larger volume of setup data to be migrated from one instance to another, then export and import of all of that data in a single batch may not be possible due to memory constraints and performance degradation. FSM recommends and supports batch processing for such cases, which allows invoking export and import services multiple times to process data in batches.

See also:

Developing ADF BC Object-Based Export and Import Web Services

Developing Non ADF BC Object-Based Export and Import Web Services

Developing Export and Import Web Services of Larger Data Volume

FSM Export and Import Service Methods

To facilitate export and import capabilities, the web services must include EXPORTDATA() and IMPORTDATA() methods provided by FSM. See ADF BC based export and import services and non ADF BC based services for more details.

Service Granularity

A service for each setup business object is desirable. However, a single service can be developed to potentially handle multiple objects at a time. It is up to the application developers to make their setup services as coarsely-grained or granular as needed by their business requirements.

For example, application developers may choose to provide two separate setup export and import services for business objects called Accounting Periods and Accounting Period Types or combine them into one service.

Entity Object (EO) Association Types

There are two common types of EO associations that have impact on setup export and import services.

Composite Associations

If a business object is implemented using more than one EOs that have tightly coupled relationships, which means that child entities cannot exist without the existence of the parent entity, then the relationships between the parent and the children are known as composite associations.

For example, the Purchase Order business object may be comprised of a header with one or more line items and each of the line items may have one or more shipments. The shipments cannot exist without a line item, and the line items cannot exist without a header. Therefore, Purchase Order Header - Purchase Order Lines - Shipments is a composite relationship. Purchase Order Lines and Shipments should be exposed in the Purchase Order Header SDO and not separately.

Defining composite associations for entity objects provides the following key behavior:

- Database changes are made in proper order guaranteeing parent entity is inserted before child entities.

- Child entities are protected from orphan rows. Deleting parents can be prevented if any children exist or cascade delete of parent can be implemented by deleting the children when a parent is deleted.

When defining composite associations, applicable properties can be set as follows:

- Cascade delete
- Cascade update of key attributes
- Locking of top-level container
- Update of top-level history columns

Non-Composite Associations

An entity may be referred in another entity to provide fuller context (foreign key references). These relations are referred to as non-composite associations.

For example, purchase orders may include relationships to suppliers to signify which PO is for which supplier.

Role of Alternate Keys

Since FSM export and import moves setup data between different instances, system-generated primary keys from the source instance cannot be used to determine the existence of a row in the target instance. All export Services must therefore exclude surrogate keys during export, and provide functionality to restore these keys during the import process using alternate keys.

Similarly, foreign keys also cannot be used between different instances. Therefore, alternate keys of the foreign key references must also be included in the export. The restoration of the surrogate foreign keys during the import process can then be accomplished by using the List of Value (LOV) feature in the View Object.

What to Consider When Defining Alternate Keys

Non-Translatable Attributes

A non-translatable, nonchangeable short name or code attribute is most preferable as an alternate key. A translatable attribute is not prevented from being part of an alternate key, however, may have the following consequences:

- If the translatable attribute value changes in between different import processes, the same row may be duplicated at the target instance since Setup import will not have any way to identify the link between the old alternate key and the newly changed alternate key.
- Setup export and import has dependency on language. As a result, if the languages of the source instance and the target instance are different, then alternate keys will not match and therefore, setup import may create duplicate rows at the target instance.

Global Unique Identifier as Alternate Key

The second best option to using user-defined, non-changeable attribute as alternate key is to use an auto-generated global unique identifier (GUID). GUID works best when data is always created at the source instance and moved to target instance but never created at the target instance.

For example, consider the case when tax rates are always created at the Test (source) instance and then migrated to Production (target) instance. In this case, any new tax rate created in Test will be assigned a GUID and conflicting GUID generation is avoided because no other instance creates tax rates. On the other hand, if tax rates are created in both Test and Production instances and moved back and forth between the two using export and import, then same tax rates may be created at both instances with two different GUIDs, preventing setup import from being able to detect them as the same data.

Foreign Key as Part of Alternate Key

In case of non-composite associations, a foreign key can be part of an alternate key of an associated object. In such cases, alternate key for the foreign key ID is used first to completely resolve the alternate key of the associated object and then it is used to resolve the primary key at the target instance.

For example, if the alternate key for Tax Regime is NAME+ORGID, then ORGID will be used first and then NAME+ORG CODE as the alternate key to resolve the surrogate primary key of Tax Regime at the target instance.

Alternate Key for Composite Associations

The parent object in the composite association will require an alternate key following the same rules as specified. However, if a child object in a composite association does not contain a surrogate primary key, then there is no need to define an alternate key on the child. In these cases, the primary key of the child object is typically defined as the primary key of

the parent object combined with an attribute of the child object. Since primary key of the parent will be resolved using its alternate key, FSM import will be able to use parent ID plus the child attribute to uniquely identify proper children.

For example, Purchase Order Header is the parent object and has PO_HEADER_ID as the surrogate primary key. Purchase Order Line Items, the child object, has its primary key defined as PO_HEADER_ID + PO_LINE_ITEM_NUM. Therefore, Purchase Order Line Items will not require alternate key because PO_HEADER_ID will be resolved by using the alternate key of Purchase Order Header object and then setup import will use this newly resolved PO_HEADER_ID + PO_LINE_ITEM_NUM to correctly match the children data.

Filtering Exported Setup Data

When a business object is defined as a scope of a task list or setup data export, then you will have the option to filter setup data by scope values during setup export. See Business Objects as Scope for more information.

The EXPORTDATA() API is able to accept a list of filter (scope) values that can be used to filter the view objects as well as propagate filtering to related view objects if needed.

Excluding Oracle Defined (Seeded) Data

Oracle-defined data, known as seed data, cannot be moved between two instances by using FSM export and import processes. Oracle seed data must be managed at different instances using the standard Oracle Fusion patching process.

Multi-Language Support

The export and import APIs support retrieval and upload of all translations along with the base data. Application developers, however, must add View Objects for their translation tables and create View Links from their base view objects to their translation view objects.

Developing ADF BC Object-Based Export and Import Services

Warning

You should be experienced in developing applications using Oracle platforms and knowledgeable in ADF, JDeveloper, and web service technologies. You should also be familiar with Oracle Fusion Applications standard and guidelines.

In most typical cases, setup export and import services will be created based on ADF BC objects. To develop these services the following steps should be used.

1. Create ADF BC objects
2. Implement FSM export and import methods
3. Generate web services

Note

You should review Design Considerations to understand various design implications before you start developing export and import web services, especially if you are doing so for the first time.

Creating ADF BC Objects

Before web services for setup export and import can be created, appropriate ADF BC objects needed by those services must be created.

Entity Object (EO) Requirements

1. Define entity associations and identify whether or not they are composite associations. See EO Association Types for more information.
2. Create alternate keys for Entity Objects (EO).
    - Since setup export and import moves setup data across different instances, system generated primary keys cannot be used as identifiers. An entity constraint must therefore be created for each setup related EO to generate unique alternate keys. See Role of Alternate Keys for more information.
3. Enable bulk processing.

- Go to the tuning panel of the EO wizard, and check Use Update Batching option, which will enable to use bulk DML to upload data into database.
- Specify an appropriate value for When Number of Entities to Modify Exceeds parameter. Profiling may be needed to determine the best value.
- Specify validation logic in EO using declarative validators, which can be optimized for batch processing. Avoid programmatic validation logic.
- Avoid the following cases as they disable bulk DML:
    - EO with CLOB-BLOB
    - EO with attributes marked as Retrieve-on-Insert or Retrieve-on-Update
    - EO with ROWID attribute
    - Custom coding that overrides doDML method

View Object (VO) Requirements

1. Add alternate key to View Object (VO).
    - In the General tab of the VO, go to the Alternate Keys section and add an alternate key.
    - In the Alternate Key popup, select one or more Entity Usages and select the alternate key from the corresponding usage.
2. Define alternate keys for any foreign keys. See Role of Alternate Keys for more information.
    - Include alternate key of the object referenced by the foreign key.
    - Define a list of values (LOV) for the alternate key of the referenced object.
    - Make sure the foreign key is updated by the LOV.
3. Create View Links for master-detail relationships, if applicable.
    - If master-detail relationships exist between two objects, then View Links can be set up to automatically export detail object when the master object is exported.
4. Enable for multi-language support (MLS) by creating VOs for translation EOs and then creating view links between the VOs for the base EO and the corresponding translation EO.

- When setup data is moved from one instance into another, corresponding translated version must also be moved.

5. Enable bulk processing.
    - Go to the tuning panel of the VO and specify correct fetch size. Profiling may be needed to determine the best value.

Exception Handling Requirements

If any errors occur during setup export and import, they must be reported to the user. The error scenarios may include but not limited to: data validation, primary key constraint violation, unique key constraint violation, invalid foreign keys, business rule violation, and so on.

FSM will delegate all exception handling to the underlying ADF BC objects and their methods. Application developers, therefore, must implement all validation logic and business rules in BC4J Layer. They must also define appropriate error messages either declaratively or programmatically by throwing appropriate JboExceptions. The error messages should identify the invalid rows.

Implementing Setup Export and Import APIs

Once the ADF BC objects are properly defined, application developers will first prepare service application module and then implement exportData() and importData() methods provided by FSM in the Application Module class.

Preparing Service Application Module (AM)

Before implementing setup export and import APIs, prepare service Application Module (AM) using the following steps.

1. Include the Functional Setup Model library in your project
    - Create your AM
    - Generate Application Module class
    - Go to the Java tab in your AM and select the option to generate the Application Module class
    - Implement oracle.apps.setupHub.remoteApp.publicModel.migrator.Migrator interface from your AMImpl
2. Add the following Setup VOs from oracle.apps.setupHub.remoteApp.publicModel.view to your AM:

- ExportCriteriaVO - Usage: ExportCriteria
- ExportControlVO - Usage: ExportControl
- ExportReturnVO - Usage: ExportReturn
- ImportDataVO - Usage: ImportData()
- ImportControlVO - Usage: ImportControl
- ImportReturnVO - Usage: ImportReturn

3. Also add your VOs that you want to be exported and imported via this Service to the AM. Make sure to include in the AM the children VOs in their respective parent VOs.

Implementing Export API

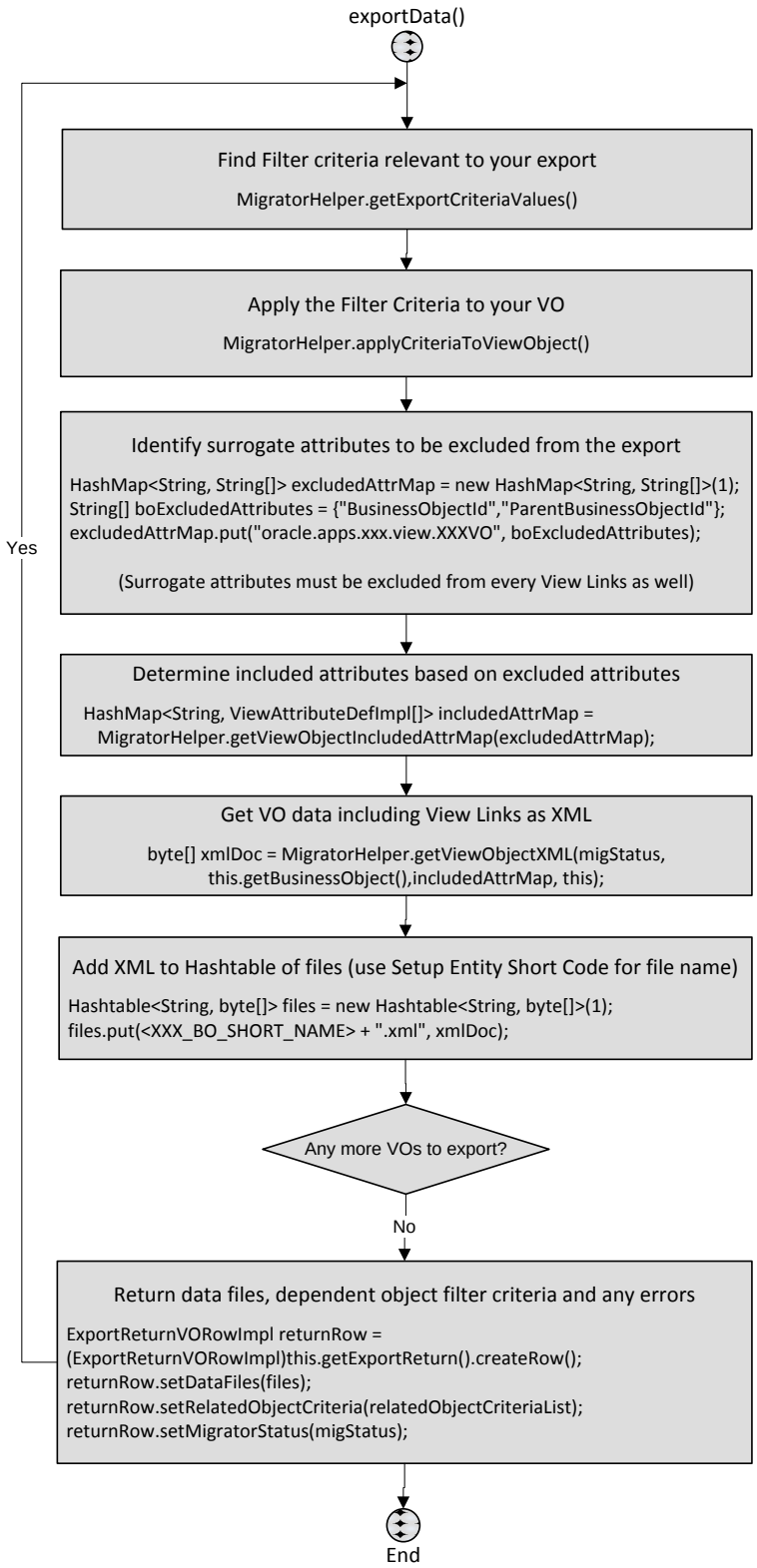
A method called exportData() must be implemented in Application Module class as follows. See Export Algorithm at a Glance for an overview, an example of an export API, and how to generate report on exported metadata.

