Oracle® Functional Testing

Flow Builder User's Guide

Release 12.4.0.1

E48651-04

March 2014

ORACLE®

Oracle Functional Testing Flow Builder User's Guide Release 12.4.0.1

E48651-04

Primary Author: Rick Santos

Contributing Author: Abhishek Kumar Choudhary, Prasanti Madireddi, Yamala Smita

Contributor: Drupad Panchal , Srinivas Potnuru

Contents

8 Using Reports

9 Administering Oracle Flow Builder

List of Figures

List of Tables

Preface

Welcome to the Oracle Flow Builder User's Guide. This guide explains how to use the features and options of Oracle Flow Builder for creating test flows and generating OpenScript scripts for testing Web/Oracle E-Business Suite applications.

Oracle Flow Builder is licensed as a feature in Oracle Functional Testing Suite for Oracle Applications.

Audience

This document is intended for Business analysts, QA engineers, and test engineers who will be developing Oracle Flow Builder components and flows for testing a Web site or application. The guide does require an understanding of software or Web/Oracle E-Business Suite application testing concepts. Test engineers using Oracle Flow Builder should be familiar with the concepts of regression testing, load testing, and scalability testing.

The keyword paradigm of Oracle Flow Builder does not require any programming experience to develop scripts for testing. However, the advanced programming features available in Oracle OpenScript scripts do require experience with the Java programming language.

Using This Guide

This guide is organized as follows:

Chapter 1, "Introduction" introduces Oracle Flow Builder and provides an overview of the features and user interface.

Chapter 2, "Setting Up Oracle Flow Builder" explains the requirements, installation, and initial setup of the Oracle Flow Builder application.

Chapter 3, "Defining Components" explains how to add and update components in the Component Tree.

Chapter 4, "Defining Components Sets" explains how to add and update components sets in the Component Set Tree.

Chapter 5, "Defining Flows" explains how to add and update components sets in the Component Set Tree.

Chapter 6, "Using Notifications" explains how to use the Notifications options in the Oracle Flow Builder application.

Chapter 7, "Using History" explains how to use the History options in the Oracle Flow Builder application.

Chapter 8, "Using Reports" explains how to use the Report options in the Oracle Flow Builder application.

Chapter 9, "Administering Oracle Flow Builder" explains how to perform administrative tasks within the Oracle Flow Builder application.

Appendix A, "Keyword Reference" lists the keywords and objects used to specify component code in Oracle Flow Builder.

Appendix B, "Function Library Reference" lists the functions in the default function libraries installed with the Oracle Flow Builder application.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to 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.

Related Documents

For more information, see the following documents in the Oracle Application Testing Suite documentation set:

- *Oracle Application Testing Suite Release Notes*
- *Oracle Application Testing Suite Getting Started Guide*
- *Oracle Functional Testing OpenScript User's Guide*
- *Oracle Functional Testing OpenScript Programmer's Reference*
- *Oracle Functional Testing Flow Builder Starter Pack Reference Guide for E-Business Suite Release 12.1.3*
- *Oracle Load Testing Load Testing User's Guide*
- *Oracle Load Testing Load Testing ServerStats Guide*
- *Oracle Test Manager Test Manager User's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

Oracle Flow Builder (OFB) is a Keyword-driven testing application that business analysts and QA engineers use to build business test automation flows. The test automation flows can be translated into executable OpenScript scripts. Technical QA engineers define and implement keywords for the Web/Oracle E-Business Suite application's components. Business analysts and QA engineers then connect components together to define a larger business process, or "flow" and generate OpenScript scripts to automate testing of the application.

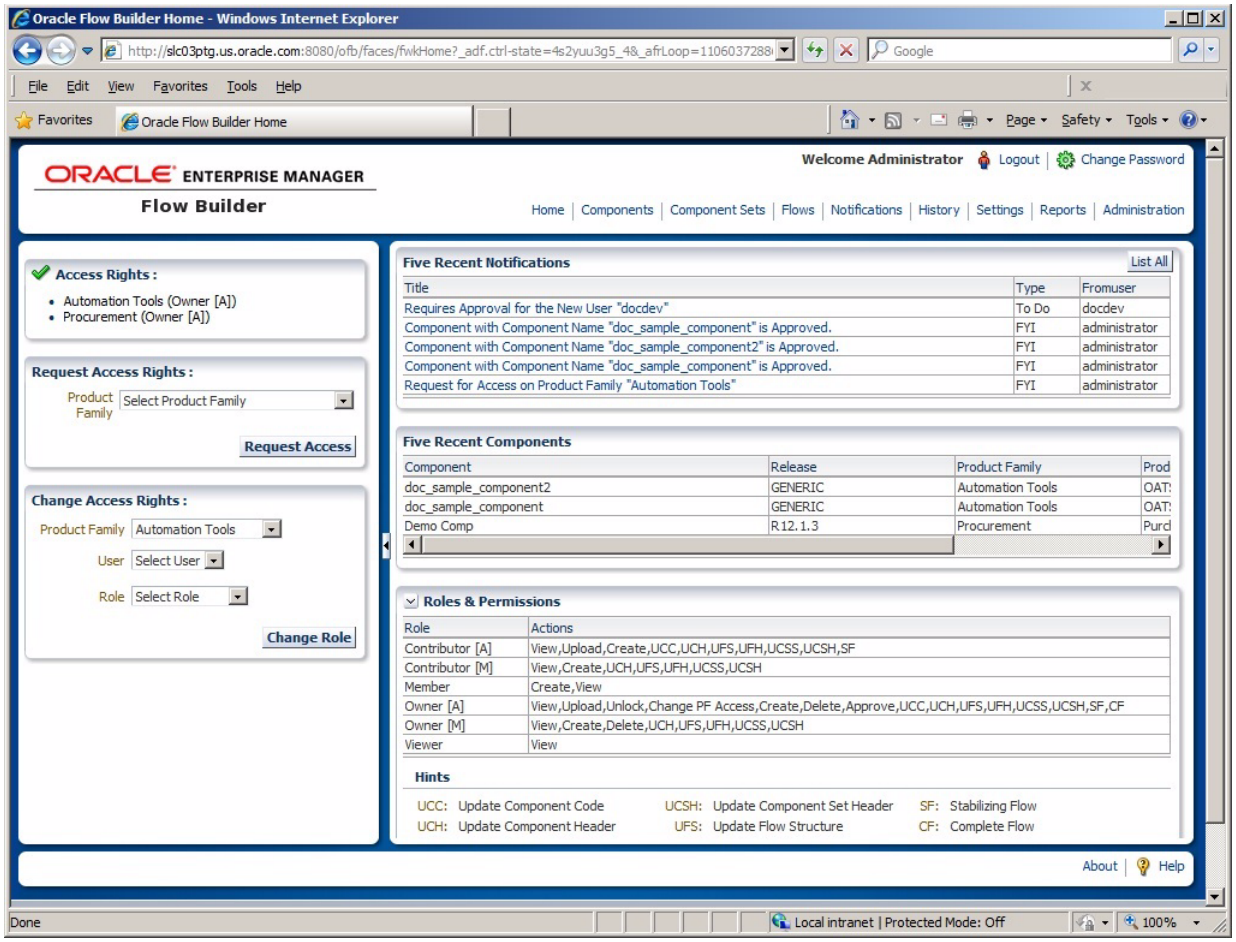
This chapter introduces the Oracle Flow Builder application and provides an overview of the features available. It contains the following sections:

1.1 About Oracle Flow Builder

The Oracle Flow Builder application is a keyword-driven component based testing framework for testing Oracle E-Business Suite applications. Oracle Flow Builder starter kit includes 2100+ components and 200 flows for testing Oracle E-Business Suite.

The application consists of the following features:

Figure 1–1 Main Window Home Page



The Oracle Flow Builder application consists of the following pages:

- **Home**: shows the logged in users' Access Rights with options for requesting changes and lists recent notifications and components. Roles and permissions information is also provided.
- **Components**: shows the currently available component tree and provides options for adding new components and defining the keywords and parameters that define the component code.
- **Component Sets**: shows the currently available component sets tree and provides options for adding new component sets that link multiple individual components that are used together frequently in test flows.
- **Flows**: shows the currently available flows tree and provides options for adding new flows that define the sequence of components, component sets, and test data to use to generate test automation scripts.
- **Notifications**: shows the informational and To Do messages generated during use of the Oracle Flow Builder application.
- **History**: provides options for searching the history of components, component sets, flows, and users.
- **Settings**: provides options for specifying the email notification preferences.

- **Reports**: provide options for searching and viewing reports for components, component sets, and flows.
- **Administration**: provides options for setting up the release and product structure within the application, setting up the Email server, managing function libraries, managing approvals, and managing product family access.

1.2 Basic Processes

This section describes the main steps involved with using the application after it is installed and the initial setup has been performed.

- An administrator defines the Releases, Product Families, Products, and Features in the Oracle Flow Builder application.
- An administrator sets up user roles and assigns privileges.
- Users request registration from the login page. Upon approval of the user registration, the user will be able to log in and request product family access rights.
- Component developers define components and component code (keywords and parameters) in the component library. Component developers must be very familiar with the application to be tested in order to define components and component code (keywords, objects, and parameters).
- A designated approver approves or rejects component code created by component developers. Designated approvers review the keywords and parameters defined in the component code for accuracy and completeness and approve or reject the code.
- After approval, components become available to be added to component sets and flows.
- Component developers define component sets (groups of components in a set or sequence that can be added to a flow as a group).
- Flow developers create flows by adding components and/or component sets and defining test data for each component in the flow. Flow developers must be familiar with the application to be tested in order to select components and component sets for the flow and define the test data for each component.
- Once a flow is finalized, flow developers set the flow to "assembled" status.
- Flow developers or other testers generate OpenScript code to an OpenScript zip file. The OpenScript zip file can be used in OpenScript to perform functional tests against the application under test.

2

Setting Up Oracle Flow Builder

This chapter explains how to get started using Oracle Flow Builder. This chapter contains the following sections:

2.1 Getting Started

The following table provides an overview of the Oracle Flow Builder application setup tasks.

Table 2–1 Getting Started with Oracle Flow Builder

Step	Task	Role
1	Verify the system requirements. See Section 2.2, "System Requirements"	Administrator
2	Verify the Prerequisites have been meet. See Section 2.3, "Prerequisites"	Administrator
3	Install the application. See Section 2.4, "Installing Oracle Flow Builder"	Administrator
4	Perform the initial set up. See Section 2.5, "Initial Setup for Administrators"	Administrator
5	Perform the initial set up. See Section 2.6, "Initial Setup for Users"	Users

2.2 System Requirements

Oracle Flow Builder has the following system requirements:

- OS: Oracle Enterprise Linux version 5.
- CPU: P4 or higher @ 2GHz (Intel Dual core recommended, Intel Xeon Quad core preferred).

- Memory: 4GB or higher (8GB preferred).
- Free Disk space: 20 GB (50 GB recommended, 200 GB preferred).
- Browser: Firefox 17 ESR.

2.3 Prerequisites

Oracle Flow Builder installer has the following prerequisites:

- An installed and functioning Oracle Database 11*g* Enterprise Edition (v. 11.2.0.3.0).
- Oracle Enterprise Linux version 5.

2.4 Installing Oracle Flow Builder

This section describes the basic installation procedures required for setting up the database and installing the application.

Oracle Flow Builder requires a functioning database which:

- Is running Oracle Database 11*g* Enterprise Edition (v. 11.2.0.3.0), which can be accessed from:

  `https://updates.oracle.com/Orion/PatchDetails/process_form?patch_num=10404530`

- Can be accessed from a remote machine (i.e., connect using the machine *hostname* instead of using `localhost`).

> **Note:** This section provides basic summary steps for installing Oracle Database 11*g* Enterprise Edition and remote access configuration for use with the Oracle Flow Builder application. These steps are not intended to provide complete database installation instructions. For detailed database installation instructions, see the "Installing Oracle Database" chapter of the *Oracle Database Installation Guide 11g Release 2 (11.2) for Linux* at the following location:
>
> `http://docs.oracle.com/cd/E11882_01/install.112/e47689/toc.htm`

2.4.1 Installing Oracle Database 11*g* Enterprise Edition

The Oracle Flow Builder application requires an installed and functioning Oracle Database 11*g* Enterprise Edition (v. 11.2.0.3.0). If necessary, download and install the database prior to installing the Oracle Flow Builder application.

To download and install Oracle Database 11*g* Enterprise Edition (v. 11.2.0.3.0):

1. Download the seven zip files from:

   `https://updates.oracle.com/Orion/PatchDetails/process_form?patch_num=10404530`

2. Extract all seven zip files: `p10404530_112030_Linux-x86-64_1of7` through `p10404530_112030_Linux-x86-64_7of7`.
3. Run `./database/runInstaller`.
4. Clear the **I wish to receive security updates** checkbox and click **Next**.
5. Click **Skip software updates**.
6. Select **Create and configure a database** and click **Next**.

7. Select **Desktop Class** and click **Next**.
8. Specify the Install Configuration as follows:
   - Oracle base: `/scratch/<username>/oracle/oee/app`
   - Software location: `/scratch/<username>/oracle/oee/app /product/11.2.0/dbhome_1` (auto populated).
   - Database file location: `/scratch/<username>/oracle/oee/app /oradata` (auto populated).
   - Database edition: `Enterprise Edition (4.5GB)`.
   - Character Set: `Unicode (AL32UTF8)`.
   - OSDBA Group: `dba`.
   - Global database name: `orcl.us.oracle.com`.
   - Administrative password: `oracle123`.
9. Click **Yes** on the "Admin password entered does not conform to the Oracle standards" dialog box.
10. On Prerequisite Checks, if there are errors, click **Ignore All** and click **Next**.
11. Click **Yes** on the "You have chosen to ignore some of the prerequisites" dialog box.
12. On the Summary page, confirm your entries and click **Install**.
13. If installing the database for the first time, execute the `orainstRoot.sh` script when prompted by the installer then click **OK**, as follows:

    ```
    > sudo <Path to oraInventory>/orainstRoot.sh
    ```

    for example:

    ```
    > sudo /scratch/${USER}/oracle/oee/oraInventory/orainstRoot.sh
    ```

14. Execute the `root.sh` script when prompted by the installer then click **OK** to complete the installation, as follows:

    ```
    > sudo /scratch/<username>/oracle/oee/app/product/11.2.0/dbhome_1/root.sh
    ```

    During the `root.sh` file execution, if you get '`/usr/local/bin is read only`', then continue without copy.

2.4.2 Configuring Oracle Database for Remote Access

The Oracle Database must be configure for remote access to be used with the Oracle Flow Builder application.

To configure Oracle Database for remote access:

1. Modify `tnsnames.ora` and change `localhost` to the fully qualified domain name (FQDN) of the machine where the Database is installed. For example:

   ```
   > sed -i -e 's/localhost/machineName.company.com/g' $ORACLE_
   HOME/network/admin/tnsnames.ora
   ```

   Note: If you receive an "Undefined variable" error, verify the `$ORACLE_HOME` environment variable is set or manually enter the full path to Oracle's home directory in place of `$ORACLE_HOME`.

2. Modify `listener.ora` and change `localhost` to the fully qualified domain name (FQDN) of the machine where the Database is installed. For example:

```
> sed -i -e 's/localhost/machineName.company.com/g' $ORACLE_
HOME/network/admin/listener.ora
```

3. Stop the Listener and then Start it back up so that the new changes are in effect, as follows:

```
> $ORACLE_HOME/bin/lsnrctl stop
> $ORACLE_HOME/bin/lsnrctl start
```

4. Restart the database (optional), as follows:

```
$> sqlplus
$> username: SYS AS SYSDBA
$> password: oracle123
$> shutdown immediate
$> startup
```

2.4.3 Installing the Oracle Flow Builder Application

The Oracle Flow Builder application is installed using the setup script included in the downloaded application zip file.

To install the Oracle Flow Builder application:

1. Download the Oracle Flow Builder zip file from Oracle Technology Network (OTN):

```
http://www.oracle.com/technetwork/oem/downloads/index-084446.html
```

2. Extract the Oracle Flow Builder Zip, as follows:

```
> unzip ./OFB_MAIN_GENERIC_xxxxxx.xxxx.S.zip -d /tmp/OFB_MAIN
```

3. Add execute permission to setup.sh, as follows:

```
> chmod 744 /tmp/OFB_MAIN/setup.sh
```

> **Note:** You may need to add execute permission to other `.sh` script files also if you receive a "Permission denied" message when running the script.

4. Start the Oracle Flow Builder setup, as follows:

```
> /tmp/OFB_MAIN/setup.sh install
```

5. Enter the configuration information as prompted by the setup script. For example:
   - Enter the Oracle Flow Builder installation directory: `/scratch/username/oracle/OracleOFB`
   - Enter the Administrator Password: `oracle123` (or any other password you wish to use as the default administrator account password).
   - Enter the Oracle Flow Builder Server host-name (press enter to accept [auto-detected-hostname]): *machineName*`.company.com`
   - Enter Database port (press enter to accept [1521]): `1521`.
   - Enter database SID (press enter to accept [orcl]): `orcl`.

- Enter database admin user name: `system`.
- Enter database admin Password: `oracle123`.

After a successful installation, the Oracle Flow Builder will be left in a running state.

6. Open a browser and start the Oracle Flow Builder application using the URL:

   `http://machineName.company.com:9090`

7. Log in to the application using the default administrator username and password (administrator/password-entered-during-product-installation is the default).

   See the Section 2.5, "Initial Setup for Administrators" section for initial setup tasks for the Oracle Flow Builder application. See the Section 2.4.4, "Oracle Flow Builder Server Maintenance" section for additional server options.

2.4.4 Oracle Flow Builder Server Maintenance

The following maintenance functions are available to start, stop, or check the running status of an Oracle Flow Builder application.

- Starting the Oracle Flow Builder Application Server

  Run `<ofb-install-dir>/scripts/control.sh start`. For example:

  ```
  > /scratch/username/oracle/ofb/scripts/control.sh start
  ```

- Stopping the Oracle Flow Builder Application Server

  Run `<ofb-install-dir>/scripts/control.sh stop`. For example:

  ```
  > /scratch/username/oracle/ofb/scripts/control.sh stop
  ```

- Getting the Oracle Flow Builder Application Server running status

  Run `<ofb-install-dir>/scripts/control.sh status`. For example:

  ```
  > /scratch/username/oracle/ofb/scripts/control.sh status
  ```

2.4.5 Deinstalling the Oracle Flow Builder Application

The Oracle Flow Builder application can be completely removed by running the deinstall script located under the scripts folder.

To deinstall the Oracle Flow Builder application:

1. Run `<ofb-install-dir>/scripts/deinstall.sh`. For example:

   ```
   > /scratch/username/oracle/ofb/scripts/deinstall.sh
   ```

Deinstalling Oracle Flow Builder only deletes the application files, any user data created on the file system and in the database will not be deleted by the deinstaller.

To remove user files and data:

> **Caution:** The following steps remove user data from the machine. Be sure you wish to do this before proceeding.

- User created files on the file system can be removed by simply deleting the product installation folder.

- User created data in the database can be removed by running the following query as a database admin user:

```
> Drop user OFB cascade;
```

2.5 Initial Setup for Administrators

This section explains the initial setup procedures for a newly installed Oracle Flow Builder. See Chapter 9, "Administering Oracle Flow Builder" for additional information about the administrator tasks and procedures.

Table 2–2 Initial Setup Tasks

Step	Task	Role
1	Set up the Email Server. See Section 9.2.8, "Setting Up Email"	Administrator
2	Review and customize Releases. See Section 9.2.1, "Setting Up Releases"	Administrator
3	Review and customize Product Families. See Section 9.2.2, "Setting Up Product Families"	Administrator
4	Review and customize Products. See Section 9.2.3, "Setting Up Products"	Administrator
5	Review and customize product Features. See Section 9.2.4, "Setting Up Features"	Administrator
6	Review and customize user roles. See Section 9.2.5, "Setting Up Roles"	Administrator
7	Add users and specify application roles. Note: Users can also request registration from the login screen. An administrator or other user with an "approver" role would then approve the user registration request. See Section 2.6, "Initial Setup for Users" and Section 9.2.6, "Setting Up Users"	Administrator/users
8	Optional. Review and customize function libraries. See Section 9.2.7, "Setting Up Function Libraries"	Administrator

2.6 Initial Setup for Users

This section explains the basic steps to get started using the Oracle Flow Builder application. The following are the basic steps new users must do to get started using the application:

- Starting the Application
- Registering User Credentials
- Changing Passwords
- Requesting Product Family Access

2.6.1 Starting the Application

To start the Oracle Flow Builder application:

1. Launch a Web browser.

2. Enter the URL:

   `http://<machineName>:9090`

   Where *machineName* is the name of the machine where Oracle Flow Builder is installed, such as *machine.company.com*.

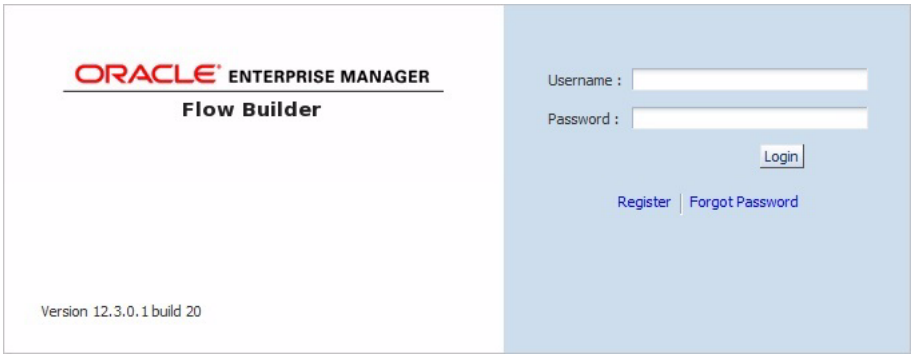
2.6.2 Registering User Credentials

New users can request access to the application by registering a user name and password at the main log in screen. A request will be sent to the specified approver to authorize the user registration and login credentials.

To register user credentials:

1. Start the application to get to the log in screen.

Figure 2–1 Login Screen



2. Click the **Register** link.

Figure 2–2 Register User Options Screen

3. Enter the user details, select the Approver, and click **Register**. The User Name must be between 4 and 8 characters. The Confirmation message appears upon successful registration. Once the Approver approves the request, the user can log into the application from the main login screen.

2.6.3 Changing Passwords

Users who are added to the application by an administrator rather than through the Register screen will receive an initial system-generated password via Email to log in to the application. You can change the system-generated password to a new password.

To change a user password:

1. Log in to the application using the username and system-generated password provided.
2. Click the **Change Password** link at the top of the page.
3. Enter a new password.
4. Confirm the new password.
5. Click **OK**.

2.6.4 Requesting Product Family Access

First time users must request access rights to specific product families defined within the application.

To request access rights to a product family:

1. Start the application and login with your user credentials.
2. If necessary click the **Home** link at the top of the page.
3. Select a product family from the **Request Access Rights** list.

Figure 2–3 Request Access Rights Options

4. Click **Request Access**. An Access Request notice appears indicating that the request has been sent to the designated approver.

Figure 2–4 Request Access Rights Confirmation Notice

5. Click **OK**.
6. Repeat steps 3 through 5 to request access to additional product families. You can request access to more than one product family available in the application. However, only one can be requested at a time.

7. Log out of the application until notified by the designated approver that the access rights have been approved. After the designated approver approves the access rights, the Access Rights list shows the Product Families with the Application role assigned to the user after the next log in.

Figure 2–5 Approved Access Rights to Product Families List

3

Defining Components

This chapter explains how to add and update components in the Component Tree. This chapter contains the following sections:

3.1 Overview

The Component Tree represents a library of defined components which can then be added to Component Sets or Flows that specify a test instance. A Component in the Component Tree contains a set of lines of code, which will perform a unique functional action at a given point of time. The lines of code are specified in the component using keywords and parameters that define how to test a particular component in the application under test. A Component should be a single entity. It cannot contain multiple complex dependent functionality.

Components have the following general types:

- **Functional Components**: Components that perform application specific functionality. Validations may or may not be part of these components based on the developed component's requirement.
- **Validation Components**: Components that perform comparison of expected and actual values. These types of components can also be referred to as check points. Validation components return a Boolean value after the validation is performed.
- **Generic Components**: Components that perform application tasks regardless of the product family.

The Component Tree on the Components page has the following structure:

Figure 3–1 Component Tree Structure

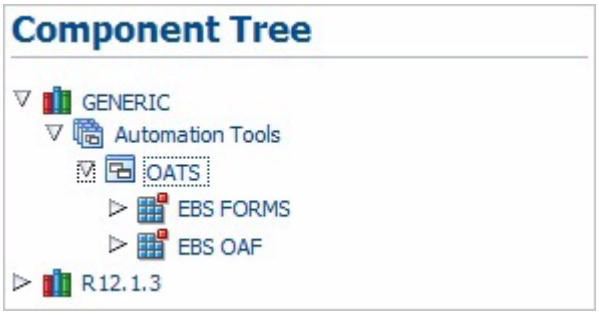
An administrator defines the Release, Product Family, Product, and Feature structure of the Component Tree. Users add Components to the tree and define the keywords and parameters that will be used to specify Component Sets and Flows that generate the OpenScript scripts used to test the application under test.

Clicking the **Components** link at the top of the main window shows the Component Tree and Search Component options. The default Component Tree includes two releases, Generic and R12.1.3, as follows:

Figure 3–2 Default Component Tree

Generic components consist of those components that are functions that can be used across all product families. Expanding the Generic tree shows the list of products and features contained in the Generic tree:

Figure 3–3 Generic Component Tree



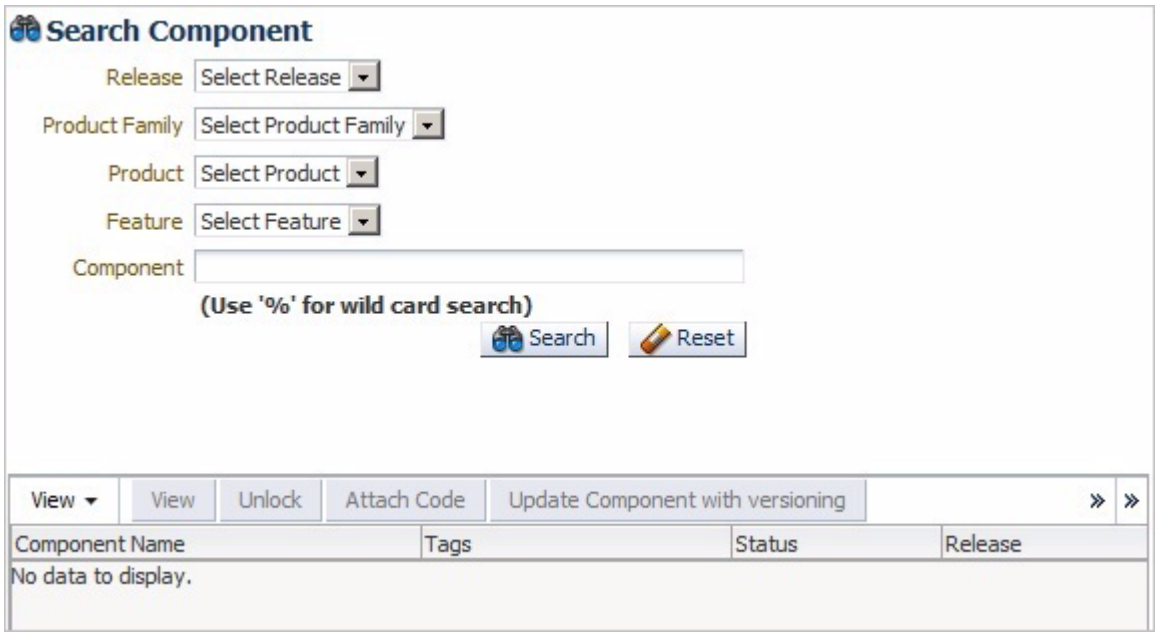
Release and application specific components are listed below the Release(s) listed in the Components Tree. Expanding the tree shows the list of products and features in the release tree:

Figure 3–4 Release Component Tree

Note that the list of available product and features may vary based upon your specific installation.