1. Implement exportData() method
2. Identify filter criteria
3. Apply filter criteria to the VOs that are applicable to export and import of Setup data
4. Identify surrogate keys
5. Write VO data to XML file
6. Maintain a Hashtable for all XML files generated by export
7. Return collection of XML files from the Hashtable to FSM
8. Return export status and if there are any errors

Note

Use standards and guidelines provided in Oracle ADF API reference manuals.

Export Algorithm at a Glance



Implementing exportData()

Use exportData() method as follows:

```
public ExportReturnVORowImpl exportData(
              List<ExportCriteriaVORowImpl> exportCriteria,
              ExportControlVORowImpl exportControl)
```

View Object	Method	Description
ExportCriteriaVORowImpl	getObjectKey()	Gets CODE of setup business object
	getAttributeSet()	Gets a set of attributes and values
	getAttributeName()	Gets name of the attribute being used for filtering a setup business object
	getAttributeValue()	Gets the value of the filter attribute
ExportControlVORowImpl	getExportMetadata()	Used by FSM only
	getExportData()	Used by FSM only
ExportReturnVORowImpl	setDataFiles()	Returns a list of XML files created by export
	setServiceVersion()	Returns version of the Service API used in export. If a Service API changes over time, this information may help developers to take appropriate actions based on versioning if applicable.
	setMigratorStatus()	Returns errors if any occurred during export

Identifying Filter Criteria

Setup Export Filter Parameters

FSM will pass filter parameters to the export services when configuration packages are created if any of the following conditions are met.

Business Object Has Search Task Defined

When a business object with Search task is exported, when generating the configuration package for setup export, you have the option to select specific values of the business object, which in turn are passed as filter parameters. See Business Object Details for more information of Search tasks of business objects.

Task List Scopes Are Defined

Any scope value selected when executing the setup tasks are passed as filter parameters unless you add or remove those selected scope values. If

changes are made, then the newly selected scope values are passed as filter parameters. On the other hand, if a scope is defined as allow scope selection only during export and import, then no scope values are selected when tasks are performed, however, when performing setup export, you can select scope values to be used as filtering exported data. See Task List Scope for more information.

Task Parameters Are Defined

FSM will pass any and all parameters defined on a task to the corresponding export service whether or not the parameter is applicable for filtering data of associated business object. The export service may choose to ignore the parameters that are not applicable to filter exported data. Parameters defined for task lists, however, will not be passed to the export service. See Task Parameters for more information.

Business Object Is Defined As Deployment Method

When business objects of type Deployment Method are exported, the attribute values defined for the Deployment Method are passed as parameters.

Note

Although FSM will pass filter parameters to the setup export services, it is up to the service to use those parameters as necessary. If the parameters are not utilized by the services to filter data then all data of that business object will be exported and imported.

Finding Setup Export Filter Parameters

The filter parameters for setup export are available through exportCriteria and are a list of business object values. For example:

Object Key	Attribute Set	Attribute Name	Attribute Value
HR_LOCATIONS	Set1	LocationId	101
FUN_BUSINESS_UNIT	Set2	BuId	202
FUN_BUSINESS_UNIT	Set2	LocId	1000

The object key represents the short name (or code) of the business object. The export API should match the object keys with the objects it will export and then match the attribute values. If matched, then attribute value can be applied to the VOs as filter criteria. The attribute set is useful when multiple attributes are used to specify a given filter condition.

For example, if the filter condition needed is Location Name = ABC and Country = US, then they can be grouped together in a set to specify a single filter criteria. If the set is blank, then all attributes and their values are considered as part of a default set.

The following helper method returns a HashMap of the attribute sets and their corresponding filter criteria for a given business object. Subsequently, it can be used to create ViewCriteria for the related VO and with applyCriteriaToViewObject API.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static HashMap<String, List> getExportCriteriaValues(
    List<ExportCriteriaVORowImpl> exportCriteria,
    String objectKey)
```

Sample Code

```
HashMap<String, List> attributeCriteriaValues =
  MigratorHelper.getExportCriteriaValues(exportCriteria,
      <XXX_BO_SHORT_NAME>);

Iterator iter = attributeCriteriaValues.keySet().iterator();
while (iter.hasNext()) {
  // Key = attribute set
  String key = (String) iter.next();
  List list = attributeCriteriaValues.get(key);
  // Export Criteria values for each attribute set
  for (int i = 0; i < list.size(); i++) {
     ExportCriteriaVORowImpl criteriaRow =
(ExportCriteriaVORowImpl)list.get(i);
     System.out.println(criteriaRow.getAttributeName());
            System.out.println(criteriaRow.getAttributeValue());
  }
}
```

Additional Helper Methods

The following helper methods are provided to achieve additional actions that may be needed to properly identify filter criteria passed to the services by FSM.

- **Transforming Attribute Names:** Use the following helper method to transform the attribute names passed as filter parameter if they are different from the corresponding attribute names of the VOs.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static void transformExportCriteriaAttributeNames(
List< ExportCriteriaVORowImpl> exportCriteria,
String objectKey, HashMap<String, String> attributeNameChangeMap)
```

For example, OrgId can be changed to BuId by passing a HashMap as shown to the API:

```
HashMap<String, String> attributeNameChangeMap=new HashMap<String,
      String>(1);
attributeNameChangeMap.put(OrgId, BuId);
```

- **Checking For Valid Filter Criteria:** This helper method may be relevant if the Service requires taking specific action if no valid filter criteria are available.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static boolean
      isValidCriteriaAttributeAvailable(ViewObjectImpl vo,
    HashMap<String, List> attributeCriteriaValues);
```

- **Collecting Filter Criteria Details:** This helper method returns a HashMap of attribute sets, attributes and attribute values, which can be used to apply where and how they are needed.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static HashMap<String, HashMap<String, List>>
getAttributeCriteriaValuesBySet (
      List<ExportCriteriaVORowImpl> exportCriteria,
    String objectKey)
```

Tip

Use the same attribute names that will be returned as filter parameters for your service VOs. If different attribute names are used in the service VOs, then the service must transform the attribute names properly so that the names match and therefore, the filters are appropriately identified.

Applying Filter Criteria to VOs

Once the filter criteria passed by FSM are identified, they can be applied to VOs to filter a subset of the data in the VOs. The following helper method creates a dynamic ViewCriteria on a VO using the HashMap of attributes for filter criteria and their values.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static void applyCriteriaToViewObject(ViewObject vo,
HashMap<String, List> attributeCriteriaValues)
```

Special Case: VO Contains Both Oracle-Defined and User-Defined Data

Seeded data, which is data defined and shipped by Oracle, is always excluded from setup export and import process and must only be maintained via standard Oracle patching process. However, if setup data is exported for reporting purposes, then the export may need to extract both seeded and unseeded data. In such cases, use the following helper method to include the seeded data in the export process.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static void applyCriteriaToViewObject(ViewObject vo,
HashMap<String, List> attributeCriteriaValues, Boolean
      includeSeedData)
```

Sample Code

```
MigratorHelper.applyCriteriaToViewObject(this.get<XXXVO>(),
exportCriteriaValues, exportControl.getIncludeSeedData());
```

Note

exportControl.getIncludeSeedData() must be passed as a parameter to the method and not True-False to include seeded data in export.

The columns CREATEDBY and LASTUPDATEDBY must be included in the VOs that contain seeded data since LASTUPDATEDBY= SEED_DATA_FROM_APPLICATION identifies the seeded data rows.

Adding Additional View Criteria to Filter Data

The following helper method can be used if there is a need to apply additional criteria to VOs to filter data.

```
ViewCriteria vc1 =
    this.get<XXXVO>().getViewCriteria(YourCustomViewCriteriaName);
 // Get Criteria before applying
ViewCriteria vc2 =
      MigratorHelper.getViewCriteria(this.get<XXXVO>(),
exportCriteriaValues, exportControl.getIncludeSeedData());
vc1.copyFrom(vc2);
this.get<XXXVO>().applyViewCriteria(vc1);
```

Identifying Surrogate Keys

Since setup data is migrated from one instance to another, surrogate keys from the source instance is not applicable at the target instance. Therefore,

all surrogate keys must be identified and excluded from writing to the XML file during setup export.

Surrogate keys must be identified in the VOs as well as in the view links.

The following HashMap may be used to maintain a list of VOs and an array of excluded attributes for each of those VOs:

```
HashMap<String, String[]> excludedAttrMap
```

The following helper method may then be used to get a map of included attributes.

```
public static HashMap<String, ViewAttributeDefImpl[]>
      getViewObjectIncludedAttrMap(HashMap<String, String[]>
      voExcludedAttrMap)
```

The helper method is found in:

oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorHelper

Writing Setup Data to XML Files

Once appropriate filters are applied and the attributes for export are determined, setup data is ready be written to XML. The View Link objects are automatically exported and written to XML along with their parents unless they are explicitly excluded in voExcludedAttrMap.

Although optional, it is also recommended that you set forwardOnly mode to True by using MigratorControl to improve performance of your export. If the export will involve larger data volume then batch processing is better suited than forwardOnly mode. See Batch Processing of Large Data Volume for more information.

The following helper method provides ability to write to XML:

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static byte[] getViewObjectXML(MigratorStatus migStatus,
      ViewObject vo,   HashMap<String, ViewAttributeDefImpl[]>
      voAttrMap, boolean debugMode, MigratorControl migControl)

MigratorControl is set as follows:
MigratorControl migControl = new MigratorControl();
migControl.setForwardOnly(true);
```

Tip

During development phase, you can set debugMode to True to print VO XML and review correctness more easily. However, remember to set debugMode to False once verified.

Maintaining Hash Table for Export Related XML Files

A Hashtable should maintain all XML files related an export process and XML filename should be used as the key.

The format of the XML filename must be <SetupEntityShortName>.xml such as HR_LOCATIONS.xml, ASM_TASK.xml, and so on. This is imperative for import process to identify the right XML file for right setup data by using: Hashtable<String, byte[]> files.

Managing Multiple Business Objects in a Single Task or Service

If a single task or task flow manages data from multiple business objects, then the corresponding service can also manage export and import of those objects as a single unit.

This requires that the higher level export API follows one of the following two approaches as best suited:

- Invokes export APIs of the related business objects from the nested application modules
- Invokes export APIs at more granular VO level

All generated XMLs from multiple objects should then be added to file list so that all related data is exported as one set. The following helper method is provided to assist in appending files and statuses from nested export APIs to file and status of the higher level export API.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static void addAdditionalExport(Hashtable<String,
      byte[]>fromList, Hashtable<String, byte[]>toList,
      MigratorStatus fromStatus, MigratorStatus toStatus);
```

Note

If a setup task does not manage data for dependent objects, but just depends on other tasks and business objects, then handling those related objects in a single service is not recommended. Instead, let FSM use task list hierarchy of the implementation project to invoke related business objects services.

Returning XML File Collection Generated by Export Method

The collection of XML files generated must be returned to FSM as shown.

```
ExportReturnVORowImpl returnRow =
(ExportReturnVORowImpl)this.getExportReturn().createRow();
returnRow.setDataFiles(files);
```

Returning Errors

The following method can be used to return errors to FSM, if applicable.

```
MigratorStatus migStatus = new MigratorStatus();

// If processing error/exception occurs
migStatus.setProcessingError(xxx);
returnRow.setMigratorStatus(migStatus);
```

Reporting on Exported Metadata

To generate BIP reports on XML files created from ADF view objects, the following method can be used to pass the metadata as XSL-FO.

```
migStatus.getObjectReportMetadataMap().put(<xmlRootNodeName>,
      <reportMetadata>);
```

The following method will retrieve XSL-FO metadata from the view objects.

```
reportMetadata = getReportMetadata((ViewObjectImpl)vo);
```

Example of Export API

```
import
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper;
public ExportReturnVORowImpl exportData (
      List<ExportCriteriaVORowImpl> exportCriteria,
      ExportControlVORowImpl exportControl){

ExportReturnVORowImpl returnRow = (ExportReturnVORowImpl)
      this.getExportReturn().createRow();

MigratorStatus migStatus = new MigratorStatus();

 try {

   // Identify Filter Criteria
   HashMap<String, List> exportCriteriaValues =

      MigratorHelper.getExportCriteriaValues(exportCriteria,<XXX_BO
      _SHORT_NAME>);

   // Apply Filter Criteria to your business object VO
   MigratorHelper.applyCriteriaToViewObject(this.get<XXXVO>(),
                                            exportCriteriaValues);

   // Identify surrogate attributes to be excluded from the export
```

```
    HashMap<String, String[]> excludedAttrMap = new HashMap<String,
       String[]>(1);
    String[] boExcludedAttributes =
       {BusinessObjectId,ParentBusinessObjectId};
    excludedAttrMap.put(oracle.apps.xxx.view.XXXVO,
       boExcludedAttributes);

    // Build included attributes using excluded attributes
    HashMap<String, ViewAttributeDefImpl[]> includedAttrMap =
       MigratorHelper.getViewObjectIncludedAttrMap(excludedAttrMap);

    // Write View Object XML using included attributes
    byte[] xmlDoc = MigratorHelper.getViewObjectXML(migStatus,
       this.get<XXXVO>(),
                                          includedAttrMap, false);

    // Add XML to hashtable for XML files
    Hashtable<String, byte[]> files = new Hashtable<String,
       byte[]>(1);
    files.put(<XXX_BO_SHORT_NAME> + .xml, xmlDoc);

    // Return data files to FSM
    returnRow.setDataFiles(files);
  }
  catch (Exception e){
    migStatus.setProcessingError(e.getLocalizedMessage());
  }

   // Return any exceptions/errors to FSM
   returnRow.setMigratorStatus(migStatus);
   return returnRow;
}
```