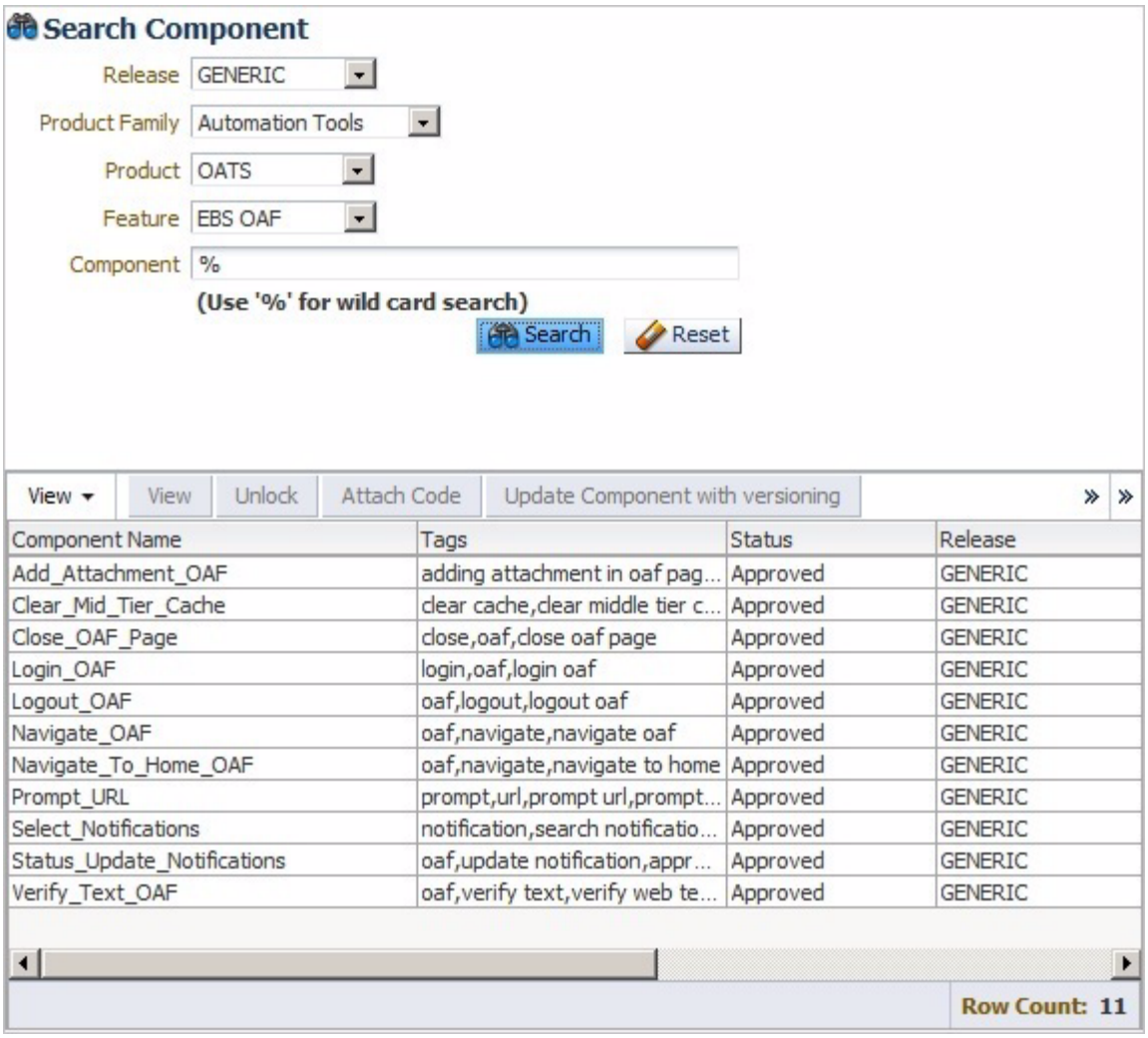
The Search pane of the Components page lets you search for specific components:

Figure 3–5 Search Component Options



You can select the Release, Product Family, Product, Feature, and Component to narrow the search criteria. You can also use the % wildcard character in the Component field to narrow the search:

Figure 3–6 Search Component Options with Search Criteria



3.2 Adding Components

This section explains the procedure for adding components to the Component Tree. Components can be added to the Component Tree in the following two ways:

- Creating/uploading component one at a time under each feature. Components can be created directly in the application UI or uploaded from a pre-defined Excel file.
- Using the bulk upload tool to add multiple components.

3.2.1 Adding Individual Components to the Component Tree

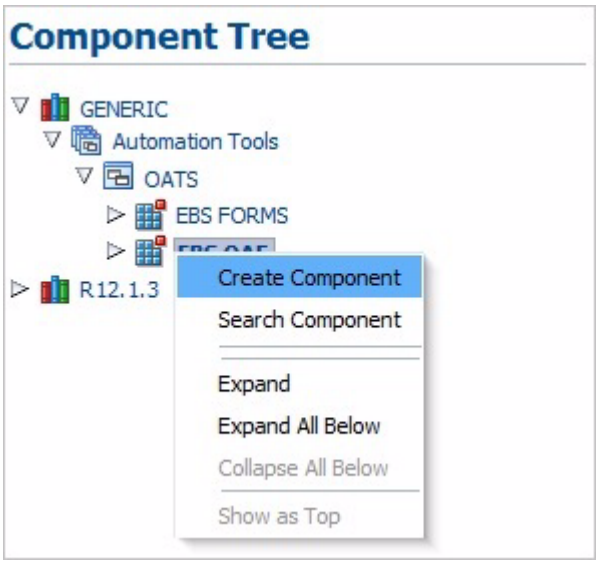
This section explains how to use the Component Tree to add an individual component to a product feature.

To add a component to the Component Tree using the application UI:

1. Expand the Component Tree to the product feature where you want to add the component.

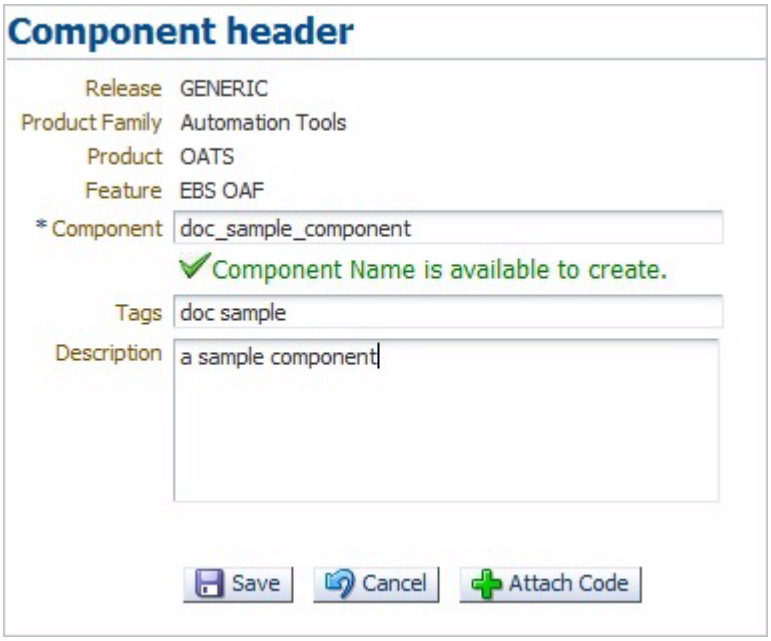
2. Right-click the Feature name and select **Create Component** from the shortcut menu.

Figure 3–7 Create Component Option of the Feature Shortcut Menu



3. In the Component header pane, enter a Component name, Tags and Description.

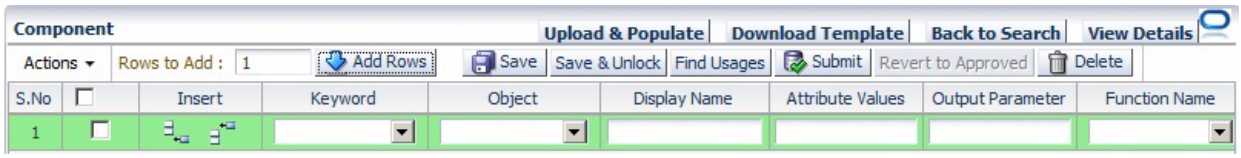
Figure 3–8 Component Header Pane



The component name should be between 5 and 30 characters. The *Component Name is available to create* check mark appears if the name does not currently exist in the database.

4. Click **Attach Code**. The Component Code pane opens for specifying the keywords, objects, and attributes to use to test the component.

Figure 3–9 Component Code Pane



5. Set the number of rows to add and click **Add Rows**.
6. Define the component code by selecting keywords from the **Keywords** list and specifying any objects, attributes and parameters required for the each keyword added to the component code.

   You can also download an Excel spreadsheet template file and specify the component code in the Excel spreadsheet. You can then upload the Excel spreadsheet to the component code pane. See Section 3.2.2, "Uploading Component Code from a Spreadsheet" for additional information about uploading component code from an Excel spreadsheet.
7. Click **Save** when finished. Any syntax errors will be highlighted.
8. Fix any errors and submit the component for approval by clicking on **Submit** or **Submit for Approval**. If your User Role privileges include Approve privileges, the **Submit** button is displayed instead of **Submit for Approval**.

   The component submitted for approval should be approved by respective Product Family owner before it can be used in a flow.
9. Click **Save and Unlock** to save and exit the Component Code pane.

3.2.2 Uploading Component Code from a Spreadsheet

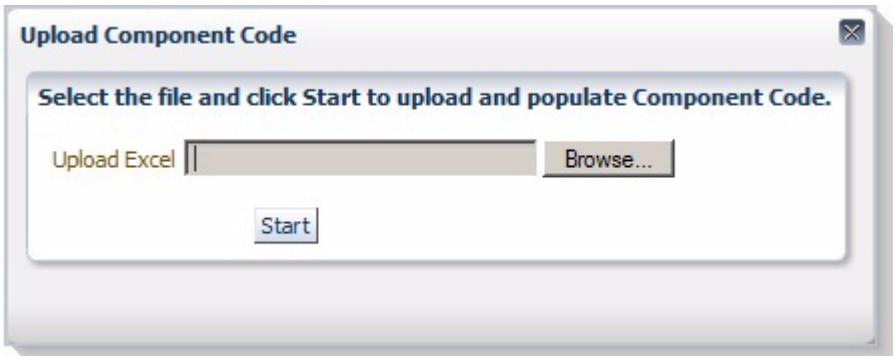
This section explains how to upload component code specified in an Excel spreadsheet. The Oracle Flow Builder includes a template Excel spreadsheet file that can be used to specify keywords, objects, and attributes to use to define a component. The Excel spreadsheet can be uploaded to the component code for a component defined in the Component Tree.

1. Expand the Component Tree to the product feature where you want to add the component.
2. Right-click the Feature name and select **Create Component** from the shortcut menu.
3. In the Component header pane, enter a Component name, Tags and Description.

   The component name should be between 5 and 30 characters. The *Component Name is available to create* check mark appears if the name does not currently exist in the database.
4. Click **Attach Code**. The Component Code pane opens for specifying the keywords, objects, and attributes to use to test the component.
5. Click **Download Template**.
6. Specify a location and save the Excel file.
7. Open and edit the Excel file to define the product information, keywords, objects, attributes and parameters required for the each keyword added to the component code.
8. Save the Excel file as a new name.

9. In the Component Code pane, click **Upload and Populate**.

Figure 3–10 Upload Component Code Dialog Box



10. Click **Browse** and select the Excel spreadsheet file.
11. Click **Start**.
12. Click **Save**. The system displays syntactical errors, if any.
13. Fix any errors and submit the component for approval by clicking on **Submit** or **Submit for Approval**. If your User Role privileges include Approve privileges, the **Submit** button is displayed instead of **Submit for Approval**.
14. Click **Save and Unlock** to save and exit the Component Code pane.

3.2.3 Copying Existing Components

You can copy an existing component to a new component with a new name.

To copy an existing component:

1. Expand the Component Tree to the component that you want to copy.
2. Right-click the component name and select **Copy Component** from the shortcut menu. The component is copied to the clipboard.
3. Expand the Component Tree to the product feature where you want to add the copied component.
4. Right-click the Feature name and select **Paste Component** from the shortcut menu.
5. If you are pasting to the same Product Feature tree or a component with the same name as the copied component exists, enter a new name for the new component and click **Create**.

   The component is added to the tree with an In Progress status.

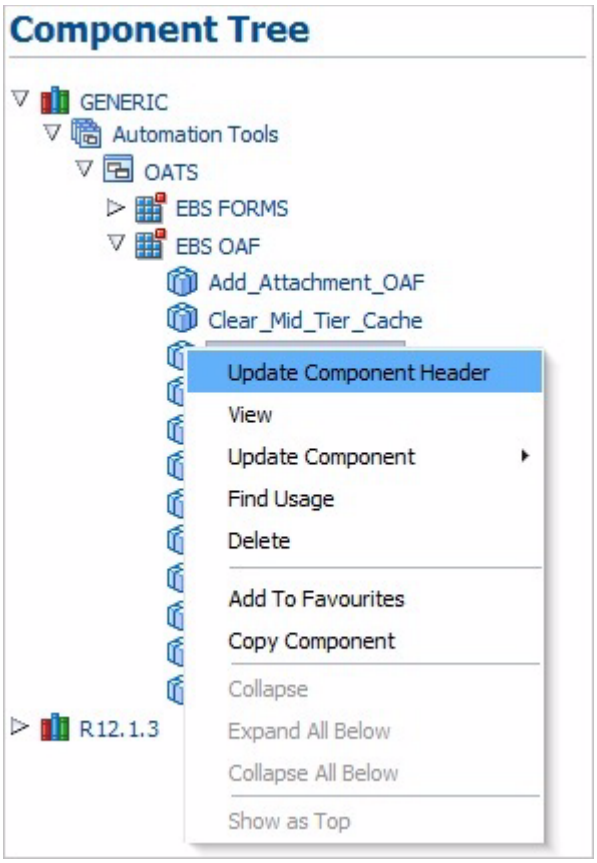
3.3 Updating Components

Existing components can be updated to add or remove keywords or change attributes.

To update an existing component:

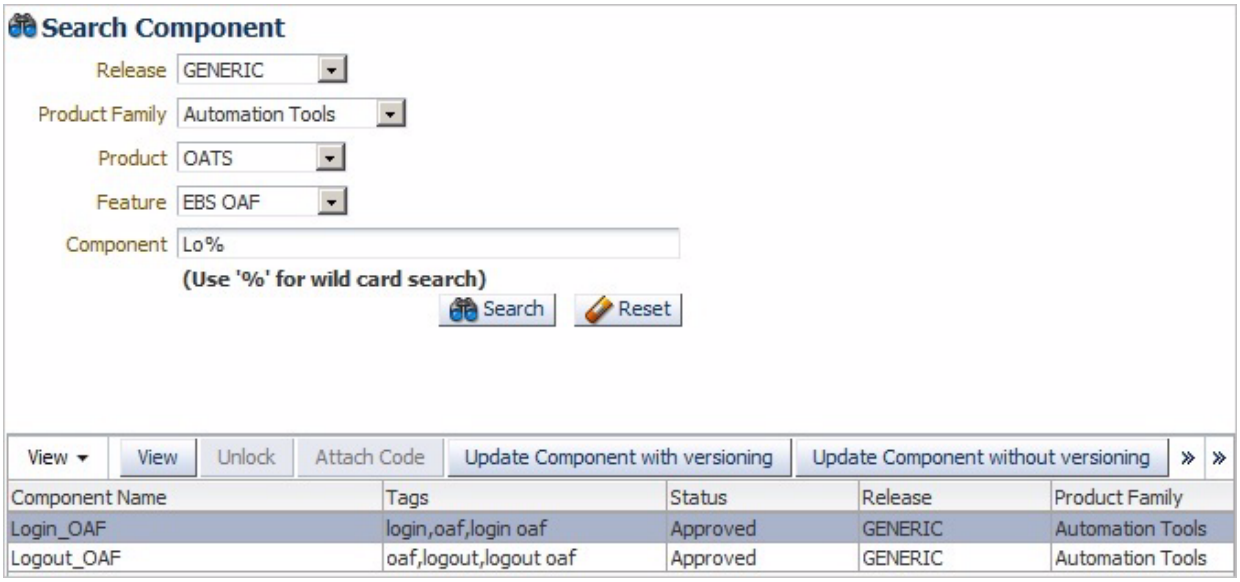
1. Select the component to update using the Component Tree or the Search Component options.
   - If using the Component Tree, right-click the component to open the shortcut menu:

Figure 3–11 Component Update Shortcut Menu



- If using the Search Component options, the update options appear at the top of the search results pane:

Figure 3–12 Component Search with Update Options



The update options are as follows:

Update Component Header: opens the Update Component pane for updating the component header details such as Component name, Tags, and Description.

View: opens the Component Code view in read-only format.

Update Code: opens the Component Code view for editing keywords and attributes. Component code can be updated in two ways:

- **Update code without versioning**: only Attribute Value, Mandatory, Re-runnable and Display Name fields can be updated. The component modified with **Update code without versioning** option does not require approval.
- **Update code with versioning**: allows full component updating for add, delete, modify, surround, and insert rows with various options. The component modified with **Update code versioning**, it must be submitted for approval.

Find Usage: lists all the flows in which the component has been used.

Delete: deletes the component from the Component Tree.

Add to Favourites: adds components that are frequently used in Flows to the Favourites Tree.

Copy Component: copies the component to a new component.

Detach: opens the table in a larger view.

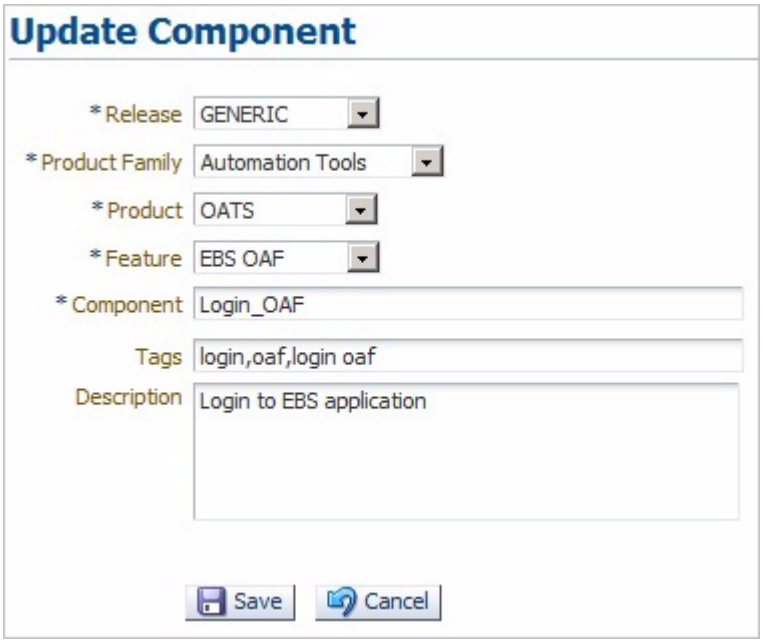
2. Select the update option.

3.3.1 Updating Component Headers

To update the header information for an existing component:

1. Expand the Component Tree to the component to update.
2. Right-click the component and select **Update Component Header** from the shortcut menu.

Figure 3–13 Update Component Pane



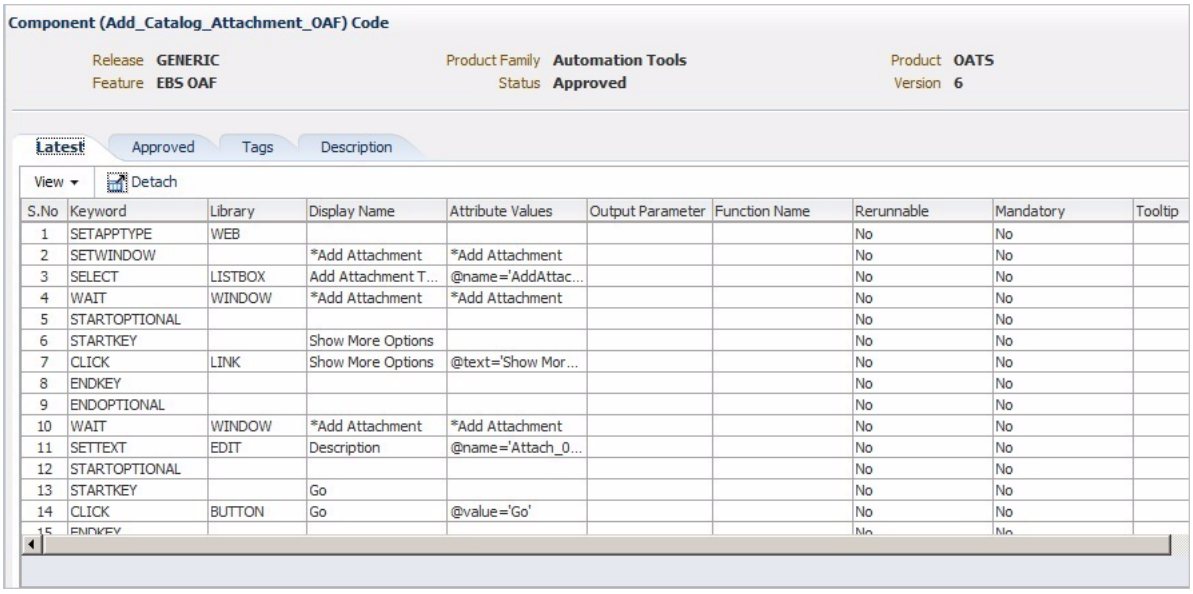
3. Update the component header information as needed and click **Save**.

3.3.2 Viewing Component Code

To view component code for an existing component:

1. Expand the Component Tree or use the search options to select the component to view.
2. Select **View** from the Component Tree shortcut menu or the search pane.

Figure 3–14 View Component Code Pane



S.No	Keyword	Library	Display Name	Attribute Values	Output Parameter	Function Name	Rerunnable	Mandatory	Tooltip
1	SETAPPTYPE	WEB					No	No	
2	SETWINDOW		*Add Attachment	*Add Attachment			No	No	
3	SELECT	LISTBOX	Add Attachment T...	@name='AddAttac...			No	No	
4	WAIT	WINDOW	*Add Attachment	*Add Attachment			No	No	
5	STARTOPTIONAL						No	No	
6	STARTKEY		Show More Options				No	No	
7	CLICK	LINK	Show More Options	@text='Show Mor...			No	No	
8	ENDKEY						No	No	
9	ENDOPTIONAL						No	No	
10	WAIT	WINDOW	*Add Attachment	*Add Attachment			No	No	
11	SETTEXT	EDIT	Description	@name='Attach_0...			No	No	
12	STARTOPTIONAL						No	No	
13	STARTKEY		Go				No	No	
14	CLICK	BUTTON	Go	@value='Go'			No	No	
15	ENDKEY						No	No	

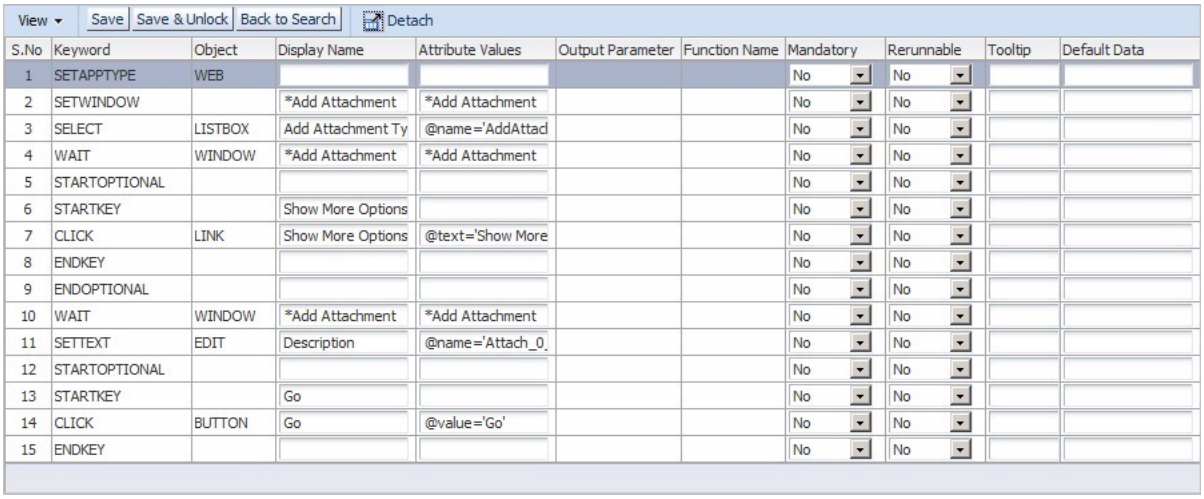
3. Use the **View** menu options to re-order columns and select specific columns to view.
4. Click **Detach** to open the component code in a larger view.
5. Click the **[X]** button to close the component code view when finished.

3.3.3 Updating Component Code without Versioning

To update component code without changing the component code version:

1. Expand the Component Tree or use the search options to select the component to update.
2. Select **Update Code without Versioning** from the Component Tree shortcut menu or the search pane.

Figure 3–15 Update Component Code without Versioning Pane



S.No	Keyword	Object	Display Name	Attribute Values	Output Parameter	Function Name	Mandatory	Rerunnable	Tooltip	Default Data
1	SETAPPTYPE	WEB					No	No		
2	SETWINDOW		*Add Attachment	*Add Attachment			No	No		
3	SELECT	LISTBOX	Add Attachment Ty	@name='AddAttacl			No	No		
4	WAIT	WINDOW	*Add Attachment	*Add Attachment			No	No		
5	STARTOPTIONAL						No	No		
6	STARTKEY		Show More Options				No	No		
7	CLICK	LINK	Show More Options	@text='Show More			No	No		
8	ENDKEY						No	No		
9	ENDOPTIONAL						No	No		
10	WAIT	WINDOW	*Add Attachment	*Add Attachment			No	No		
11	SETTEXT	EDIT	Description	@name='Attach_0			No	No		
12	STARTOPTIONAL						No	No		
13	STARTKEY		Go				No	No		
14	CLICK	BUTTON	Go	@value='Go'			No	No		
15	ENDKEY						No	No		

Only Attribute Value, Mandatory, Re-runnable and Display Name fields can be updated in this pane.

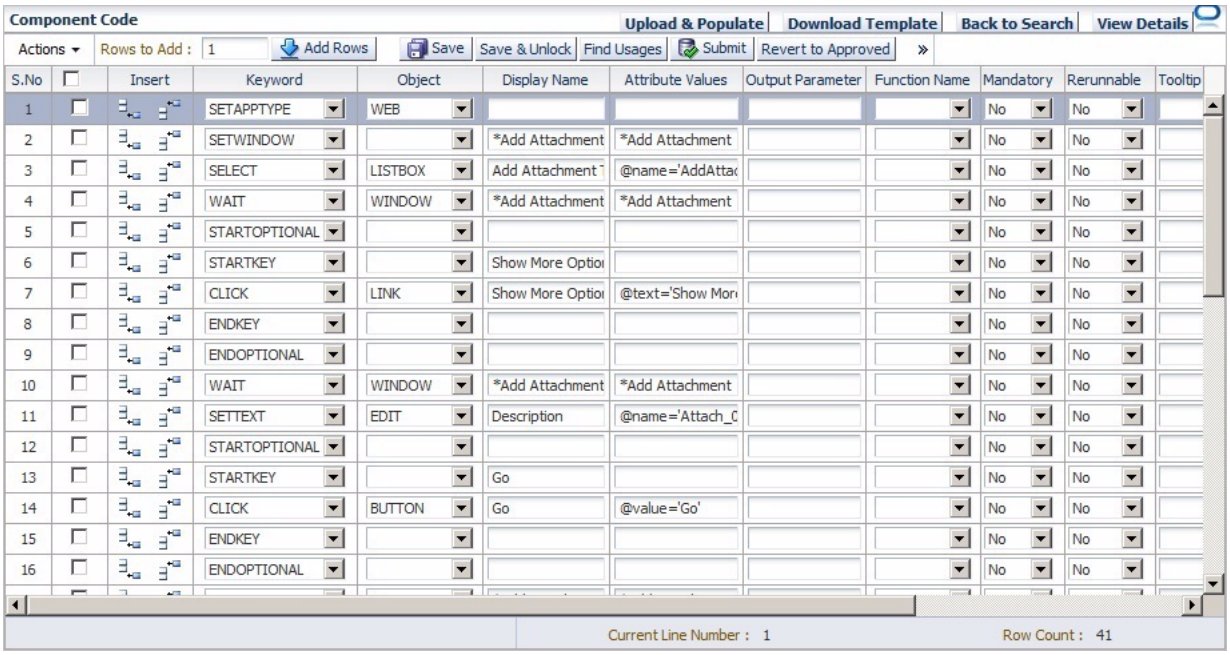
3. Click **Save** or **Save & Unlock** after necessary modifications are made. Click **Back to Search** to exit without saving changes.

3.3.4 Updating Component Code with Versioning

To update component code and change the component code version:

1. Expand the Component Tree or use the search options to select the component to update.
2. Select **Update Code with Versioning** from the Component Tree shortcut menu or the search pane.

Figure 3–16 Update Component Code with Versioning Pane



S.No	Insert	Keyword	Object	Display Name	Attribute Values	Output Parameter	Function Name	Mandatory	Rerunnable	Tooltip
1		SETAPPTYPE	WEB					No	No	
2		SETWINDOW		*Add Attachment	*Add Attachment			No	No	
3		SELECT	LISTBOX	Add Attachment	@name='AddAttac			No	No	
4		WAIT	WINDOW	*Add Attachment	*Add Attachment			No	No	
5		STARTOPTIONAL						No	No	
6		STARTKEY		Show More Optio				No	No	
7		CLICK	LINK	Show More Optio	@text='Show Mor			No	No	
8		ENDKEY						No	No	
9		ENDOPTIONAL						No	No	
10		WAIT	WINDOW	*Add Attachment	*Add Attachment			No	No	
11		SETTEXT	EDIT	Description	@name='Attach_0			No	No	
12		STARTOPTIONAL						No	No	
13		STARTKEY		Go				No	No	
14		CLICK	BUTTON	Go	@value='Go'			No	No	
15		ENDKEY						No	No	
16		ENDOPTIONAL						No	No	

This pane allows full component updating for add, delete, modify, surround, and insert rows with various options.

Adding rows: specify the number of rows to add in **Rows to Add** and click **Add Rows**. The specified number of rows are added to the end of the table.