Implementing Import API

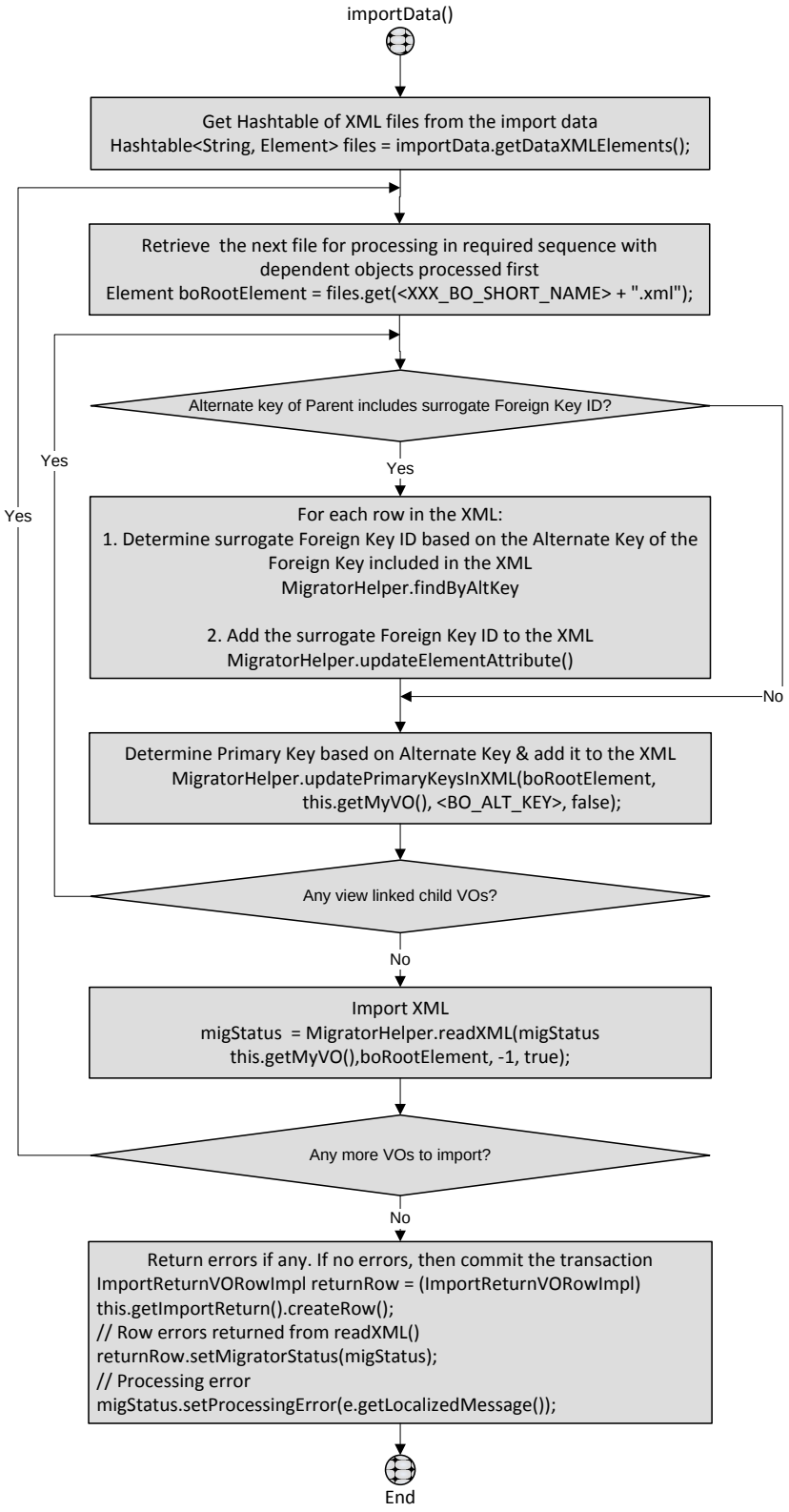
A method called importData() must be implemented in Application Module class as follows. See Import Algorithm at a Glance for an overview and an example of import API.

1. Implement importData() method
2. Identify XML data elements being passed
3. Update primary keys based on alternate keys
4. Read XML
5. Commit transaction
6. Return import status and if there are any errors

Note

Use standards and guidelines provided in Oracle ADF API reference manuals.

Import Algorithm at a Glance



Implementing importData()

Use importData() method as follows:

```
public ImportReturnVORowImpl importData (
        ImportDataVORowImpl importData,
        ImportControlVORowImpl importControl)
```

View Object	Method	Description
ImportDataVORowImpl	getDataFiles()	Gets a list of XML files to be processed during import
	getServiceVersion()	Returns version of the Service API used in export. If a Service API changes over time, this information may help developers to take appropriate actions based on versioning if applicable.
ImportControlVORowImpl	getImportMetadata()	Used by FSM only
	getImportData()	Used by FSM only
ImportReturnVORowImpl	setMigratorStatus()	Returns errors if any occurred during export

Note

Foreign keys that are included with an alternate key must be resolved within the import method. All other foreign key references should be resolved using view object LOV option.

For business objects with parent-child relationships, the associated child attribute is automatically updated with the parent primary key value when updatePrimaryKeysInXML() method is used on the parent view object.

Identifying XML Data Files

The exported data can be retrieved as separate XML elements using the following helper method.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
Hashtable<String, Element> files = importData.getDataXMLElements();
Element xmlRootElement = files.get(XXX_BO_SHORT_NAME + .xml);
```

Use following imports:

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

Updating Primary Keys in XML Data Elements

Since the primary keys of the business objects were excluded from XML during setup export, they must be included again before setup import is performed.

The following helper method will update the corresponding XML elements based on their alternate keys provided in their VOs.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static void updatePrimaryKeysInXML(Element rootElement,
                    ViewObjectImpl vo, String altKeyName, boolean
      skipWhere)
```

View Linked Objects

If there are view linked (which means dependent) business objects within the XML, then the helper method must be invoked for each of the view linked objects. The helper method will automatically update the associated child attribute in all children VOs that are directly view linked to the parent VO.

The skipWhere parameter used in ViewObjectImpl.findByAltKey() within the helper method can be set to true for linked VOs. If set as such, the alternate keys are searched using the entire row set.

Tip

For better performance, use VOs without any view links.

Foreign Key IDs as Part of Alternate Keys

If VOs that are not view linked, has alternate keys which include a surrogate foreign key ID, then those foreign key references must be resolved before calling updatePrimaryKeysInXML(). The following helper method will find the surrogate ID of the Foreign Key and appended it to the XML using the alternate key of the foreign key ID provided during export.

```
Helper methods in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static Row findByAltKey(ViewObjectImpl vo, String
      altKeyName, Object[] values)
public static void updateElementAttribute(Element element,  String
      attribute, String value)
```

Reading XML

The following helper method is used to import XML elements once all surrogate IDs are resolved.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static void readXML(MigratorStatus migStatus, ViewObjectImpl
      vo, Element xmlRootElement, int depthCount, boolean
      bundledExceptionMode, boolean debugMode);
```

- depthCount: This parameter represents how many levels of VIEW LINKS will be traversed during processing. For example:
    - 0 (zero) indicates no view links will be traversed
    - 1 (one) indicates the direct view links of the object will be traversed
    - 2 (two) indicates two successive levels of view links will be traversed, and so on…
    - (-1) (minus one) indicates all levels will be traversed
- **bundledExceptionMode:** When this flag is set to True, all attribute level validations are run. On the other hand, if this is set to False, validation is stopped after the first failed attribute level validation. ADF, however, will try all attribute validations no matter what.
- **debugMode:** Set it to True to print the XML being imported for easier debugging of code during development. Once verified, the flag should be set to False.

Managing Multiple Business Objects in a Service

FSM supports creating a single web service to export data from multiple business objects. If an export API is created to process multiple business objects, then the corresponding XMLs must also be imported via a single import API.

This means that the import API of the parent business object needs to support one of the following approaches and match whichever approach was followed by the corresponding export API:

- Invokes import APIs of the related Business objects from the nested application modules
- Invokes import APIs at more granular VO level

The following helper method appends the status from a nested import API to the import status of a parent.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static void addAdditionalImport(
MigratorStatus fromStatus, MigratorStatus toStatus);
```

Note

If a setup task does not manage data for dependent objects, but just depends on other tasks and business objects, then handling those related objects within a single service is not recommended. Instead, let FSM use task list hierarchy of the implementation project to invoke related business objects services.

Committing Transactions

Once the data is imported, the transactions can then be committed. When importing multiple business objects, the transaction may be committed after all objects have been processed.

```
this.getTransaction().commit();
```

The decisions on transaction management are left to the application developers based on their requirements. They may use the MigratorStatus.isErrorFound() to check if any errors occurred during import. In addition, isObjectErrorFound(ViewObjectImpl VO) is available to check errors encountered in a specific VO. The second method is particularly relevant when import API processes multiple VOs together.

In either case, it is up to the application developers to decide if they wish to commit a transaction to save a partially successful set of objects and rows.

Returning Errors

Any error encountered during import of an individual row is available within MigratorStatus that is returned by readXML method. Any other errors may be reported using the following method.

```
// If processing error/exception occurs
migStatus.setProcessingError(xxx);

// Return the status using ImportReturnVORowImpl
returnRow.setMigratorStatus(migStatus);
```

Returning Errors When Using Custom Programmatic Validations

If additional programmatic validations are used in import API and any rows failing those validations should be excluded from Migrator.readXML(), then return the errors to FSM using MigratorStatus.

```
// Get a list of your XML elements
NodeList xmlElements =

       parentRootElement.getElementsByTagName(<vo>.getXMLRowElemen
      tTag());

ArrayList<MigratorRowStatus> migRowStatusList = new
                   ArrayList<MigratorRowStatus>(1);

// Loop through each row in your XML, and capture errors
for (int i = 0; i < xmlElements.getLength(); i++) {
Element element = (Element)boElements.item(i);

// Add your validation code

// If row validation is successful, then continue

// If row validation fails, then remove the row element & capture
      error
 // Remove the row element from the XML
   parentRootElement.removeChild(element);

 // Capture information about the error
   StringWriter rowXML = new StringWriter();
   ((XMLNode)element).print(rowXML);

MigratorRowStatus migRowStatus = new
MigratorRowStatus( <rowXML>, // Row that caused the error
<errorMessage>, // Detailed Error message
<errorCode>, // Error Code, if any
<errorSeverity>, // Error Severity, if any
<alternateKey> // Use of alternate key is preferable instead of an
      Id, since it is visible to user
));

   migRowStatusList.add(migRowStatus);
}

// Invoke readXML for remaining rows
MigratorHelper.readXML(…);

// Append errors from above to any errors that may have occurred
      within readXML()
if (migRowStatusList.size() > 0){
ArrayList errorList =
       migStatus.getObjectErrorMap().get(<vo>.getName());
if (errorList != null)
       errorList.addAll(migRowStatusList);
else
```

```
        migStatus.getObjectErrorMap().put(<vo>.getName(),migRowStatus
        List);
}
```

Note

It is recommended that ALL validations are captured at the EO layer and triggered within readXML() rather than having different sets of validations in the import API. Use custom validation in import API only on an exception basis.

Handling Special Cases of Export and Import APIs

Date Effective Business Objects

The recommended approach for handling primary keys in setup export and import services is to:

- Suppress the primary key ID at the source instance during export and to use alternate key instead.
- Retrieve the primary key ID at the target instance during import by using the exported alternate keys.

With date effective business objects, the alternate key approach for determining the surrogate primary key ID will not work because multiple rows can share the same surrogate ID with different effective dates.

Consider the following example:

ID	Start Effective Date	End Effective Date	User Key
10	1-Feb-2009	10-Feb-2009	ABC
10	11-Feb-2009		ABC

If the standard alternate key approach is used to import at a target instance where none of these rows existed before, each of these rows would be inserted with a different ID, which will not be the desired result.

To circumvent this problem, the application developers should use view criteria to determine the surrogate ID based on the user key. That is:

- Apply the view criteria to the VO to determine the surrogate ID (for each XML row) based on its corresponding user key that is available in the exported XML.
- After the surrogate ID is retrieved, append it to the XML. See the section called Foreign Key IDs as Part of Alternate Keys under Updating Primary Keys in XML Data Elements for details on how this

can be done. The only difference is, instead of using an alternate key to locate a row you will use the view criteria.

Once the primary key attributes , the surrogate ID and effective dates, are retrieved, readXML() can be performed as per usual process.

Note

Do not use updatePrimaryKeysInXML() for parents because the primary key ID is already determined using the view criteria.

Propagating Primary Keys To Children: If there are children VOs that need the parent primary key ID to be propagated to resolve their alternate keys, then use propagatePrimaryKeysInXML(Element, ViewObjectImpl) to propagate primary keys to the underlying view object tree.

Custom Validation: If custom validation needs to be performed during the import process, application developers must implement them within the EO validation as per their requirements. Consider the following guidelines:

- EO layer should contain all date effective validations.
- No special handling is required during the export process.
- The target instance should be treated as the source of the truth and conflicting records that violate date effective principles should be rejected during import.

Generating Web Services

Once the setup export and import methods are properly implemented, application developers should generate web services and expose those methods.

Generating Web Services from Application Modules

Using the Application Module Editor, enable a service interface for the application module.