Deleting rows: select the checkbox for the row to delete and click **Delete** or click the Trash can icon at the end of the row.

Inserting rows: select the row where you want to insert a new row before or after. Click the Insert Before or Insert After icon in the **Insert** column of the row.

Inserting a structure before or after a row: select the row where you want to insert a new structure. From the **View** menu, select **Insert Structure**, select **Above** or **Below**, then select the type of structure: **Start Optional**, **Start Group**, **Start Iterate**.

Surrounding a row with a structure: select the row that you want to surround with a new structure. From the **View** menu, select **Surround With**, then select the type of structure: **Start Optional**, **Start Group**, **Start Iterate**.

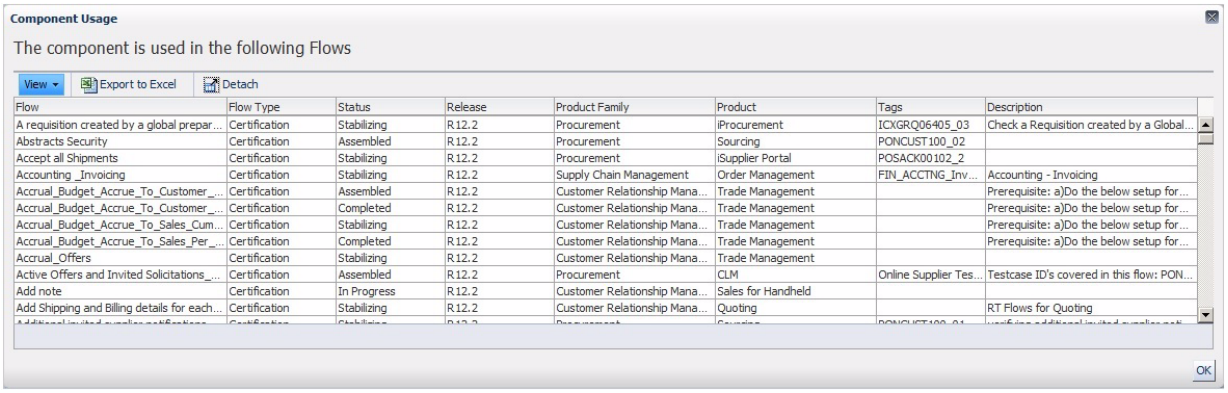
3. Click **Save** or **Save & Unlock** after necessary modifications are made. Click **Back to Search** to exit without saving changes.

3.3.5 Finding Component Usage

To find the flows in which a component is used:

1. Expand the Component Tree or use the search options to select the component to find usage.
2. Select **Find Usage** from the Component Tree shortcut menu or the search pane.

Figure 3–17 Component Usage Window



Use the **View** menu options to show or hide columns, detach the report to a separate view, or reorder the columns.

Use the **Export to Excel** option to save the data to an Excel spreadsheet file.

3. Click the window close button when finished.

3.4 Component Development Guidelines

This section provides guidelines for developing component code.

- The data sheet that is created in the flow will contain the columns which when populated will generate the code for execution for each component.
- One component cannot be called from any other component.
- Input and output parameters must be specified using GET and SET keywords.
- Any looping construct must be created in generic functions and not in the components.

3.4.1 Component Code

The Component Code defines the keywords, objects, and attributes that define the test actions for the component.

3.4.1.1 Keywords

This section provides guidelines for using Keywords to specify component code.

Setting Application Type and Window

Component creation in the UI should begin with setting the Application Type and Window using the SETAPPTYPE and SETWINDOW keywords to specify the type of application and the window name. The following table shows an example of the component code steps:

Table 3–1 Keywords for Setting Application Type and Window

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	Web		
2	SETWINDOW		Login	Login

Step 1 sets the application type as a web application.

Step 2 sets the window to the window with the display name Login and attribute value Login.

Grouping Statements

Statements in component code must be defined within STARTGROUP and ENDGROUP keywords in the following scenario:

Statement 1

Statement a

Statement b

Statement c

If all statements a, b, c need to be executed only if input is provided to Statement 1. The following table shows an example of the component code steps:

Table 3–2 Keywords for Grouping Statements

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	Web		
2	SETWINDOW		Login	Login
3	STARTGROUP			

Table 3–2 (Cont.) Keywords for Grouping Statements

Step #	Keyword	Object Type	Display Name	Attribute Values
4	SELECT	List	Actions	Actions
5	FUNCTIONCALL	GENLIB		Refresh List Box
6	ENDGROUP			

Step 1 sets the application type as a web application.

Step 2 sets the window to the window with the display name Login and attribute value Login.

Step 3 defines the start of the statement group.

Step 4 specifies a SELECT action which requires test data to select a value in list box.

Step 5 specifies a FUNCTIONCALL. This step does not require any test data to be provided, but is a mandatory step or code to be generated if test data is provided for Step 4.

Step 6 defines the end of the statement group.

Setting Tab Pages

Components that have multiple tabs/pages in the application (a simple set of pages, for example, train) should be placed within STARTTAB and ENDTAB keywords in the following scenario:

Statement 1

Statement 2

Statement 3

Statement 4

If input is provided for any of Statements 2, 3, or 4, then only Statement 1 will be executed.

The STARTTAB and ENDTAB Keyword set can contain any other Keyword combinations. For example, in between STARTTAB and ENDTAB can have a STARTGROUP and ENDGROUP Keyword set or a STARTITERATE and ENDITERATE Keyword set.

The following table shows an example of the component code steps:

Table 3–3 Keywords for Setting Tab Pages

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	FORMS		
2	SETWINDOW		Purchase Order	PO_HEADER
3	STARTTAB			Lines
4	CLICK	TAB	Lines	TAB_LINES
5	SETTEXT	EDIT	Item Description	Item_Description_0
6	SETTEXT	EDIT	Promise Date	PROMISEDATE_0
7	ENDTAB			Lines
8	STARTTAB			More

Table 3–3 (Cont.) Keywords for Setting Tab Pages

Step #	Keyword	Object Type	Display Name	Attribute Values
9	CLICK	TAB	More	TAB_MORE
10	SETTEXT	EDIT	Amount	AMOUNT_HEADER
11	ENDTAB			More

Step 1 sets the application type as a FORMS application.

Step 2 sets the window to the window with the display name Purchase Order and attribute value PO_HEADER.

Step 3 defines the start of the Lines tab.

Step 4 clicks the Lines tab if a value is provided for either "Item Description" or "Promised Date". Otherwise it will be skipped.

Steps 5 and 6 set the text for the edit boxes "Item Description" and "Promise Date".

Step 7 defines the end of the Lines tab.

Step 8 defines the start of the More tab.

Step 9 clicks the More tab if a value is provided for "Amount". Otherwise it will be skipped.

Steps 10 sets the text for the edit box "Amount".

Step 11 defines the end of the More tab.

Setting Optional Navigation

Components that have multi-page Next/Submit options for further navigation use the STARTOPTIONAL and ENDOPTIONAL Keyword set. Figure 3–18 shows an example of a component with Next/Submit options:

Figure 3–18 Component with Next/Submit Navigation

The STARTKEY and ENDKEY Keywords are required within the STARTOPTIONAL and ENDOPTIONAL Keyword set. Only one statement is required within the STARTKEY and ENDKEY keyword set.

The following table shows an example of the component code steps:

Table 3–4 Keywords for Optional Navigation

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	Web		
2	SETWINDOW			Create Transaction
3	SETTEXT	EDIT	Field 1	FIELD_1
4	SELECT	LIST	Field 2	FIELD_2
5	STARTOPTIONAL			
6	STARTKEY			

Table 3–4 (Cont.) Keywords for Optional Navigation

Step #	Keyword	Object Type	Display Name	Attribute Values
7	CLICK	BUTTON	ClickNext	SUBMIT_NEXT
8	ENDKEY			
9	WAIT	WINDOW	Page 2	Login
10	ENDOPTIONAL			
11	STARTOPTIONAL			
12	STARTKEY			
13	CLICK	BUTTON	ClickSubmit	SUBMIT
14	ENDKEY			
15	WAIT	WINDOW	Page 3	Page 3
16	ENDOPTIONAL			

Step 1 sets the application type as a web application.

Step 2 sets the window to the window with the attribute value "Create Transaction".

Step 3 sets the text for the edit box "Field 1".

Step 4 selects the list box "Field 2".

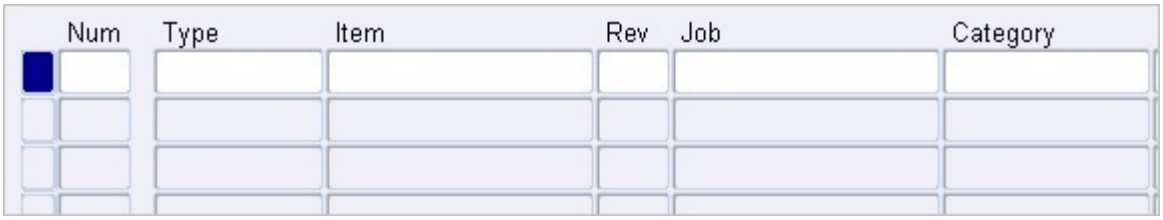
Step 5 through 10 define the STARTOPTIONAL and ENDOPTIONAL Keyword set for the Next button click.

Step 11 through 16 define the STARTOPTIONAL and ENDOPTIONAL Keyword set for the Submit button click.

Setting Line Items

Components that have UI with line items use the STARTITERATE and ENDITERATE Keyword set with the MAXVISIBLELINES and SETLINE Keywords. Figure 3–19 shows an example of a component with line items.

Figure 3–19 Component with Line Items



MAXVISIBLELINES and SETLINE are required within STARTITERATE and ENDITERATE.

- The MAXVISIBLELINES keyword specifies the maximum number of rows visible in the form of the component. The value is set in the "Attribute Value" column. There is no input parameter for this keyword.
- The SETLINE keyword specifies the current line using one input parameter as the "Display Name".
- All lines item functionality UI components that have Attribute value changes such as _0, _1, etc. must be between the STARTITERATE & ENDITERATE Keywords.

The following table shows an example of the component code steps:

Table 3–5 Keywords for Setting Line Items

Step #	Keyword	Object Type	Display Name	Attribute Values
1	STARTITERATE			
2	MAXVISIBLELINES			4
3	SETLINE		PO Line Number	
4	SETFOCUS	EDIT	Item Description	ITEM_DESCRIPTION_0
5	SETTEXT	EDIT	Type	ITEM_TYPE_0
6	SETTEXT	EDIT	Category	ITEM_CATEGORY_0
7	ENDITERATE			

Step 1 starts the STARTITERATE/ENDITERATE Keyword set.

Step 2 sets the maximum visible lines in the form.

Step 3 sets the current line to the PO Line Number.

Step 4 set the focus to the Item Description.

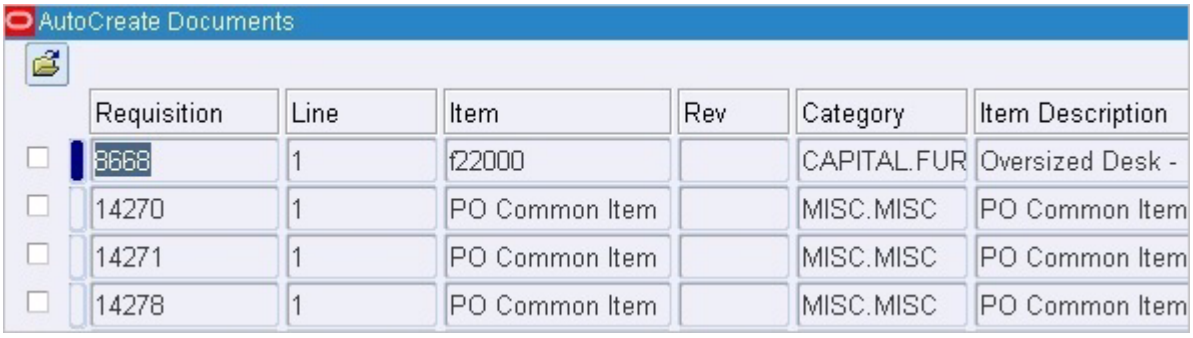
Step 5 and 6 set the text in the fields.

Step 7 ends the STARTITERATE/ENDITERATE Keyword set.

Setting Line Items with Column Search

Components that have UI with line items where column are searched using the STARTITERATE and ENDITERATE Keyword set with the MAXVISIBLELINES and SEARCHCOLUMN Keywords. Figure 3–20 shows an example of a component with line items with searchable columns.

Figure 3–20 Component with Line Items and Columns



The following table shows an example of the component code steps:

Table 3–6 Keywords for Setting Line Items with Column Search

Step #	Keyword	Object Type	Display Name	Attribute Values
1	STARTITERATE			
2	MAXVISIBLELINES			4
3	SEARCHCOLUMN		Requisition	REQUISTION_NUM_0

Table 3–6 (Cont.) Keywords for Setting Line Items with Column Search

Step #	Keyword	Object Type	Display Name	Attribute Values
4	SEARCHCOLUMN		Line	REQ_LINE_0
5	CHECK	CHECKBOX	Select REQ Line	SELECT_REQ_LINE_0
6	SETFOCUS	EDIT	Item	REQ_ITEM_0
7	ENDITERATE			

Step 1 starts the STARTITERATE/ENDITERATE Keyword set.

Step 2 sets the maximum visible lines in the form.

Step 3 searches the Requisition column.

Step 4 searches the Line column.

Step 5 checks the check box.

Step 6 set the focus to the Item field.

Step 7 ends the STARTITERATE/ENDITERATE Keyword set.

Setting Wait for Window

Component code that causes a page navigation action (for example, clicking on a link, image, or button) should include a WAIT Keyword on the new page to provide better stability of the script.

The following table shows an example of the component code steps:

Table 3–7 Keywords for Setting Wait for Window

Step #	Keyword	Object Type	Display Name	Attribute Values
1	CLICK	BUTTON	Login	Login
2	SETWINDOW		Oracle Applications Home page	Oracle Applications Home page
3	WAIT	WINDOW	Oracle Applications Home page	Oracle Applications Home page

Step 1 clicks the login button.

Step 2 sets the new window.

Step 3 waits for the new window.

Setting Actions on Form Tabs

Component code for form tabs uses the "Display Name" column to specify the visible text on the tab and the "Attribute Value" to specify the @name attribute of the tab.

The following table shows an example of the component code steps:

Table 3–8 Keywords for Setting Actions on Form Tabs

Step #	Keyword	Object Type	Display Name	Attribute Values
1	CLICK	TAB	Include Function	REGIONS

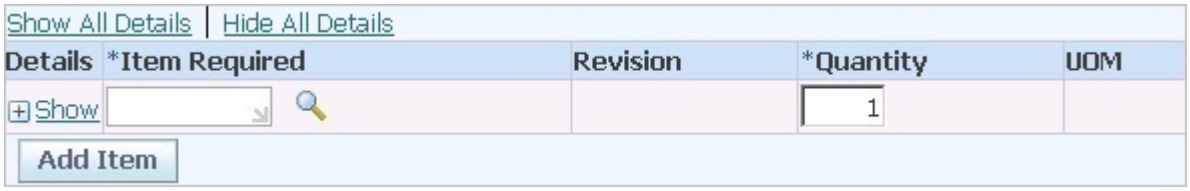
Step 1 clicks the Include Function tab.

Setting Table Name

Components that have UI with an HTML/OAF table are specified using the SETTABLENAME Keyword with the STARTITERATE/ENDITERATE Keywords set and the SETLINE Keyword. SETTABLENAME should specify the first row, first column value. If not available, any column in the table.

The HTML/OAF table should be a separate component. Figure 3–21 shows an example of a component with an HTML/OAF table.

Figure 3–21 Component with HTML/OAF Table



Display Name: should be the column name of the table.

Attribute Value: should be the column name of the table.

Note: if there is more than one table named "Details", specify the table attribute value as "Details", "Details;1" or "Details;2" depending on the sequence. By default "Details" is Details;0 (index starts from zero).

The following table shows an example of the component code steps:

Table 3–9 Keywords for Setting Table Name

Step #	Keyword	Object Type	Display Name	Function Name	Attribute Values
1	SETTABLENAME		Details		Details
2	STARTITERATE				
3	SETLINE		Item Line Number		
4	FUNCTIONCALL	GENLIB	*Item Description	WEBSELECTLOV	Search for Item Description
5	SETTEXT	EDIT	*Quantity		quantity
7	ENDITERATE				

Step 1 sets the table name.

Step 2 starts the STARTITERATE/ENDITERATE Keyword set.

Step 3 sets line number.

Step 4 calls the WEBSELECTLOV function from the GENLIB function library.

Step 5 sets the quantity text.

Step 6 ends the STARTITERATE/ENDITERATE Keyword set.

Setting Table Name and Column Search

Components that have UI with an HTML/OAF table are specified using the SETTABLENAME Keyword with the STARTITERATE/ENDITERATE Keywords set and the SEARCHCOLUMN Keyword. SETTABLENAME should specify the first row, first column value. If not available, any column in the table.

If a combination of search columns is unique, 2 search columns can be specified, For example:

SETTABLENAME

STARTITERATE

SEARCHCOLUMN 1st column name as display name

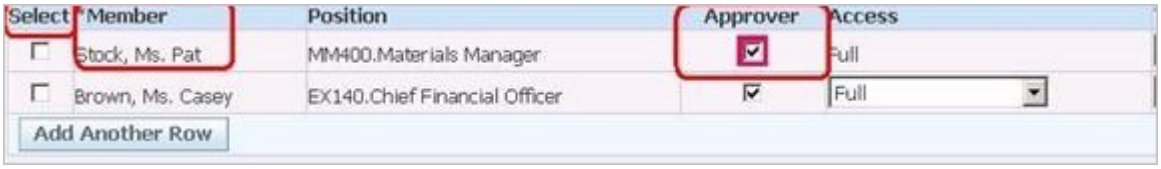
SEARCHCOLUMN 2nd column name as display name

CLICK LINK

ENDITERATE

Figure 3–22 shows an example of a component with an HTML/OAF table with columns.

Figure 3–22 Component with HTML/OAF Table with Columns



Display Name: should be the column name of the table.

Attribute Value: should be the column name of the table.

Note: if there is more than one table named "Select", specify the table attribute value as "Select", "Select;1" or "Select;2" depending on the sequence. By default "Select" is Select;0 (index starts from zero).

The following table shows an example of the component code steps:

Table 3–10 Keywords for Setting Table Name and Column Search

Step #	Keyword	Object Type	Display Name	Function Name	Attribute Values
1	SETTABLENAME		Select		Select
2	STARTITERATE				
3	SEARCHCOLUMN		*Member		
4	SEARCHCOLUMN		Position		
5	FUNCTIONCALL	GENLIB	*Item Description	WEBSELECTLOV	Search for Item Description
6	SETTEXT	EDIT	*Quantity		quantity
7	CHECK	CHECKBOX	Approver		approver
8	ENDITERATE				

Step 1 sets the table name.

Step 2 starts the STARTITERATE/ENDITERATE Keyword set.

Step 3 search for *Member column.

Step 4 search for Position column.

Step 5 calls the WEBSELECTLOV function from the GENLIB function library.

Step 6 sets the Quantity text.

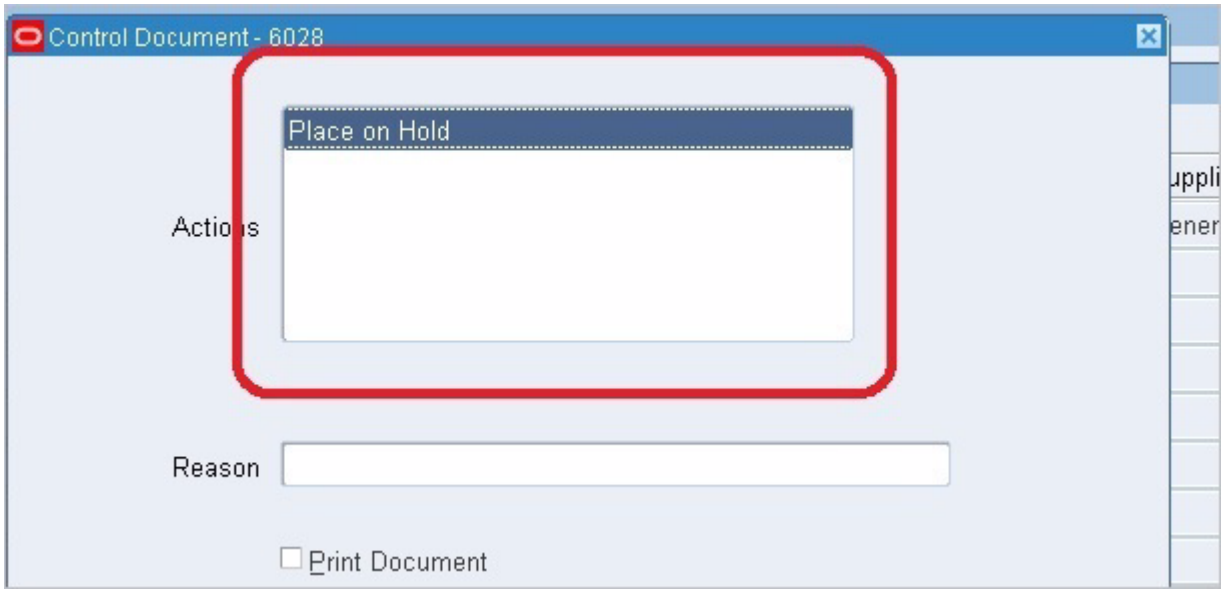
Step 7 sets the Approver checkbox.

Step 8 ends the STARTITERATE/ENDITERATE Keyword set.

Setting Form Treelists

Components that have UI with Form treelist objects are specified using the SELECT Keyword specifying the object as TREELIST. Figure 3–23 shows an example of a component with a Form Treelist.

Figure 3–23 Component with Forms Treelist



The following table shows an example of the component code steps:

Table 3–11 Keywords for Form Treelist

Step #	Keyword	Object Type	Display Name	Attribute Values
1	STARTITERATE			
2	MAXVISIBLELINES			4
3	SEARCHCOLUMN		Requisition	REQUISTION_NUM_0
4	SEARCHCOLUMN		Line	REQ_LINE_0
5	CHECK	CHECKBOX	Select REQ Line	SELECT_REQ_LINE_0
6	SETFOCUS	EDIT	Item	REQ_ITEM_0
7	ENDITERATE			

Step 1 starts the STARTITERATE/ENDITERATE Keyword set.

Step 2 sets the maximum visible lines in the form.

3.4.1.2 Objects

Objects specify the type of object on which to perform the keyword action. Specific keywords have specific object types. The list of object types for a keyword will appear in the Object list when the keyword is specified in the component code window. See Appendix A for additional information.

3.4.1.3 Display Name

The Display Name column specifies the text that displays in the UI of the component. The Display Name is mandatory for all action related statements. For example, SETTEXT, CLICK, etc.

3.4.1.4 Attribute Values

The Attribute Values column specifies attribute name-value pairs. For Form objects the default is its name attribute. For example, id=usernamefield, name=USERNAME_0 for web.

If name-value pairs are not provided, the following default attributes are used for each UI component.

Table 3–12 Default Attributes for Object Types

Object	Attribute Defaults
ALERT	Web: id, Form: not required
BUTTON	id
CHECKBOX	id
CHOICEBOX	id
EDIT	name
FLEXWINDOW	name
IMAGE	alt
LINK	text
LIST	id
LISTBOX	id
LOV	Web: alt attribute of the torch icon in the web page, Form: name attribute of the text field on which the List of Values needs to be invoked.
RADIOBUTTON	id
TAB	name (display name is mandatory with the visible tab value or you can record and get the text)
TABLE	all tables are handled through first cell value or first non blank header.
TEXTAREA	id
TOOLBAR	none
TREE	Web: need to check, Form: name
STATUS	none
STATUSBAR	none
WINDOW	Web: title, Form: name

3.4.1.5 Output Parameter

Output parameters must be specified using SET keywords.

3.4.1.6 Function Name

Specifies the name of the function to call when the Keyword FUNCTIONCALL is specified.

3.4.1.7 Mandatory

Specify Yes in this column for all mandatory fields. A No value is the default if this field is left empty.

3.4.1.8 Rerunable

Specify Yes for fields where the value entered should be unique. Specifying Yes for the Rerunnable column for any field appends a random variable to the data passed. The random variable will be preceeded with question mark. For example, fields such as Username or Item name, etc. should be unique. There cannot be two users or items with same name.

3.4.1.9 Tooltip

The data provided in this column will be shown as a tooltip to the user during flow creation. The tooltip column is used to provide details such as:

- Field is defaulted
- Field is mandatory
- Field is dependent on other previous fields

4

Defining Components Sets

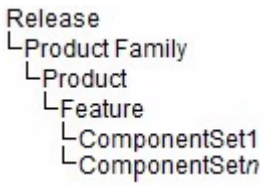
This chapter explains how to add and update components sets in the Component Set Tree. This chapter contains the following sections:

4.1 Overview

Component Sets group components that are used together for specific functionality. The Components Set Tree represents a library of defined components sets which can then be added to Flows that specify a test instance. A Component Set in the Component Set Tree contains a set of related components that can b e used to quickly build Flows.

The Component Set Tree on the Component Sets page has the following structure:

Figure 4–1 Component Set Tree Structure



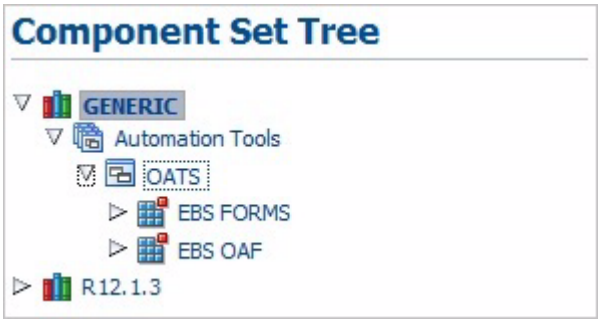
A Oracle Flow Builder administrator defines the Release, Product Family, Product, and Feature structure of the Component Set Tree. Oracle Flow Builder users add Components to the tree and define the keywords and parameters that will be used to specify Component Sets and Flows that generate the OpenScript scripts used to test the application under test.

Clicking the **Component Sets** link at the top of the main window shows the Component Set Tree and Search Component Set options. The default Component Set Tree includes two releases, Generic and R12.1.3, as follows:

Figure 4–2 Default Component Set Tree

Generic components sets consist of those component sets that group functions that can be used across all product families. Expanding the Generic tree shows the list of products and features contained in the Generic tree:

Figure 4–3 Generic Component Set Tree



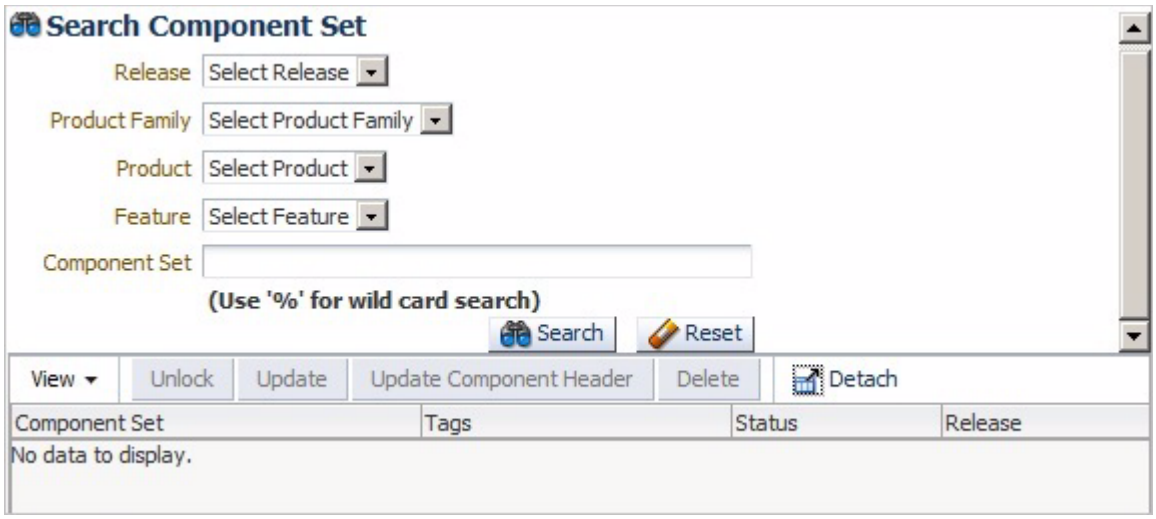
Release and application specific components are listed below the Release(s) listed in the Components Set Tree. Expanding the tree shows the list of products and features in the release tree:

Figure 4–4 Release Component Set Tree

Note that the list of available product and features may vary based upon your specific installation.