1. Go to the Service Interface tab and add a new service interface.
2. Add MTOM annotation
    - Add @MTOM to <xxx>ServiceImpl.java
    - Add it just above: public class <xxx>ServiceImpl extends ServiceImpl implements <xxx>Service {
    - Use import javax.xml.ws.soap.MTOM;
3. Generate asynchronous web service methods.

- Go to the Service Interface
- Click Edit
- Check Generate Asynchronous Web Service methods

Expose exportData() and importData() Methods in the Service

Use Service Interface Custom Methods section in the Service Interface tab in your Application Module and click Edit to expose the custom methods.

Expose exportData() Method

1. Select exportData() by moving it from Available to Selected
2. Expand exportData() node
3. Expand Return node and select View Object: ExportReturn
4. Expand Parameters node and select the following:
    - View Object: ExportCriteria
    - View Object: ExportControl

Expose importData() Method

1. Select importData() by moving it from Available to Selected
2. Expand importData() node
3. Expand Return node and select View Object: ImportReturn
4. Expand Parameters node and select the following:
    - View Object: ImportCriteria
    - View Object: ImportControl

Web Service Security

All setup export and import web services should be secured using the Oracle Fusion standards and guidelines. FSM does not have any special requirements.

Developing Non ADF BC Object-Based Export and Import APIs

Setup export and import service APIs, as described for ADF BC view objects, can also support data that are not based on ADF BC view objects

when appropriately customized. For example, to migrate MDS content, a service can be created which invokes MBean export and import APIs of MDS within the setup export and import APIs.

Use the following steps to develop these services.

1. Create Service Application Module (AM) that implements Migrator interface
2. Implement FSM export and import methods
3. Generate web services

Implementing Export API

A custom method called exportData() must be implemented in Application Module class as follows.

Identifying Filter Criteria

The following helper method returns a HashMap of attribute sets and their respective criteria in terms of attributes and their values.

```
Helper method in
      oracle.apps.setupHub.remoteApp.publicModel.migrator.MigratorH
      elper
public static HashMap<String, HashMap<String, List>>
getAttributeCriteriaValuesBySet (
      List<ExportCriteriaVORowImpl> exportCriteria,
    String objectKey)
```

Note

If Search task is defined for the object, then when creating the configuration package for setup export, you can search and select values which can then be used to filter exported data. See Business Object Details for more information of Search tasks of business objects.

Implementing Custom Export API for Non ADF BC Objects

Any logic necessary to export non ADF BC objects should be implemented. For example, when exporting MDS content, MBean export APIs of MDS should be invoked.

If multiple objects are processed during export, the following helper methods can be used to track the start and the end times of each object.

```
migStatus.setObjectStartTime(Object1, Calendar.getInstance());
// Process object….
migStatus.setObjectEndTime(Object1, Calendar.getInstance());
```

```
// If any error occurs for a specific object, it can also be
      tracked as follows:
migStatus.setObjectProcessingError(Object1, Some error occurred for
      Object1.);
```

Maintaining Hash Table for Export Related XML Files

Once XML files are generated by setup export, they must be added to a Hashtable. The keys will be the filenames, which can be of the following format: <ObjectName>.txt, <ObjectName>.xml, and so on.

As a standard, the file extensions can be used to represent the type of the content.

During import, the filenames will be used to identify the data being imported.

```
Hashtable<String, byte[]> files
```

Returning XML File Collection Generated by Export

The collection of XML files generated by export must be returned to FSM as shown.

```
migStatus.getObjectReportMetadataMap().put(node.getNodeName(),
      getReportMetadata((ViewObjectImpl)vo));ExportReturnVORowImpl
      returnRow =
(ExportReturnVORowImpl)this.getExportReturn().createRow();

returnRow.setDataFiles(files);
```

Returning Errors

If there are any errors, they can be returned to FSM as follows.

```
MigratorStatus migStatus = new MigratorStatus();

// General Processing error maybe reported as follows:
migStatus.setProcessingError(xxx);

// Additional detail object errors may be reported as follows:
migStatus.getObjectProcessingErrorMap().put(Object1, error);

// Additional detail object row errors may be reported as follows:
ArrayList<MigratorRowStatus> migRowStatusList = new
                   ArrayList<MigratorRowStatus>(1);

MigratorRowStatus migRowStatus = new
MigratorRowStatus( <rowXML>, // Row that caused the error
<errorMessage>, // Detailed Error message
<errorCode>, // Error Code, if any
<errorSeverity>, // Error Severity, if any
```

```
<alternateKey> // Use of alternate key is preferable instead of an
      Id, since it is visible to user
));

migRowStatusList.add(migRowStatus);

migStatus.getObjectErrorMap().put(Object1, migRowStatusList);

// Return status
returnRow.setMigratorStatus(migStatus);
```

Reporting on Exported Metadata

To generate BIP reports on exported XML data, the same method should be used as specified for Reporting on Exported XML data for ADF BC based Objects.

However, for custom XML files, application developers need to generate report metadata themselves in XSL-FO format.

Warning

Detailed information on BIP and XSL-FO standards are beyond the scope of this document. Consult appropriate documentation and training material to learn about BIP technology and recommended standards and guidelines.

Example of Export API

```
public ExportReturnVORowImpl exportData
      (List<ExportCriteriaVORowImpl>
                  exportCriteria, ExportControlVORowImpl
      exportControl) {

  ExportReturnVORowImpl returnRow = (ExportReturnVORowImpl)
           getExportReturn().createRow();
  MigratorStatus migStatus = new MigratorStatus();

  try {
    // Get Filter Criteria
    HashMap<String, List> attributeCriteriaValues =

      MigratorHelper.getExportCriteriaValues(exportCriteria,
                  MigratorConstants.CFG_PACKAGE_BO_SHORT_NAME);

    // Perform custom export of objects using criteria that is
      passed in
    Hashtable<String, byte[]> files = new  Hashtable<String,
      byte[]>(1);

    // Process Object 1 - this can be any custom process
    migStatus.setObjectStartTime(Object1, Calendar.getInstance());
    String object1 = Object1 content.;
```

```
    // Save the file bytes to the hashtable
    files.put(Object1.txt, object1.getBytes());
    migStatus.setObjectEndTime(Object1, Calendar.getInstance());

    // Process Object 2 – this can be any custom process
    migStatus.setObjectStartTime(Object1, Calendar.getInstance());
    String object2 = Object2 content.;
    // Save the file bytes to the hashtable
    files.put(Object2.txt, object2.getBytes());
    migStatus.setObjectEndTime(Object2, Calendar.getInstance());

    // If any error occurred for a specific object, track it
    migStatus.setObjectProcessingError(Object2, Error occurred for
      object2.);

    returnRow.setDataFiles(files);
 }
 catch (Exception e){
    // If any general processing error occurred, track it
    migStatus.setProcessingError(e.getLocalizedMessage());
 }

 // Return Status + Data
 returnRow.setMigratorStatus(migStatus);
 return returnRow;
}
```

Implementing Import API

A custom method called importData() must be implemented in Application Module class as follows.

Identify XML Data Files

The following helper method retrieves the XML files generated during export:

```
Hashtable<String, Element> files = importData.getDataFiles();
byte[] object1 = files.get(Object1.txt);
```

Implementing Custom Validation

Similar to export process, any custom validation that is required for proper data import should be implemented. If multiple objects are processed during import, the following helper method can be used to track the start and the end times of each object.

```
migStatus.setObjectStartTime(Object1, Calendar.getInstance());
// Process object….
migStatus.setObjectEndTime(Object1, Calendar.getInstance());
```

```
// If any error occurs for a specific object, it can also be
      tracked as follows:
migStatus.setObjectProcessingError(Object1, Some error occurred for
      Object1.);
```

Returning Errors

Same as in export process. See Returning Errors during Export for more details.

Reporting on Imported Metadata

Same as in export process. See Reporting on Exported Metadata for more details.

Example of Import API

```
public ImportReturnVORowImpl importData (ImportDataVORowImpl
      importData,
                                          ImportControlVORowImpl
      importControl) {
  ImportReturnVORowImpl returnRow = (ImportReturnVORowImpl)
                                  getImportReturn().createRow();
  MigratorStatus migStatus = new MigratorStatus();

  try {
    Hashtable<String, byte[]> files = importData.getDataFiles();

    // Retrieve the file data the way it was exported
    byte[] object1 = files.get(Object1.txt);

    // Process Object 1 - this can be any custom process
    migStatus.setObjectStartTime(Object1, Calendar.getInstance());
    if (object1 != null)
       System.out.println(new String(object1));
    migStatus.setObjectEndTime(Object1, Calendar.getInstance());

    // Retrieve the file data the way it was exported
    byte[] object2 = files.get(Object2.txt);

    // Process Object 2 - this can be any custom process
    migStatus.setObjectStartTime(Object2, Calendar.getInstance());

    if (object2 != null)
       System.out.println(new String(object2));
    migStatus.setObjectEndTime(Object2, Calendar.getInstance());

    // If any error occurred for a specific object, track it
    migStatus.setObjectProcessingError(Object2, Error occurred for
      object2.);
 }
 catch (Exception e){
   // If any general processing error occurred, track it
```

```
      migStatus.setProcessingError(e.getLocalizedMessage());
    }

    // Return Status
    returnRow.setMigratorStatus(migStatus);
    return returnRow;

  }
```

Setting Up Export and Import of Larger Data Volume

When larger volume of setup data needs to be exported and imported, handling all data in a single batch may not be possible due to memory constraints and performance degradation. For such cases, FSM provides ability to invoke a setup export and import Service multiple times and process data in batches.

When to Consider Batch Processing

While it is impossible to provide the exact parameters when batch processing is preferable, application developers may consider factors such as:

- Number of View Objects to be exported or imported
- Number of columns in the View Objects
- Column types and sizes
- Number of estimated rows to be exported or imported in typical cases

Tip

Follow the performance guidelines for Oracle Fusion ADF BC Services.

Configurable Batching Properties

The following properties can be configured when using batch processing for setup export and import.

Property	Required	Used By	Description
setBatchMode(boolean)	Yes	Export	Indicates if batch processing will be used
setRangeSize(int)	No	Export	Specifies the number of rows to be processing in one batch. The default value is set to 500.
setBatchImportSequence(ArrayList)	No	Export	Specifies the order in which the exported view objects will be imported. The default

			setting is to use the same order in which the view objects were exported.
setStopBatchProcess(boolean)	No	Export	Indicates whether or not to use batch processing, meaning to invoke the corresponding service again.
setStopBatchProcess(boolean)	No	Import	Indicates whether or not to use batch processing, meaning to invoke the corresponding service again.

How Batch Processing Works

When setBatchMode() is set to True indicating the batch processing will be used, FSM invokes the export and import Service as many times as needed to process all data or an error is encountered.

During export, FSM uses specified batch size and creates as many data files needed to export all data.

During import, each exported batch is returned to the service until all batches are processed. Each batch is committed separately.

Please note that data for each batch is stored in its own ZIP file, and every batch of data is committed separately. It is important that batching set is designed keeping these facts in mind.

For example, consider a view object called TablePerf for setup business object called TEST_BO which is exported in batches with batch size of 500. The result after an export is:

- TEST_BO.zip contains:
- TablePerf/1_BATCH.zip
- TablePerf/2_BATCH.zip
- TablePerf/3_BATCH.zip
- TablePerf/4_BATCH.zip
- TablePerf/5_BATCH.zip
- TablePerf/6_BATCH.zip
- TablePerf/7_BATCH.zip
- TablePerf/8_BATCH.zip
- TablePerf/9_BATCH.zip
- TablePerf/10_BATCH.zip

Since TEST_BO had 5000 rows, the service was invoked ten times during export. Although separate zip file is created for each batch, each <n>_BATCH.zip will contain TEST_BO.xml. Similarly, during import, FSM will invoke the service ten times, processing each of the batches separately.

Note

If a setup object contains very large volume of data (for example, has millions of rows or contains BLOB), then services are not recommended for exporting and importing the data. A loader program may be more appropriate in such cases. Follow standard and guidelines of ADF BC services to determine when a loader program is a better approach.

Implementing Batch Processing

To enable batch processing, add the following steps to setup export and import APIs.

Exporting API Modification

Exporting Single View Object

To enable batch processing of a single view object:

Initialize migratorControl(); for example:

```
MigratorControl migControl = new MigratorControl(exportControl);
migControl.setBatchMode(true);
migControl.setRangeSize(500);
```

Pass migControl to getViewObjectXML() API

```
public static byte[] getViewObjectXML(MigratorStatus migStatus,
      ViewObject vo, HashMap<String, ViewAttributeDefImpl[]>
      voIncludedAttrMap, boolean debugMode, MigratorControl
      migControl)
```

This API exports data per the specified range size and start of the next range if more data is found. The stopBatchProcess is set to True when all rows in the view object are processed.

Application developers can optionally use setStopBatchProcess() method in their export services any time to stop batch processing (such as when an error is encountered) and consequently stop FSM from invoking the service again during the processing.

Sample Code

```
public ExportReturnVORowImpl
      exportData(List<ExportCriteriaVORowImpl> exportCriteria,
                                              ExportControlVORowImpl
      exportControl) {

   ExportReturnVORowImpl returnRow =
            (ExportReturnVORowImpl)getExportReturn().createRow();

   MigratorStatus migStatus = new MigratorStatus();

   // 1. Initialize Batching
   MigratorControl migControl = new MigratorControl(exportControl);
   migControl.setBatchMode(true);

   try {
     boolean debugMode = false;
     Hashtable<String, byte[]> files = new Hashtable<String,
      byte[]>(1);


     HashMap<String, String[]> excludedAttrMap1 = new
      HashMap<String,
                  String[]>(2);
     String[] tableAExcludedAttributes1 = {  };


      excludedAttrMap1.put(oracle.apps.setup.test.model.view.TableL
      ookupVO,
                  tableAExcludedAttributes1);

     // 2. Include MigratorControl
     byte[] xmlDoc1 = MigratorHelper.getViewObjectXML(migStatus,
                  getTableLookup(),

      MigratorHelper.getViewObjectIncludedAttrMap(excludedAttrMap1)
      ,
                  debugMode, migControl);
     files.put(TABLE_LOOKUP_SHORT_NAME + .xml, xmlDoc1);

     returnRow.setDataFiles(files);
     this.getTableA().applyViewCriteria(null);

   } catch (Exception e) {
       migStatus.setProcessingError(e.getLocalizedMessage());
 }

 System.out.println(migStatus.gand so ononsolidatedErrors());
 returnRow.setMigratorStatus(migStatus);

 return returnRow;
}
```

Exporting Multiple View Objects

To enable batch processing of a multiple view objects:

- Initialize MigratorControl

```
MigratorControl migControl = new MigratorControl(exportControl);
migControl.setBatchMode(true);
migControl.setRangeSize(500);
```

- Check batch name before processing the first view object

```
if (migControl.gand so onurrentBatch() == null ||
getVO1().equals(migControl.gand so onurrentBatch()))
```

- Pass migControl to existing getViewObjectXML() API

```
public static byte[] getViewObjectXML(MigratorStatus migStatus,
      ViewObject vo, HashMap<String, ViewAttributeDefImpl[]>
      voIncludedAttrMap, boolean debugMode, MigratorControl
      migControl)
```

- If processing of current view object is complete, but there are view objects to be processed in the next batch, then specify the next batch name, and indicate that batch processing should continue.

```
if (migControl.isStopBatchProcess()){
                    migControl.setNextBatch(getVO2().getName());
                    migControl.setStopBatchProcess(false);
    }
```

- Check batch name before processing next view object

```
if (getVO2().equals(migControl.gand so onurrentBatch()))
```

- Specify the sequencing of batches to be used during import. FSM will send data in the specified sequence to import processing. By default, the import sequence is the same in which the view objects were exported.

```
ArrayList<String> list = new ArrayList<String>(2);
list.add(this.getVO2().getName());
list.add(this.getVO1().getName());
migControl.setBatchImportSequence(list);
```

Sample Code

```
public ExportReturnVORowImpl
      exportData(List<ExportCriteriaVORowImpl> exportCriteria,
                                          ExportControlVORowImpl
      exportControl) {

   ExportReturnVORowImpl returnRow =
            (ExportReturnVORowImpl)getExportReturn().createRow();
```

```
MigratorStatus migStatus = new MigratorStatus();

// 1. Initialize Batching
MigratorControl migControl = new MigratorControl(exportControl);
migControl.setBatchMode(true);

try {
  boolean debugMode = false;
  Hashtable<String, byte[]> files = new Hashtable<String,
   byte[]>(1);


  //2. Export VO 1 in first batch set
  if (migControl.gand so onurrentBatch() == null ||
           getTableLookup().equals(migControl.gand so
   onurrentBatch())){
      HashMap<String, String[]> excludedAttrMap1 = new
   HashMap<String,
               String[]>(2);
      String[] tableAExcludedAttributes1 = {  };


   excludedAttrMap1.put(oracle.apps.setup.test.model.view.TableL
   ookupVO,
              tableAExcludedAttributes1);

      // 3. Include MigratorControl
      byte[] xmlDoc1 =
   MigratorHelper.getViewObjectXML(migStatus,
               getTableLookup(),

   MigratorHelper.getViewObjectIncludedAttrMap(excludedAttrMap1)
   ,
               debugMode, migControl);
      files.put(TABLE_LOOKUP_SHORT_NAME + .xml, xmlDoc1);

      // 4. If batching processing of this VO is complete but
   you need to
            process next VO
      // Set the next VO and indicate that batch processing is
   not complete yet
      if (migControl.isStopBatchProcess()){
                migControl.setNextBatch(getTableA().getName());
                migControl.setStopBatchProcess(false);
      }
  }

  // Export VO 2 in next batch
  if (getTableA().getName().equals(migControl.gand so
   onurrentBatch())){
       // TableA
       HashMap<String, List> attributeCriteriaValues =
       MigratorHelper.getExportCriteriaValues(exportCriteria,
               TABLE_A_SHORT_NAME);


   MigratorHelper.applyCriteriaToViewObject(this.getTableA(),
```

```
                attributeCriteriaValues);

            HashMap<String, String[]> excludedAttrMap = new
        HashMap<String,
                String[]>(2);
            String[] tableAExcludedAttributes = {  ABC, Id1, ParentId
        };

        excludedAttrMap.put(oracle.apps.setup.test.model.view.TableAV
        O,
                tableAExcludedAttributes);

            // 5. Include MigratorControl
            byte[] xmlDoc =
        MigratorHelper.getViewObjectXML(migStatus, getTableA(),

        MigratorHelper.getViewObjectIncludedAttrMap(excludedAttrMap),
                                debugMode, migControl);

            files.put(TABLE_A_SHORT_NAME + .xml, xmlDoc);

       }
       returnRow.setDataFiles(files);
       this.getTableA().applyViewCriteria(null);

     } catch (Exception e) {
         migStatus.setProcessingError(e.getLocalizedMessage());
  }

  System.out.println(migStatus.gand so ononsolidatedErrors());
  returnRow.setMigratorStatus(migStatus);

  return returnRow;
}
```

Import API Modification

During import, FSM sends data to the import services in batches if the same data was exported through batch processing. As in the case export, setStopBatchProcess() method can be optionally used at any time in the import service to stop batch processing (such as if an error is encountered), and consequently stop FSM from invoking the service again during the processing.

Use the following steps to enable batch processing during setup import.

- Initialize MigratorControl

```
MigratorControl migControl = new MigratorControl(importControl);
```

- Identify data that was exported in batches

```
Element vo1Element = files.get(VO1.xml);
            if (vo1Element!= null) {
    // Process that XML
```

```
}
```

- Pass migControl to existing readXML() API

```
public static void readXML(MigratorStatus migStatus, ViewObjectImpl
      vo, Element element, int depthCount, boolean
      bundledExceptionMode, boolean debugMode, MigratorControl
      migControl)
```

Sample Code

```
public ImportReturnVORowImpl importData(ImportDataVORowImpl
      importData,
                                              ImportControlVORowImpl
      importControl) {

     ImportReturnVORowImpl returnRow =
           (ImportReturnVORowImpl)getImportReturn().createRow();
     MigratorStatus migStatus = new MigratorStatus();

     // 1. Initialize MigratorControl
     MigratorControl migControl = new
      MigratorControl(importControl);

     try {
       Hashtable<String, Element> files =
      importData.getDataXMLElements();

       Element lookupRootElement =
      files.get(TABLE_LOOKUP_SHORT_NAME + .xml);

       // 2. Check for each element
       if (lookupRootElement != null) {
         boolean partialFailureAllowed = false;
         boolean debugMode = true;
         // 3. Include MigratorControl
         MigratorHelper.readXML(migStatus,
              this.getTableLookup(),lookupRootElement, -1,
      partialFailureAllowed,
              debugMode, migControl);
            }

            Element parentRootElement =
      files.get(TABLE_A_SHORT_NAME + .xml);
            // 4. Check for each element
            if (parentRootElement != null) {
                boolean partialFailureAllowed = false;
                boolean debugMode = false;


      MigratorHelper.updatePrimaryKeysInXML(parentRootElement,
                    this.getTableAFind(), AltKey,
      partialFailureAllowed);
                // 5. Include MigratorControl
                MigratorHelper.readXML(migStatus, this.getTableA(),
```

```
                    parentRootElement, -1,
      partialFailureAllowed, debugMode,
                    migControl);

            }
            returnRow.setMigratorStatus(migStatus);

            if
      (this.getName().equals(this.getRootApplicationModule().getNam
      e()))
                this.getTransaction().commit();
        }
        catch (Exception e) {
            e.printStackTrace();
            migStatus.setProcessingError(e.getLocalizedMessage());
        }

        return returnRow;
}
```

Recommended Testing of Setup Export and Import Services

Testing Export Services

AM Tester

Launch the AM Tester by right clicking on Application Module (AM), select View Object, and verify that all view object data including that of children view objects exist.

Direct AM Call

FSM has provided the following method in MigratorHelper class to verify export within AM. This method can be used as a sample and added to your test method in your test class or can be added to your main() method in your AMImpl class.

```
public static void main (String[] args) {
testExport();
}

private static void testExport() {

 // Create your application module
        XXXAMImpl am =

      (XXXAMImpl)Configuration.createRootApplicationModule(oracle.a
      pps.xxx.XXXService.applicationModule.
        XXXAM, XXXAMLocal);
```

```
 // Populate any scope values to use as criteria
List<ExportCriteriaVORowImpl> exportCriteria = new
      ArrayList<ExportCriteriaVORowImpl>(1);

        ExportCriteriaVORowImpl criteriaRow =
      (ExportCriteriaVORowImpl)am.getExportCriteria().createRow();
        criteriaRow.setObjectKey(<XXX_BO_SHORT_NAME>);
        criteriaRow.setAttributeName(XXXXShortName);
        criteriaRow.setAttributeValue(ABC);
        exportCriteria.add(criteriaRow);

 // Create the Export Control
        ExportControlVORowImpl exportControl =
      (ExportControlVORowImpl)am.getExportControl().createRow();

 // Provide a name to save the exported zip
        String exportFileName = /home/mdamle/temp/testExport.zip;

 // Invoke the testExport() method that saves the exported file to
      provided location
 MigratorStatus migStatus = MigratorHelper.testExport(am,
      exportFileName, exportCriteria,
             exportControl);

       // Check MigratorStatus methods for additional processing
      errors if any
       System.out.println(Error found (Check migStatus object for
      details):  +
               migStatus.isErrorFound());

       Configuration.releaseRootApplicationModule(am, true);

}
```

Junit Test

Use the following export APIs for your Junit tests. This test will let you verify export functionality without specifying the physical location of an export file.

AM Layer Test

Use as follows:

```
public static MigratorStatus testExport(Migrator m,
      OutputStream outputStream, List<ExportCriteriaVORowImpl>
      exportCriteria, ExportControlVORowImpl exportControl)
```

Sample Code

```
List<ExportCriteriaVORowImpl> exportCriteria = new
      ArrayList<ExportCriteriaVORowImpl>(1);
```

```
    ExportCriteriaVORowImpl criteriaRow =
          (ExportCriteriaVORowImpl)am.getExportCriteria().createRo
          w();

            ByteArrayOutputStream baos = new
          ByteArrayOutputStream();
            ExportControlVORowImpl exportControl =
          (ExportControlVORowImpl)am.getExportControl().createRow(
          );
            MigratorStatus migStatus  =
          MigratorHelper.testExport(am, baos, exportCriteria,
          exportControl);

            System.out.println(migStatus.gand so
          ononsolidatedErrors());
```

Testing Import Services

AM Tester

Launch the AM Tester by right clicking on Application Module (AM), select View Object, and verify that all view object data including that of children view objects exist.

Direct AM Call

FSM has provided the following method in MigratorHelper class to verify import within AM. This method can be used as a sample and added to your test method in your test class or can be added to your main() method in your AMImpl class.

```
public static void main (String[] args) {
testImport();
}

private static void testImport() {

 // Create your application module
        ConfigurationTemplateCategoryAMImpl am =
(ConfigurationTemplateCategoryAMImpl)Configuration.createRootApplic
      ationModule(oracle.apps.setup.commonSetup.configurationTempla
      tes.ConfigurationTemplateCategoryService.applicationModule.Co
      nfigurationTemplateCategoryAM,
      ConfigurationTemplateCategoryAMLocal);

 // Create Import Control
ImportControlVORowImpl importControl =
      (ImportControlVORowImpl)am.getImportControl().createRow();

 // Provide name of zip to import
String importFileName = /home/mdamle/temp/testExport.zip;
try {
```

```
    // Invoke the testImport() method to import the zip file
          MigratorStatus migStatus = MigratorHelper.testImport(am,
      importFileName, importControl);
    // Check MigratorStatus methods for additional processing
      errors if any
   System.out.println(Error found (Check migStatus object for
      details):  +
                  migStatus.isErrorFound());
}
catch (Exception e) {
    e.printStackTrace();
}

Configuration.releaseRootApplicationModule(am, true);

}
```

Junit Test

Use the following import APIs for your Junit tests. This test will let you verify import functionality without specifying the physical location of an import file.

AM Layer Test

Use as follows:

```
public static MigratorStatus testImport(Migrator m,
      InputStream inputStream, ImportControlVORowImpl
      importControl)
```

Sample Code

```
List<ExportCriteriaVORowImpl> exportCriteria = new
      ArrayList<ExportCriteriaVORowImpl>(1);

ExportCriteriaVORowImpl criteriaRow =
      (ExportCriteriaVORowImpl)am.getExportCriteria().createRo
      w();

        ByteArrayOutputStream baos = new
      ByteArrayOutputStream();
        ExportControlVORowImpl exportControl =
      (ExportControlVORowImpl)am.getExportControl().createRow(
      );
        MigratorStatus migStatus  =
      MigratorHelper.testExport(am, baos, exportCriteria,
      exportControl);

        BlobDomain blob = new BlobDomain(baos.toByteArray());
```

```
  ImportControlVORowImpl importControl =
(ImportControlVORowImpl)am.getImportControl().createRow(
);

  migStatus = MigratorHelper.testImport(am,
blob.getBinaryStream(), importControl);
  System.out.println(migStatus.gand so
ononsolidatedErrors());
```

9

Integrating with FSM - Java APIs and Web Services

About Integrating with FSM

FSM provides public Java APIs and web services which can be used to set or retrieve information of offerings, options, and features (collectively also known as features) selections and task statuses of an implementation project. In some of these cases, EL expression can also be used to achieve the same results.