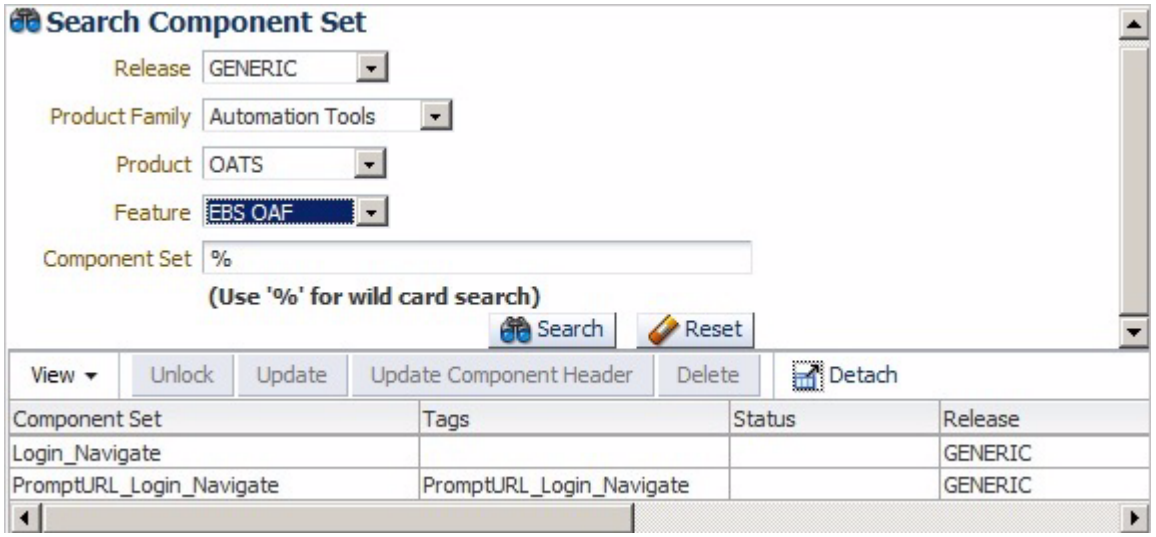
The Search pane of the Component Set page lets you search for specific component sets:

Figure 4–5 Search Component Set Options



You can select the Release, Product Family, Product, Feature, and Component Set to narrow the search criteria. You can also use the % wildcard character in the Component Set field to narrow the search:

Figure 4–6 Search Component Set Options with Search Criteria



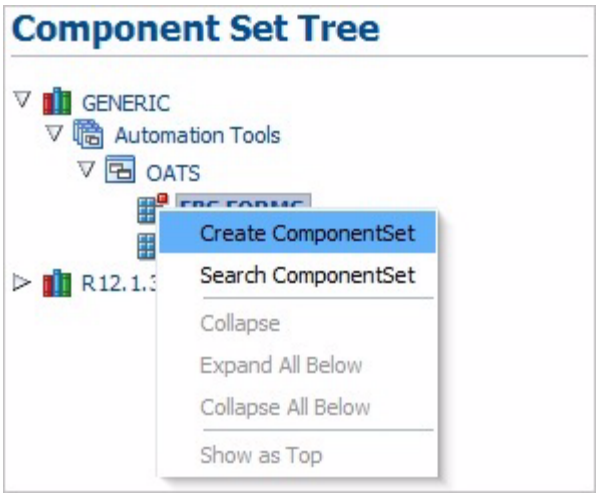
4.2 Adding Component Sets

This section explains the procedure for adding component sets to the Component Sets Tree. Component Sets can be added to the Component Set Tree directly in the application UI.

To add a component set to the Component Set tree:

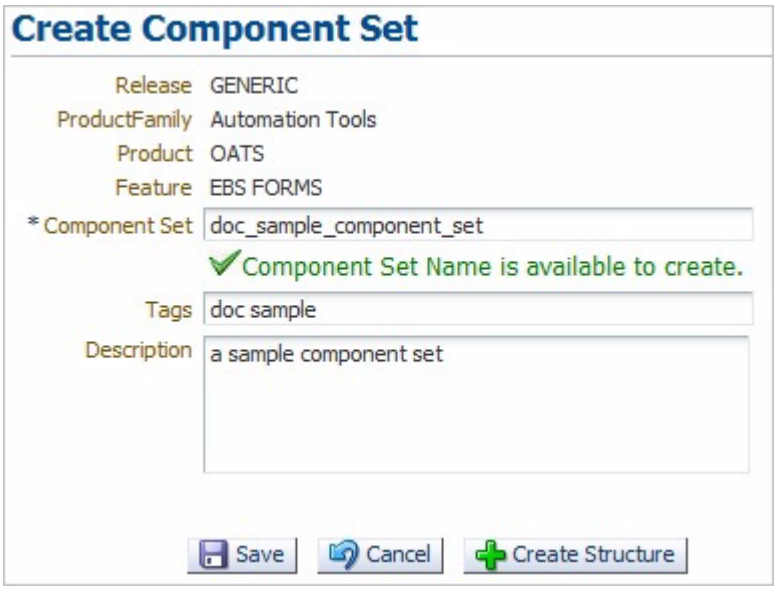
1. Expand the Component Set Tree to the product feature where you want to add the component set.
2. Right-click the Feature name and select **Create Component Set** from the shortcut menu.

Figure 4–7 Create Component Set Option of the Shortcut Menu



3. In the Create Component Set pane, enter a Component Set name, Tags and Description.

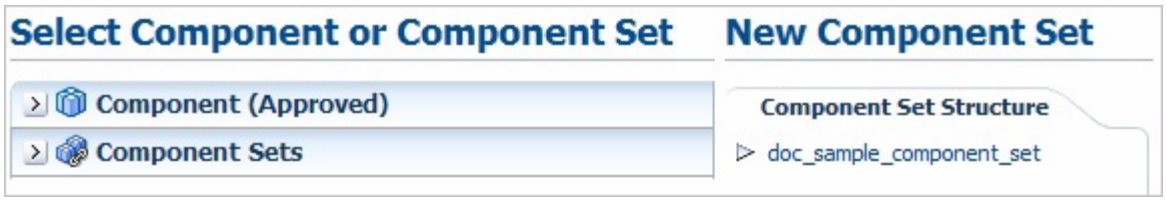
Figure 4–8 Create Component Set Pane



The component set name should be between 5 and 30 characters. The *Component Set Name is available to create* check mark appears if the name does not currently exist in the database.

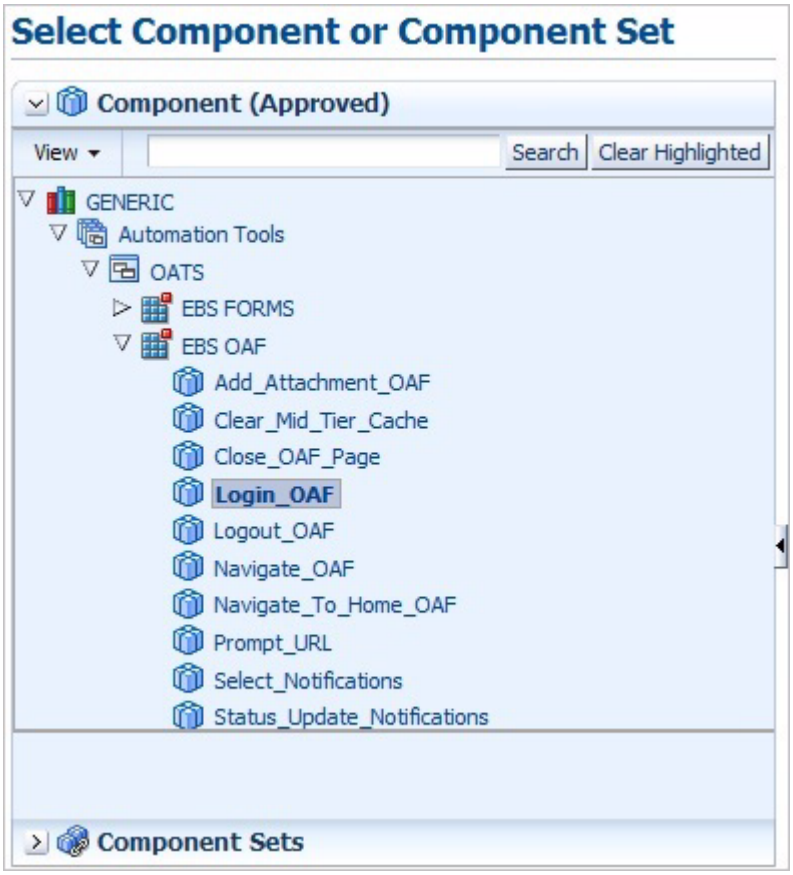
4. Click **Create Structure**. The Select Component or Components Set and the New Component Set pane opens for specifying the components or component sets to add to the new component set.

Figure 4–9 New Component Set Pane



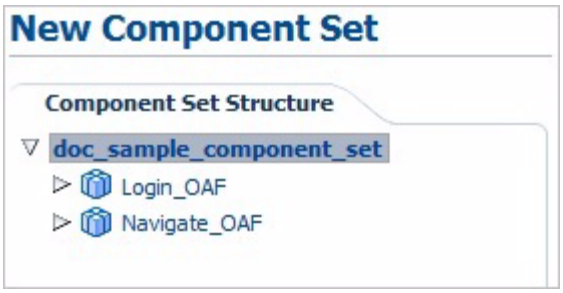
5. Expand the Component or Components Sets tree to view the available components or component sets.

Figure 4–10 Select Component Set Pane with Expanded Components Tree



6. Right-click the component or component set to add to the New Component Set and click **Move**. You can also click-and-drag components to the New Component Set tree.
7. Repeat step 6 to add additional components or component sets to the New Component Set as needed.

Figure 4–11 New Component Set Pane with Components Added



8. Click **Unlock** when finished to save and exit the New Component Set pane. The Search pane indicates *Publish Component Set Successfully.*

4.3 Updating Component Sets

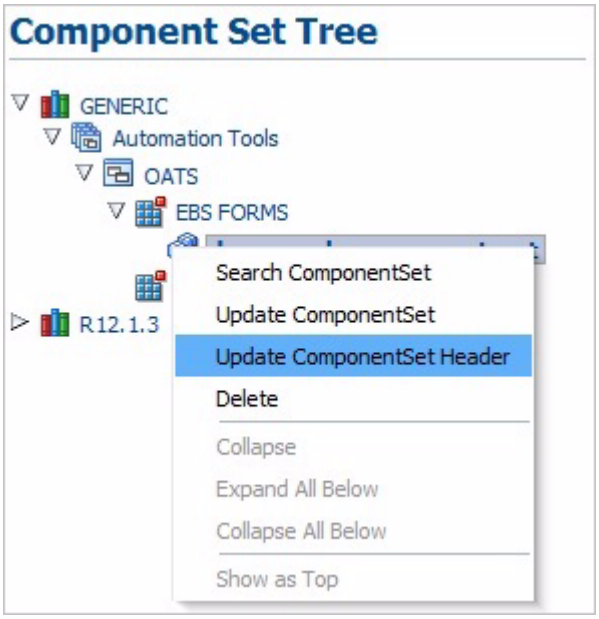
Existing component sets can be updated to add or remove components or components sets from a component set.

4.3.1 Updating Component Set Headers

To update component set header:

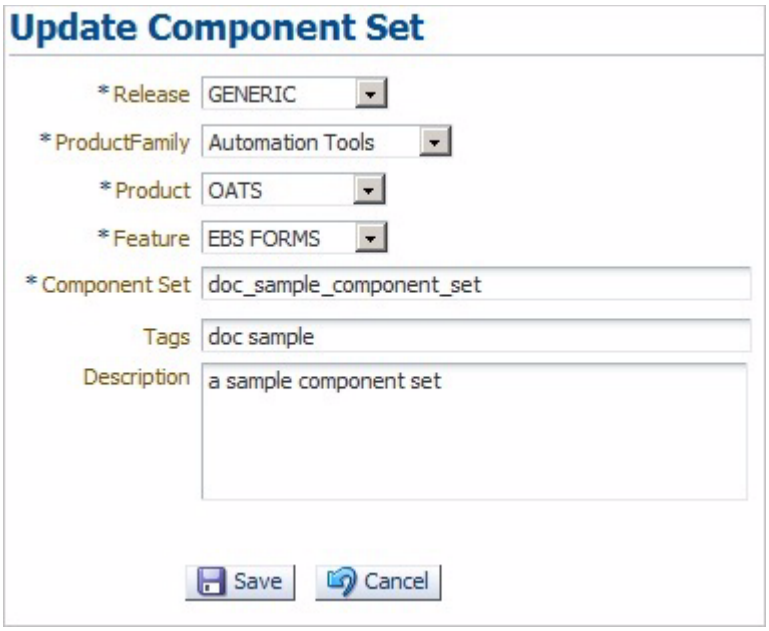
1. Expand the Component Set Tree to the component set you want to update.
2. Right-click the Component Set name and select **Update Component Set Header** from the shortcut menu.

Figure 4–12 Update Component Set Header Option of the Shortcut Menu



3. In the Update Component Set pane, edit the Component Set name, Tags and Description.

Figure 4–13 Update Component Set Pane



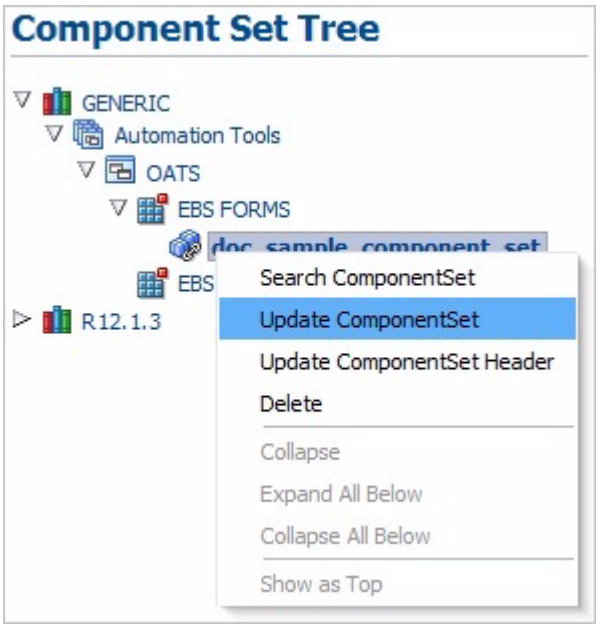
4. Click **Save** when finished to save and exit the Update Component Set pane. The Search pane indicates *Updated Component Set <set name> Successfully.*

4.3.2 Adding Components or Component Sets to an Existing Component Set

To add a component or component set to an existing component set:

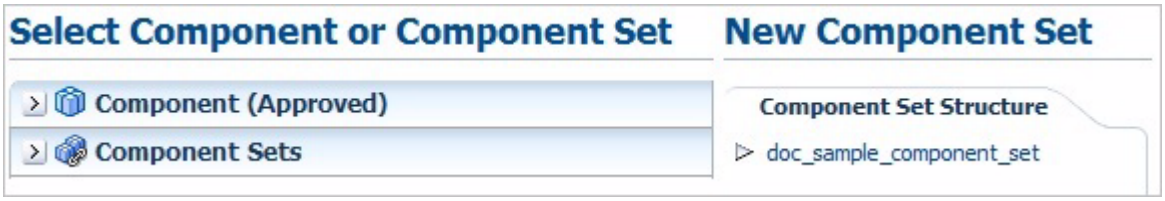
1. Expand the Component Set Tree to the component set you want to update.
2. Right-click the Component Set name and select **Update Component Set** from the shortcut menu.