Warning

Application developers who are using FSM public Java APIs and web services should be proficient in ADF, JDeveloper, and web service technologies, terminologies, standards, and guidelines.

FSM Java APIs

FSM Java APIs are available in an ADF library called AdfFeaturePublicModel.jar located in $ADE_HOME/Oracle Fusionapps/jlib/.

Currently APIs are available for offering, option, and feature-related information only.

Invoking Java APIs

To access these APIs, use the following steps:

1. Include the library in your project.
2. Embed FeatureAM in data model.
3. Access methods using the FeatureAM instance.

About FSM Public Web Services

FSM web services are available in FunctionalSetupServiceClient.jar. Currently, they are available in two different packages as follows:

- **Feature Related**: oracle.apps.setup.commonSetup.filters.FeatureService.FeatureService
- **Task Status Related**: oracle.apps.setup.commonSetup.implementationProjects.ImplementationProjectService.TaskStatusService

Invoking Web Services

You can use two types of approaches to invoke FSM web services.

ADF Standards

This approach is recommended for integrating ADF business component services with ADF applications. Use the following steps:

1. Modify CONNECTIONS.XML by adding the following snippet and replacing [PACKAGE] and [SERVICE] variables with FEATURE or TASK STATUS Package and Service names.

```
<Reference name={http://xmlns.[PACKAGE]/}[SERVICE]
      className=oracle.jbo.client.svc.Service xmlns=>
      <Factory
      className=oracle.jbo.client.svc.ServiceFactory/>
      <RefAddresses>
         <StringRefAddr addrType=serviceInterfaceName>
            <Contents>[PACKAGE]</Contents>
         </StringRefAddr>
         <StringRefAddr addrType=serviceEndpointProvider>
            <Contents>ADFBC</Contents>
         </StringRefAddr>
         <StringRefAddr addrType=jndiName>
            <Contents>[SERVICE]Bean#[PACKAGE]</Contents>
         </StringRefAddr>
         <StringRefAddr addrType=serviceSchemaName>
            <Contents>[SERVICE].xsd</Contents>
         </StringRefAddr>
         <StringRefAddr addrType=serviceSchemaLocation>
            <Contents>[PACKAGE/SERVICE DIR]</Contents>
         </StringRefAddr>
         <StringRefAddr addrType=jndiFactoryInitial>

      <Contents>weblogic.jndi.WLInitialContextFactory</Content
      s>
         </StringRefAddr>

         <StringRefAddr addrType=jndiProviderURL>
<Contents>t3:[URL OF YOUR FSM INSTANCE]</Contents>
         </StringRefAddr>
      </RefAddresses>
   </Reference>
```

2. Invoke the services as follows:

```
[SERVICE] svc = ([SERVICE]) ServiceFactory.getServiceProxy
      ([SERVICE].NAME);
where [SERVICE] is either FeatureService or TaskStatusService
```

3. Once you get the service object, all methods can be called with appropriate input parameters and exceptions can be caught in try/catch block. For example,

```
Result = svs.getFeatureChoice(newList, FeatureShortName);
svc.getTaskStatus(id);
svc.setTaskStatus(id,REQUEST_INFORMATION);
```

Industry Standards

You can also invoke these services by creating web service proxy or using SOAP.

To get WSDL, append the following after host:port information in the URL of your FSM instance:

```
Setup-SetupModel-context-root/BusinessProcessService?WSDL.
For example, http://[HOST:PORT]/Setup-SetupModel-context-
root/BusinessProcessService?WSDL
```

Note

For more detailed understanding on how to use these web services, you should be familiar with and follow the recommended approach for integrating web services into an Oracle Fusion web application and synchronously invoking an ADF business components service from an ADF application as specified in Oracle Fusion Applications Developer's Guide.

Offerings, Options, and Features Integration

Integration is available in the following categories:

- Retrieving or setting implementation status of an offering or option
- Retrieving or setting feature choices
- Retrieving offering, option, and feature hierarchy
- Retrieving lists of associated objects for enabled offerings and options

Implementation Status Integration

Each offering and option has a property called Implementation Status which indicates whether or not the offering and option has been implemented.

While FSM UI allows implementation managers to set its value, Java APIs and web services are also available to retrieve and set the property programmatically. EL expressions can also be used to retrieve the current settings.

Retrieving Implementation Status

Use EL expression or getStatus API or web services to retrieve implementation status of an offering or option.

Using EL Expression

Use the following steps to retrieve implementation status of an offering or option.

1. Define a managed bean called oracle.apps.setup.commonSetup.features.Offerings with the VARIABLE called OFFERINGS at REQUESTSCOPE LEVEL in adfc-config.xml file.
2. Refer to implementation status of an offering or option by using: #{Offerings.implementationStatus.OFFERING_SHORT_NAME}, where OFFERING_SHORT_NAME is the actual short name (code) of the desired offering or option.

Using getStatus Java API

See Invoking Java APIs for more information on how to invoke the API.

Name	getStatus
Purpose	To retrieve implementation status of an offering or option
Definition	public String getStatus(String featureShortName)
Parameters	featureShortName - Offering or option identifier as a string
Results returned	The status of the specified Offering or option returned as a String Valid values are one of the following: NOT_IMPLEMENTED - If no match for the input value is found NOT_STARTED_FEATURE IN_PROGRESS_FEATURE IMPLEMENTED
Exception condition	N/A
Sample code	Constructing Input String featureShortName = PAY_LEGISLATION;

	Call Method
	FeatureService svc = (FeatureService) ServiceFactory.getServiceProxy (FeatureService.NAME);
	
	try {
	String status = svc.getStatus(featureShortName);
	} catch(Exception e) {
	e.printStackTrace();
	}

Using getStatus Web Service

See Invoking Web Services for more information on how to invoke the web services.

Name	getStatus
Purpose	To retrieve implementation status of an Offering or Option
Definition	public String getStatus(String featureShortName)
Parameters	featureShortName – The identifier of an Offering or option as a String
Results returned	The status of the specified Offering or option returned as a String Valid values are one of the following: NOT_IMPLEMENTED – If no match for the input value is found NOT_STARTED_FEATURE IN_PROGRESS_FEATURE IMPLEMENTED
Exception condition	N/A

Sample code	Constructing Input
	String featureShortName = PAY_LEGISLATION;
	
	Call Method
	FeatureAMImpl featureAM = (FeatureAMImpl)<OriginalAM>.getFeatureAM();
	
	try {
	String status = featureAM.getStatus(featureShortName);
	} catch(Exception e) {
	e.printStackTrace();
	}

Setting Implementation Status

Use setStatus Java API or web service to set implementation status of an offering or option. EL expression cannot be used to set implementation status.

Using setStatus Java API

Name	setStatus
Purpose	To set the implementation status of a Feature
Definition	public void setStatus(String featureShortName, String status)
Parameters	featureShortName – The identifier (or Code) of an Offering or option as a String Status – the implementation status of the Offering or option as a String. Valid values are: NOT_STARTED_FEATURE IN_PROGRESS_FEATURE IMPLEMENTED
Results returned	N/A
Exception condition	Invalid Feature short name (code) and Status is specified

Sample code	Constructing Input
	
	String featureShortName = PAY_LEGISLATION;
	String status =IMPLEMENTED;
	
	Call Method
	FeatureAMImpl featureAM = (FeatureAMImpl)<OriginalAM>.getFeatureAM();
	
	try {
	featureAM.setStatus(featureShortName, status);
	} catch(Exception e) {
	e.printStackTrace();
	}

Using setStatus Web Service

Name	setStatus
Purpose	To set the implementation status of a Feature
Definition	public void setStatus(String featureShortName, String status)
Parameters	featureShortName – The identifier of an Offering or option as a String Status – the implementation status of the Offering or option as a String. Valid values are: NOT_STARTED_FEATURE IN_PROGRESS_FEATURE IMPLEMENTED
Results returned	N/A
Exception condition	Invalid Feature short name (code) and Status is specified

Sample code	**Constructing Input**
	String featureShortName = PAY_LEGISLATION;
	String status ='IMPLEMENTED';
	
	Call Method
	FeatureService svc = (FeatureService) ServiceFactory.getServiceProxy (FeatureService.NAME);
	
	try {
	svc.setStatus(featureShortName, status);
	} catch(Exception e) {
	e.printStackTrace();
	}

Feature Choice Integration

Each offering, option, or feature has two or more choices. Implementation managers make appropriate selection of these choices to configure their implementations as applicable to their business needs.

In addition to UI, feature choices can be set, current setting can be retrieved or current setting can be locked programmatically by using Java APIs and web services.

Retrieving Current Feature Choice Settings

Use getFeatureChoices Java API or web service to retrieve current settings of a specific offering, option, or feature.

Using getFeatureChoices Java API

Name	getFeatureChoices
Purpose	To get a list of Feature Choices and their values for a given Scope
Definition	public java.util.List<String> getFeatureChoices(java.util. List< oracle.apps.setupHub.remoteApp.publicModel.view.ScopeValueParamVORowImpl> scopeValueParams, java.lang.String featureShortName) throws oracle.jbo.JboException
Parameters	featureShortName – Offering, option, or feature identifier as a String scopeValueParams – List of Scope values identified by pairs of scope attribute name and short name (Code) of the Scope value. The list of Scope values could be NULL.
Results returned	A list of Feature choice short name (code); an empty list is returned if no match is found.
Exception condition	Invalid Feature short name (code) is used.

Sample code	**Constructing Input**
	FeatureAMImpl featureAM = (FeatureAMImpl)<originalAM>.getFeatureAM();
	/*create a list of ScopeValueParamVORowImpl Objects */
	ViewObject scopeVO = featureAM.getScopeValueParam();
	List<ScopeValueParamVORowImpl> newList = new ArrayList<ScopeValueParamVORowImpl>();
	ScopeValueParamVORowImpl newRow = (ScopeValueParamVORowImpl)scopeVO.createRow();
	/*set each element of the object with appropriate value */
	newRow.setattributeValue(BE-TaxAttribute1);
	newRow.setscopeValue(BE-Tax);
	newRow.setscopeValueCode(BE-Tax);
	/* Add to the list */
	newList.add(newRow);
	String featureShortName = PAY_LEGISLATION;
	
	Constructing Output
	/* construct a list of String */

	List<String> result = new ArrayList<String>();
	
	Call Method
	try {
	result = featureAM.getFeatureChoices(newList,FeatureShortName);
	} catch(JboException e) {
	e.printStackTrace();
	}
	
	Parsing Output
	if(result != null) {
	for(int i = 0; i < result.size(); i++)
	/* get the result of Feature option short name */
	System.out.println(The result is + result.get(i));
	}

Using getFeatureChoices Web Service

Name	getFeatureChoices
Purpose	To get a list of Feature Choices and their values for a given Scope
Definition	public java.util.List<String> getFeatureChoices(java.util. List< oracle.apps.setupHub.remoteApp.publicModel.view.ScopeValueParamVORowImpl> scopeValueParams, java.lang.String featureShortName) throws oracle.jbo.JboException
Parameters	featureShortName – Offering, option, or feature identifier as a String scopeValueParams – List of Scope values identified by pairs of scope attribute name and short name (code) of the Scope value. The list of Scope values could be NULL.
Results returned	A list of Feature choice short name (code); an empty list is returned if no match is found.
Exception condition	Invalid Feature short name (code) is used.
Sample code	Constructing Input /*create a list of ScopeValueParamVORowImpl Objects */ List<ScopeValueParamVORowImpl> newList = new ArrayList<ScopeValueParamVORowImpl>(); ScopeValueParamVORowImpl newRow = (ScopeValueParamVORowImpl)scopeVO.createRow();

	/*set each element of the object with appropriate value */
	newRow.setattributeValue(BE-TaxAttribute1);
	newRow.setscopeValue(BE-Tax);
	newRow.setscopeValueCode(BE-Tax);
	/* Add to the list */
	newList.add(newRow);
	String featureShortName = PAY_LEGISLATION;
	
	Constructing Output
	/* construct a list of String */
	List<String> result = new ArrayList<String>();
	
	Call Method
	FeatureService svc = (FeatureService) ServiceFactory.getServiceProxy (FeatureService.NAME);
	
	try {
	result = svc.getFeatureChoices(newList,FeatureShortName);
	} catch(JboException e) {
	e.printStackTrace();
	}
	
	Parsing Output
	if(result != null) {
	for(int i = 0; i < result.size(); i++)
	/* get the result of Feature option short name */
	System.out.println(The result is + result.get(i));
	}

Setting Feature Choices

Use setFeatureChoices Java API or web service to set feature choices of a specific offering, option, or feature.

Using setFeatureChoices Java API

Name	setFeatureChoices

Purpose	Set the Feature Choice value for a given Feature and Scope
Definition	public void setFeatureChoices (java.util.List<oracle.apps.setupHub.remoteApp.publicModel.view.ScopeValueParamVORowImpl> scopeValueParams, java.lang.String featureShortName, java.util.List<java.lang.String> featureChoice) throws oracle.jbo.JboException
Parameters	featureShortName – Offering, Option or Feature identifier as a String scopeValues – List of Scope values; they are of ScopeValueParam object type. The list of Scope values could be NULL. featureChoice - list of Feature Choices to be set via this method
Results returned	N/A
Exception condition	Invalid Feature short name (code) is used. Feature Choices used are not valid for Feature short name (code) used. More than one Feature Choices are used when the Feature type of the Feature short name (code) used is 'Single Selection'.

Sample code	**Constructing Input**
	FeatureAMImpl featureAM = (FeatureAMImpl)<originalAM>.getFeatureAM();
	/*create a list of ScopeValueParamVORowImpl Objects */
	ViewObject scopeVO = featureAM.getScopeValueParam();
	/*create a list of ScopeValueParamVORowImpl Objects */
	List<ScopeValueParamVORowImpl> newList = new ArrayList<ScopeValueParamVORowImpl>();
	ScopeValueParamVORowImpl newRow = (ScopeValueParamVORowImpl)scopeVO.createRow();
	/*set each element of the object with appropriate value */
	newRow.setattributeValue(BE-TaxAttribute1);
	newRow.setscopeValue(BE-Tax);
	newRow.setscopeValueCode(BE-Tax);
	/* Add to the list */
	newList.add(newRow);
	String featureShortName = PAY_LEGISLATION;
	List<String> result = new ArrayList<String>();
	result.add(PAY_LEGISLATION_US);
	
	Call Method
	FeatureAMImpl featureAM = (FeatureAMImpl)<OriginalAM>.getFeatureAM();

	try {
	featureAM.setFeatureChoices(newList,featureShortName,result);
	} catch(JboException e) {
	e.printStackTrace();
	}

Using setFeatureChoices Web Service

Name	setFeatureChoices
Purpose	Set the feature choice value for a given feature and scope
Definition	public void setFeatureChoices (java.util.List<oracle.apps.setupHub.remoteApp.publicModel.view.ScopeValueParamVORowImpl> scopeValueParams, java.lang.String featureShortName, java.util.List<java.lang.String> featureChoice) throws oracle.jbo.JboException
Parameters	featureShortName – Offering, Option or Feature identifier as a String scopeValues – List of scope values; they are of ScopeValueParam object type. The list of scope values could be null. featureChoice - list of feature choices to be set via this method
Results returned	N/A
Exception condition	Invalid feature short name (code) is used. Feature choices used are not valid for feature short name (code) used. More than one feature choice is used when the feature type of the feature short name (code) used is Single Selection.

Sample code	**Constructing Input**
	/*create a list of ScopeValueParamVORowImpl Objects */
	List<ScopeValueParamVORowImpl> newList = new ArrayList<ScopeValueParamVORowImpl>();
	ScopeValueParamVORowImpl newRow = (ScopeValueParamVORowImpl)scopeVO.createRow();
	/*set each element of the object with appropriate value */
	newRow.setattributeValue(BE-TaxAttribute1);
	newRow.setscopeValue(BE-Tax);
	newRow.setscopeValueCode(BE-Tax);
	/* Add to the list */
	newList.add(newRow);
	String featureShortName = PAY_LEGISLATION;
	List<String> result = new ArrayList<String>();

	result.add(PAY_LEGISLATION_US);
	
	Call Method
	FeatureService svc = (FeatureService) ServiceFactory.getServiceProxy (FeatureService.NAME);
	
	try {
	svc.setFeatureChoices(newList,featureShortName,result);
	} catch(JboException e) {
	e.printStackTrace();
	}

Freezing Current Feature Choice

Use freezeFeatureChoice Java API or web service to lock current settings of a specific offering, option, or feature.

Using freezeFeatureChoice Java API

Name	freezeFeatureChoice
Purpose	Locks feature choice settings of the specified feature . The choice selections cannot be updated anymore.
Definition	public void freezeFeatureChoice(List< java.util.List<oracle.apps.setupHub.remoteApp.publicModel.view.ScopeValueParamVORowImpl> scopeValueParams, String featureShortName) throws oracle.jbo.JboException
Parameters	featureShortName – Offering, Option, Feature identifier as String scopeValues – List of Scope values; they are of ScopeValueParams object type. The list of Scope values could be NULL.
Results returned	N/A
Exception condition	Invalid Feature short name (code) is used.
Sample code	**Constructing Input** FeatureAMImpl featureAM = (FeatureAMImpl)<originalAM>.getFeatureAM(); /*create a list of ScopeValueParamVORowImpl Objects */ ViewObject scopeVO = featureAM.getScopeValueParam(); /*create a list of ScopeValueParamVORowImpl Objects */ List<ScopeValueParamVORowImpl> newList = new ArrayList<ScopeValueParamVORowImpl>(); ScopeValueParamVORowImpl newRow =

	(ScopeValueParamVORowImpl)scopeVO.createRow();
	/*set each element of the object with appropriate value */
	newRow.setattributeValue(BE-TaxAttribute1);
	newRow.setscopeValue(BE-Tax);
	newRow.setscopeValueCode(BE-Tax);
	/* Add to the list */
	newList.add(newRow);
	String featureShortName = PAY_LEGISLATION;
	
	Call Method
	FeatureAMImpl featureAM = (FeatureAMImpl)<OriginalAM>.getFeatureAM();
	
	try {
	result = featureAM.freezeFeatureChoice(newList,featureShortName);
	} catch(JboException e) {
	e.printStackTrace();
	}

Using freezeFeatureChoice Web Service

Name	freezeFeatureChoice
Purpose	Locks Feature Choice settings of the specified Feature. The Choice selections cannot be updated anymore.
Definition	public void freezeFeatureChoice(List< java.util.List<oracle.apps.setupHub.remoteApp.publicModel.view.ScopeValueParamVORowImpl> scopeValueParams, String featureShortName) throws oracle.jbo.JboException
Parameters	featureShortName – Offering, Option, Feature identifier as String scopeValues – List of Scope values; they are of ScopeValueParams object type. The list of Scope values could be NULL.
Results returned	N/A
Exception condition	Invalid Feature short name (code) is used.
Sample code	**Constructing Input** /*create a list of ScopeValueParamVORowImpl Objects */ List<ScopeValueParamVORowImpl> newList = new ArrayList<ScopeValueParamVORowImpl>(); ScopeValueParamVORowImpl newRow =

	(ScopeValueParamVORowImpl)scopeVO.createRow();
	/*set each element of the object with appropriate value */
	newRow.setattributeValue(BE-TaxAttribute1);
	newRow.setscopeValue(BE-Tax);
	newRow.setscopeValueCode(BE-Tax);
	/* Add to the list */
	newList.add(newRow);
	String featureShortName = PAY_LEGISLATION;
	
	Call Method
	FeatureService svc = (FeatureService) ServiceFactory.getServiceProxy (FeatureService.NAME);
	
	try {
	result = svc.freezeFeatureChoice(newList,featureShortName);
	} catch(JboException e) {
	e.printStackTrace();
	}

Retrieving Feature Hierarchy

Feature hierarchy can be defined by associating dependent Features through Feature Choices of a parent. See Dependent Options and Features for more information.

Use getFeatureHierarchyChoices Java API to retrieve the feature hierarchy starting with a specific feature that includes feature choices for each level of the hierarchy. See Invoking Java APIs for more information on how to invoke the API.

No web service is available for this method.

Name	getFeatureHierarchyChoices
Purpose	Retrieves all subsequent levels of Features and their choices starting with the Feature specified as input parameter.
Definition	public List<FeatureTransVORowImpl> getFeatureHierarchyChoices(String featureShortName) throws JboException
Parameters	featureShortName – Offering, Option, Feature identifier as String
Results returned	A list of children Features and their choices. See (transient VO) FeatureTransVORowImpl for more details.

Exception condition	Invalid Feature short name (code) is used

Sample code	**Constructing Input**
	String featureShortName = PAY_LEGISLATION;
	
	Call Method
	FeatureAMImpl featureAM = (FeatureAMImpl)<OriginalAM>.getFeatureAM();
	
	try {
	List<FeatureTransVORowImpl> listOfFeatures = featureAM.getFeatureHierarchyChoices(PER_CORE);
	If(listOfFeatures != null) {
	for (int i=0;i<listOfFeatures.size();i++) {
	FeatureTransVORowImpl row = listOfFeatures.get(i);
	System.out.println(Feature:+row.getFeatureName()+:+row.getFeatureType()+:+row.getFeatureShortName()+ parent:+row.getParentFeature()+:+row.getFeatureLevel()+:+row.getParentFeatureShortName());
	RowIterator iter = row.getFeatureChoice();
	while(iter.hasNext()) {
	FeatureChoiceVORowImpl choiceRow = (FeatureChoiceVORowImpl)iter.next();
	System.out.println(Choice:+choiceRow.gand so onhoiceShortName()+:+choiceRow.gand so onhoiceName()+:+choiceRow.getFeatureShortName());
	}
	}
	}
	} catch(Exception e) {
	e.printStackTrace();
	}

Retrieving Associated Objects of Enabled Offerings and Options

Offerings and options are the driving decision points in configuring Oracle Fusion Applications to fit just right for the business needs in any given implementation. For a given installation, implementation managers will select or enable offerings and options that are applicable to their business needs.

Based on those selections, the following offering and option related associations could be programmatically enabled or disabled to make the Oracle Fusion application properly configured for a specific implementation.

- Oracle Transactional Business Intelligence (OTBI) reports
- Subject Areas of Oracle Business Intelligence Analytics (OBIA)
- Oracle Enterprise Manager (EM) Metrics

Determining Whether to Display an OTBI Report

Use EL expression to determine if an OTBI report will be displayed in the Oracle Fusion Applications based on whether or not any of the offerings with which it is associated is in the Implemented status. Use the following steps:

1. Define a managed bean called oracle.apps.setup.commonSetup.features.Offerings with the variable called OFFERINGS at REQUESTSCOPE LEVEL in adfc-config.xml file.
2. Refer to showReport property by using: #{Offerings.showReport.REPORT_NAME}, where REPORT_NAME is the name of the report whose showReport property you want to know.

If FSM finds that the specified REPORT_NAME is associated with at least one offering that has implementation status set to Implemented, then showReport returns Y. Otherwise, showReport returns N.

Retrieving OBIA Subject Areas Associated with Enabled Offerings and Options

Use getEnabledSubjectAreas Java API to retrieve a list of OBI subject areas that are associated with enabled offerings and options. An offering or option is considered enabled when implementation managers set its Enabled for Implementation flag.

No web services are available for this method.

Name	getEnabledSubjectAreas
Purpose	Retrieves all BI subject areas that are associated with enabled Offerings and Options.
Definition	public List<SubjectAreaVORowImpl> getEnabledSubjectAreas() throws JboException
Parameters	N/A
Results returned	A list of BI Subject Areas.

	See (transient VO) SubjectAreaVORowImpl for more details.
Exception condition	No match found and a NULL list is returned.

Sample code	**Call Method**
	FeatureAMImpl featureAM = (FeatureAMImpl)<OriginalAM>.getFeatureAM();
	try {
	List<SubjectAreaVORowImpl> subjectAreaList= featureAM.getEnabledSubjectAreas();
	If(subjectAreaList!= null) {
	for (int i=0;i< subjectAreaList.size();i++) {
	SubjectAreaVORowImpl row = subjectAreaList.get(i);
	System.out.println(Subject Area:+row.getSubjectAreaName()+:+row.getSubjectAreaShortName()+:+row.getSubjectAreaDesc()+:+row.getOfferingOptionShortName()+:+row.getOfferingOptionName());
	}
	}
	} catch(Exception e) {
	e.printStackTrace();
	}

Retrieving EM Metrics Associated with Enabled Offerings and Options

Use getEnabledEMMetrics Java API to retrieve a list of EM Metrics that are associated with enabled offerings and options. An Offering or option is considered enabled when implementation managers set its Enabled for Implementation flag.

No web services are available for this method.

Name	getEnabledEMMetrics
Purpose	Retrieves all EM Metrics that are associated with enabled Offerings and Options.
Definition	public List<EmMetricVORowImpl> getEnabledEMMetrics() throws JboException
Parameters	N/A
Results returned	A list of EM Metrics. See (transient VO) EmMetricVORowImpl for more details.

Exception condition	No match found and a NULL list is returned.

Sample code	**Call Method**
	FeatureAMImpl featureAM = (FeatureAMImpl)<OriginalAM>.getFeatureAM();
	
	try {
	List<EmMetricVORowImpl> emMetricList= featureAM.getEnabledEMMetrics();
	If(emMetricList!= null) {
	for (int i=0;i< emMetricList.size();i++) {
	EmMetricVORowImpl row = emMetricList.get(i);
	System.out.println(Em Metric:+row.getEmMetricName()+:+row.getEmMetricShortName()+:+row.getEmMetricDesc()+:+row.getOfferingOptionShortName()+:+row.getOfferingOptionName()+: + row.getModuleShortName()+:+ row.getServiceName()+:+row.getEndPointURL());
	}
	}
	
	} catch(Exception e) {
	e.printStackTrace();
	}

View Objects

The following transient view objects (VO) are used by feature-related APIs and web services.

ScopeValueParamVORowImpl

Method	Description
String getattributeValue()	Gets value for the calculated attribute attributeValue
setattributeValue(String value)	Sets value of the calculated attribute attributeValue
String getscopeValue()	Gets value for the calculated attribute scopeValue
setscopeValue(String value)	Sets value of the calculated attribute scopeValue
String getscopeValueCode()	Gets value for the calculated attribute scopeValueCode
setscopeValueCode(String value)	Sets value of the calculated attribute scopeValueCode

FeatureTransVORowImpl

Method	Description
String getFeatureShortName()	Returns feature short name (code)
void setFeatureShortName(String value)	Sets feature short name (code)
String getFeatureName()	Returns feature name
void setFeatureName(String value)	Sets feature name
Long getFeatureLevel()	Returns level of the feature in the hierarchy
void setFeatureLevel(Long value)	Sets feature level
String getFeatureType()	Returns type of feature, such as option or feature
void setFeatureType(String value)	Sets feature type
String getParentFeature()	Returns parent feature name. If the input is the root node, the null is returned
void setParentFeature(String value)	Sets parent feature value
void setParentFeatureShortName(String value)	Sets parent feature short name (code)
String getParentFeatureShortName()	Gets parent feature short name (code)
RowIterator getFeatureChoice()	Returns feature choices of the feature

FeatureChoiceVORowImpl

Method	Description
String getChoiceName()	Returns feature choice name
void setChoiceName(String value)	Sets feature choice name
String getChoiceShortName()	Returns feature choice short name (code)
void setChoiceShortName(String value)	Sets feature choice short name (code)
String getFeatureShortName()	Returns feature short name (code)
void setFeatureShortName(String value)	Sets feature short name (code)

SubjectAreaVORowImpl

Method	Description
String getSubjectAreaName()	Returns BI subject area name
String getSubjectAreaShortName()	Returns BI subject area name
String getSubjectAreaDesc()	Returns BI subject area description
String getOfferingOptionShortName()	Returns offering or option short name (code)

String getOfferingOptionName()	Returns offering or option name

EmMetricVORowImpl

Method	Description
String getEmMetricName()	Returns EM metric name
String getEmMetricShortName()	Returns EM metric short name (code)
String getEmMetricDesc()	Returns EM metric description
String getModuleShortName()	Returns module short name (code)
String getServiceName()	Returns service name
String getOfferingOptionShortName()	Returns offering or option short name (code)
String getOfferingOptionName()	Returns offering or option name
String getEndPointURL()	Returns end-point URL of module

Task Status Integration

FSM provides public web services that allow to set or to retrieve setting of status of a task in an implementation project.

Retrieving Current Status of Specific Task in a Specific Implementation Project

Use getTaskStatus web service to retrieve the information. No Java API is available for this method.

Name	getTaskStatus
Purpose	To retrieve status of a task in an implementation project
Definition	public java.lang.String getTaskStatus(java.lang.String iPTLItemKey)
Parameters	iPTLItemKey – The identifier of a task in an implementation project as a String.
Results returned	The status of the specified iPTLItemKey (input parameter). The return value could be NULL if no match is found for the input parameter.
Exception condition	N/A
Sample code	String id = 100000010087002; String status = getStatus(id);

Setting Status of Specific Task in a Specific Implementation Project

Use setTaskStatus web service to set task status. No Java API is available for this method.

Name	setTaskStatus
Purpose	To set status of a task in an implementation project
Definition	public void setTaskStatus (java.lang.String iPTLItemKey, java.lang.String status)
Parameters	iPTLItemKey – The identifier of a task in an implementation project as a String. Status – Status to be set for the specified iPTLItemKey. Valid values of status are: NOT_STARTED IN_PROGRESS DONE SUBMITTED COMPLETED_WITH_WARNINGS COMPLETED_WITH_ERRORS EXECUTE_ONLY_ONCE If **VALID STATUS VALUE** is not specified task Status of the specified iPTLItemKey will **NOT BE SET**.
Results returned	N/A
Exception condition	N/A
Sample code	String id = 100000010087002; setTaskStatus(id, IN_PROGRESS);

Retrieving Identifier of a Task in an Implementation Project

Since a task can be part of many implementation projects, iPTLItemKey, the key to uniquely identify a specific instance of a task in a specific implementation project, is required to set or retrieve task status.

The key, iPTLItemKey, of a setup task is always passed as a parameter when you perform the tasks from FSM. To retrieve the value, use the following approaches as appropriate:

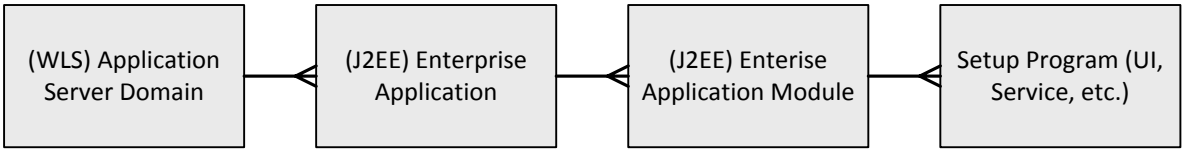
- **In UI:** See Reading Parameter Passed by FSM
- **In Web Service:** The parameter is passed via processControl of launchProcess() API. See launchProcess() Details.

10

Setup Program Deployment Topology

About Setup Program Deployment Topology

Application developers must define the deployment topology, meaning the application server domains, J2EE applications and modules, for J2EE components of setup task execution program (UI or web service) and each export and import web service (collectively referred to as setup programs).



Later, after Oracle Fusion offerings are provisioned (installed), the endpoint URLs (the actual location) of the application server domains and enterprise applications will be assigned.

FSM will use the deployment topology including the actual deployed location information to invoke the setup programs when you perform a setup task or export and import setup data.

Note

Deployment topology must be specified for every UI and web service expected to be executed from FSM. See Task Program Information and Defining Export and Import Services for more information. However, in most cases, existing domains, J2EE applications and modules may be used and application developers do not need to define and register new components.

Warning

Detailed information on the following areas is beyond the scope of this document. Consult appropriate documentation and training material for more information.

Application Server (in general) and Weblogic Server concepts.

Standards and guidelines for defining deployment topology (domain, enterprise applications, modules, and so on) of Oracle Fusion offerings.

J2EE Modules

Each setup program must be associated with a J2EE module. A module represents the basic unit of an application and consists of one or more J2EE components for the same container type and one component deployment descriptor of that type. A component deployment descriptor contains declarative data to customize the components in the module. A J2EE module without an application deployment descriptor can be deployed as a stand-alone J2EE module.

Categories

There are three categories of J2EE modules:

- **Enterprise Javabeans (EJB) Modules:** These contain class files for enterprise beans and an EJB deployment descriptor. EJB modules are packaged as JAR files with .JAR (Java Archive) extension. Application client modules are also packaged as JAR files.
- **Web Modules:** These contain JSP files, class files for servlets, GIF and HTML files, and a Web deployment descriptor. Web modules are packaged as JAR files with .WAR (Web Archive) extension.
- **Resource Adapter Modules:** These contain all Java interfaces, classes, native libraries, and other documentation, along with the resource adapter deployment descriptor. Together, these implement the connector architecture for a particular EIS. Resource adapter modules are packages as JAR files with a .RAR (Resource Adapter Archive) extension.

Module Types

Each Oracle Fusion Applications module may support J2EE components of a specific type of program, which is specified using a module type. They are:

- UI
- Web service
- SOA

(J2EE) Enterprise Applications

An enterprise application is a deployable unit of J2EE functionality. It can contain a single J2EE module or a group of modules packaged into an EAR file along with a Java EE application deployment descriptor. Its primary goals are to provide scalable modular assembly and portable deployment of various J2EE applications into a J2EE product.

Application Server Domain

An Oracle WebLogic Server domain is a logically related group of Java components. A domain typically consists of one or more WebLogic Server instances that are broadly classified as:

- **Administration Server:** A special Oracle WebLogic Server instances used to configure and manage all resources within the domain. Each domain will have one administration server.
- **Managed Server:** A collection of Oracle WebLogic Server instances where Java components, such as Web applications, Enterprise Java Beans (EJB), Web Services, and other resources, are deployed. A domain will typically have multiple managed servers.

Note

Setup programs will be deployed on the managed servers of their specified domains.

Oracle Fusion Application Pillars – Database Deployment Configurations

Oracle Fusion Applications use the concept of pillars to determine how the databases used by the (J2EE) enterprise applications can be implemented.

A pillar is a logical grouping of Oracle Fusion enterprise applications that will use the same database instance. Enterprise applications from different pillars, however, can be installed to use either of the following:

- Single database instance known As Global Single Instance (GSI) configuration.
- Different database instances, in which case cross-pillar Producer-Consumer Processing and Data Replication models are used to keep the data synchronized across pillars.

The pillar model is designed to support completely independent management of individual database instances of each pillar including patching and upgrade processes.

Supported Pillars

Currently Oracle Fusion Applications support the following three pillars:

- Financials and Supply Chain Management (FSCM)
- Human Capital Management (HCM)
- Customer Relationship Management (CRM)

In addition, a pillar called Migration is supported and used only migration process of moving to Oracle Fusion Applications from other Oracle product lines such as Oracle Enterprise Business Suites (EBS), Oracle PeopleSoft, Oracle JD Edwards, Oracle Siebel, and so on.

Enterprise Applications in Multiple Pillars

Some Oracle Fusion Applications may be designed to be deployed in multiple pillars. For example, Oracle Fusion Payables can be associated with both FSCM and HCM pillars.

Warning

Currently Oracle Fusion Applications have implemented PILLARS in very limited scope. The following restrictions apply at this time:

Only Global Single Instance (GSI) model is supported.

An enterprise application can belong to only one pillar.

Only the three predefined pillars are supported. That is, creating new pillars or editing and deleting any of the existing pillars is not supported.

Viewing Lists of Domains

1. Go to the Setup and Maintenance work area.
2. Click Manage Domains from the task pane, which will display a page with the same name. This page will show a list of domains based on your default search setting. If no personalization was performed, the default search setting of this page is to display all domains.
3. Modify search results if desired:

- **Ad hoc search:** In the Search region enter the appropriate search criteria, and click Search.
- **Saved search:** Select the desired saved search from the Saved Search field.

Editing Domains

1. From the list of domains on the Manage Domains page, perform one of the following actions to open the Edit Domain page:
    - Click the domain name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Actions menu on the table task bar.
2. Edit the domain name or description.
3. Click Save or Save and Close to save changes.

See also:

About Setup Program Deployment Topology

Note

You can view Oracle-delivered domains, but cannot edit or delete them.

Creating Domains

1. From the list of domains on the Manage Domains page, perform one of the following actions to open the Create Domain page:
    - Click Create on the table task bar.
    - Click Create from the Actions menu on the table task bar.
2. Enter the name and description for the new domain.
3. Click Save and Close to save changes.

Warning

Once created, a domain cannot be deleted.

Viewing Lists of Enterprise Applications

1. Go to the Setup and Maintenance work area.
2. Click Manage Enterprise Applications from the task pane, which will display a page with the same name. This page will show a list of enterprise applications based on your default search setting. If no personalization was performed, the default search setting of this page is to display all enterprise applications.
3. Modify search results if desired:
    - **Ad hoc search:** In the Search region enter the appropriate search criteria, and click Search.
    - **Saved search:** Select the desired saved search from the Saved Search field.
4. From this page, you can:
    - View and edit enterprise application details
    - Register new enterprise applications

Viewing and Editing Enterprise Application Details

1. From the list of enterprise applications on the Manage Enterprise Applications page, perform one of the following actions to open the Edit Enterprise Application page:
    - Click the enterprise application name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Actions menu on the table task bar.
2. View or edit enterprise application details.
3. Click Save and Close to save changes.

Note

You can view Oracle-delivered enterprise applications, but cannot edit or delete them.

Registering New Enterprise Applications

1. From the list of enterprise applications on the Manage Enterprise Applications page, perform one of the following actions to open the Register Enterprise Application page:
    - Click Register Enterprise Application on the table task bar.
    - Click Register Enterprise Application from the Actions menu on the table task bar.
2. Enter enterprise application details.
3. Click Save and Close to save changes.

Warning

Once registered, an enterprise application cannot be deleted.

Enterprise Application Details

When creating a new enterprise application or editing an existing enterprise application, enter or modify the following fields.

Field	Field Type	Required	Description
Name	Text	Yes	Enter the name of enterprise application that you want to register
Code	Text	Yes	Use a unique code to identify the enterprise application. Once created, code cannot be changed.
Domain	List of Values	Yes	Select the name of the domain to be used by this enterprise application.
Default URL	Text	No	If the enterprise application will always deployed at a constant and static location then the URL of that location
Source File	Text	No	EAR file name
Pillar	Shuttle Box	Yes	**Create Association:** Select desired row in Available Pillars (shuttle box) and click on move to right button to move it to Selected Pillars (shuttle box) **Remove Association:** Select desired row in Selected Pillars (shuttle box) and click on move to left button to move it to Available Pillars (shuttle box).

Warning

Do not associate more than one pillar with any enterprise application.

Viewing Lists of Modules

1. Go to the Setup and Maintenance work area.
2. Click Manage Modules from the task pane, which will display a page with the same name. This page will show a list of modules based on your default search setting. If no personalization was performed, the default search setting of this page is to display all modules.
3. Modify search results if desired:
    - **Ad hoc search:** In the Search region enter the appropriate search criteria, and click Search.
    - **Saved search:** Select the desired saved search from the Saved Search field.
4. From this page, you can:
    - View and edit modules
    - Register new modules

Viewing and Editing Module Details

1. From the list of modules on the Manage Modules page, perform one of the following actions to open the Edit Module page:
    - Click a module name.
    - Select the desired row and click Edit on the table task bar.
    - Select the desired row and click Edit from the Actions menu on the table task bar.
2. View or edit module details.
3. Click Save and Close to save changes.

Note

You can view Oracle-delivered modules, but cannot edit or delete them.

Registering New Modules

1. From the list of modules on the Manage Modules page, perform one of the following actions to open the Register Module window:
    - Click Register Module on the table task bar.

- Click Register Module from the Actions menu on the table task bar.

2. Enter module details.
3. Click Save and Close to save changes.

Warning

Once registered, a module cannot be deleted.

Module Details

When creating a new enterprise application or editing an existing enterprise application, enter or modify the following fields.

Field	Field Type	Required	Description
Name	Text	Yes	Enter the name of module that you want to register
Code	Text	Yes	Use a unique code to identify the module. Once created, code cannot be changed.
Description	Text	No	Although not required, provide a brief, meaningful description that will help you understand the purpose of the module.
Enterprise Application	List of Values	Yes	Select and associate the enterprise application to which this module belongs
Type	List of Values	Yes	Indicate what type of module it is by selecting one of the valid values
Context Root	Text	Yes	Enter the context root of this module.