Figure 4–14 Update Component Set Option of the Shortcut Menu



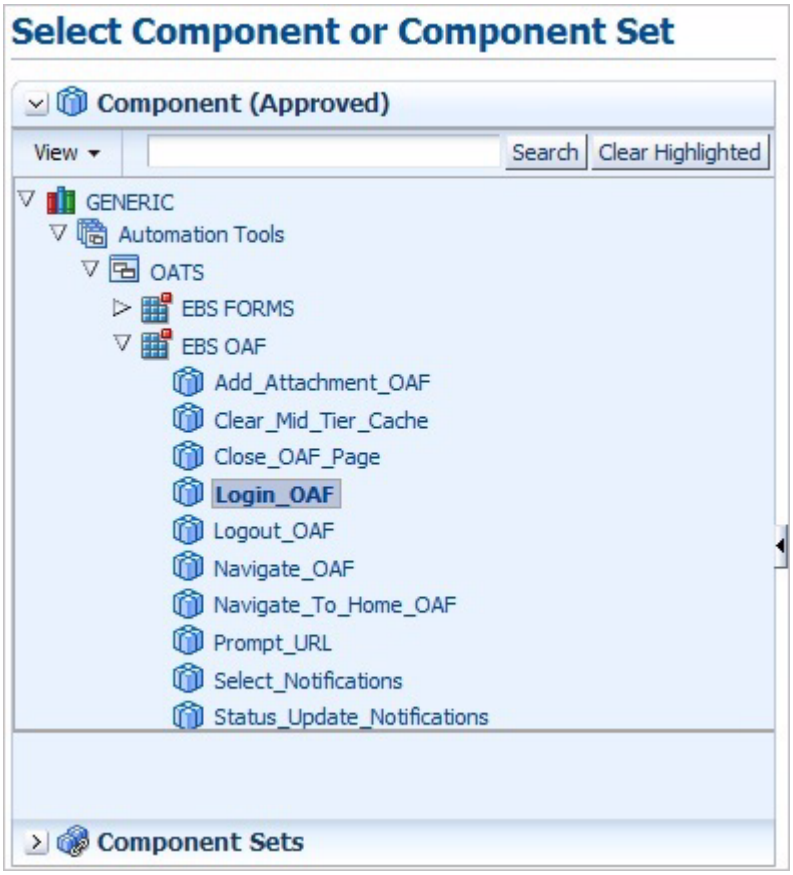
3. The Select Component or Components Set and the New Component Set pane opens for specifying the components or component sets to add to the new component set.

Figure 4–15 New Component Set Pane



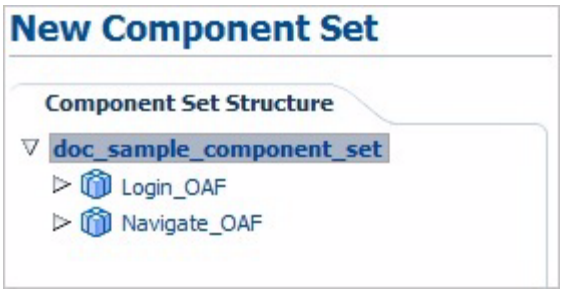
4. Expand the Component or Components Sets tree to view the available components or component sets.

Figure 4–16 Select Component Set Pane with Expanded Components Tree



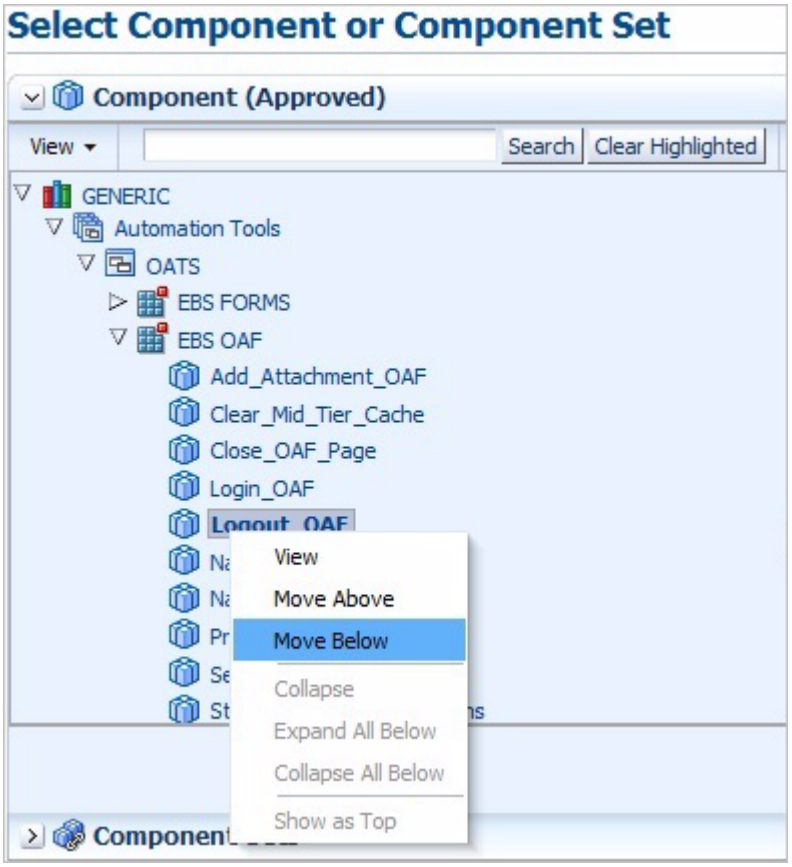
5. Expand the New Component Set tree to view the existing components or component sets.

Figure 4–17 New Component Set Pane with Components Added



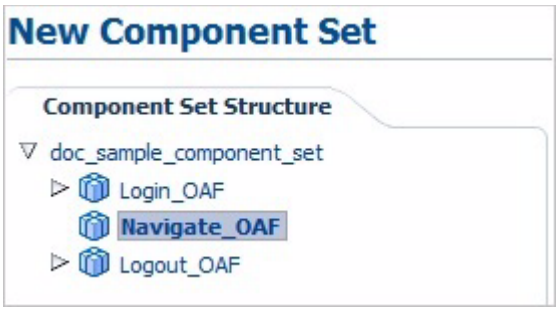
6. Select a component or component set in the New Component Set tree where you want to add the new component or component set.
7. Right-click the component or component set in the Select Component or Component Set tree that you want to add to the New Component Set and click **Move Above** or **Move Below**.

Figure 4–18 Move Options of the Shortcut Menu



8. Repeat step 7 to add additional components or component sets to the New Component Set as needed.

Figure 4–19 New Component Set with Additional Components Added



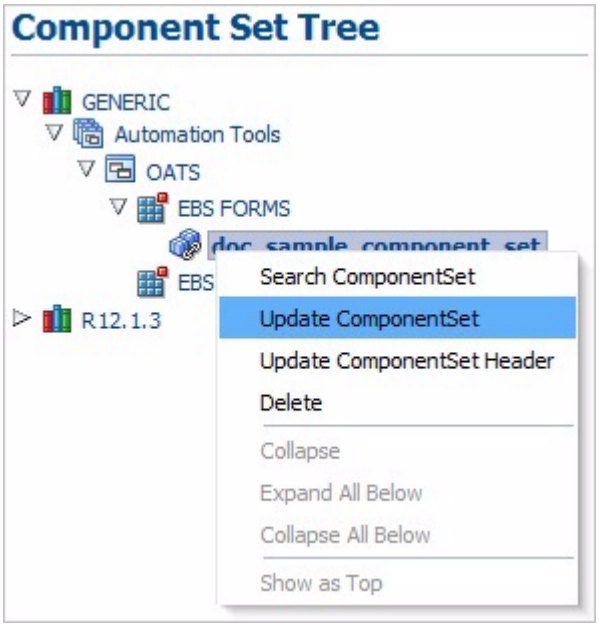
9. Click **Unlock** when finished to save and exit the New Component Set pane. The Search pane indicates *Publish Component Set Successfully.*

4.3.3 Removing Components or Component Set from an Existing Component Set

To remove a component or component set from an existing component set:

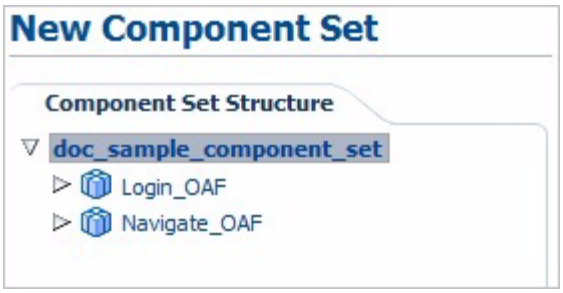
1. Expand the Component Set Tree to the component set you want to update.
2. Right-click the Component Set name and select **Update Component Set** from the shortcut menu.

Figure 4–20 Update Component Set Option of the Shortcut Menu



3. Expand the New Component Set tree to view the existing components or component sets.

Figure 4–21 New Component Set Pane with Components Added



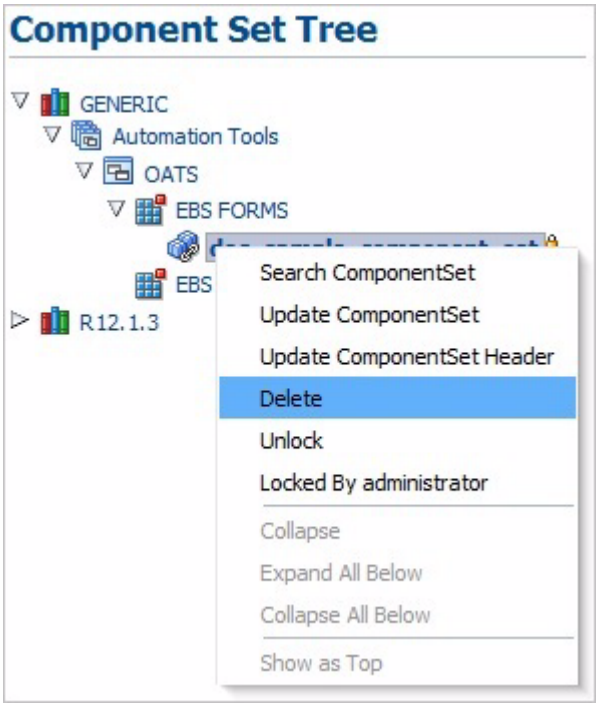
4. Right-click the component or component set in the New Component Set tree that you want to remove from the New Component Set and click **Delete**.
5. Click **OK** to confirm.
6. Repeat steps 4and 5 to remove additional components or component sets from the New Component Set as needed.
7. Click **Unlock** when finished to save and exit the New Component Set pane. The Search pane indicates *Publish Component Set Successfully.*

4.4 Deleting Component Sets

To delete component sets from the Component Set tree:

1. Expand the Component Set Tree to the component set you want to remove from the component set tree.
2. Right-click the Component Set name and select **Delete** from the shortcut menu.

Figure 4–22 Delete Component Set Option of the Shortcut Menu



3. Click **Yes** to confirm.

4. Click **OK**.

5

Defining Flows

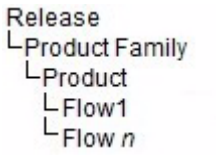
This chapter explains how to add and update flows in the Flow Tree. This chapter contains the following sections:

5.1 Overview

Flows define the test flow and test data used to generate OpenScript scripts for testing application functionality. The Flow Tree represents a library of Flows that specify test instances.

The Flow Tree on the Flows page has the following structure:

Figure 5–1 Flow Tree Structure



A Oracle Flow Builder administrator defines the Release, Product Family, Product, and Feature structure of the Flow Tree. Oracle Flow Builder users add Components to the tree and define the keywords and parameters that will be used to specify Component Sets and Flows that generate the OpenScript scripts used to test the application under test.

Clicking the **Flows** link at the top of the main window shows the Flow Tree and Search Flows options. The default Flow Tree includes two releases, Generic and R12.2, as follows:

Figure 5–2 Default Flow Tree

Generic flows consist of those component sets that group functions that can be used across all product families. Expanding the Generic tree shows the list of products and features contained in the Generic tree:

Figure 5–3 Generic Flow Tree

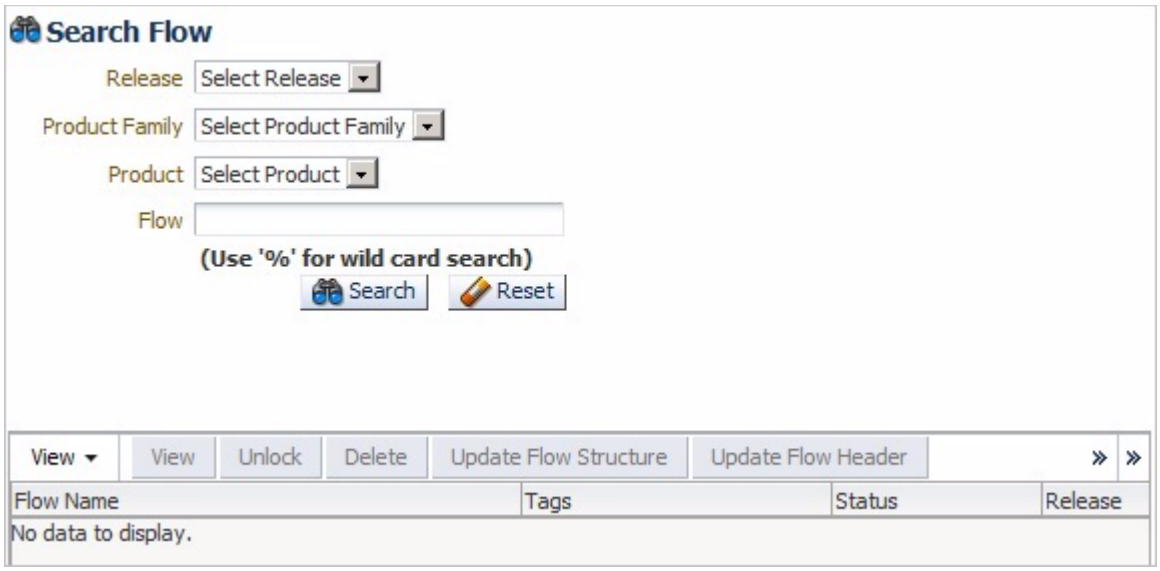
Release and application specific flows are listed below the Release(s) listed in the Flow Tree. Expanding the tree shows the list of products and features in the release tree:

Figure 5–4 Release Flow Tree

Note that the list of available product and features may vary based upon your specific installation.

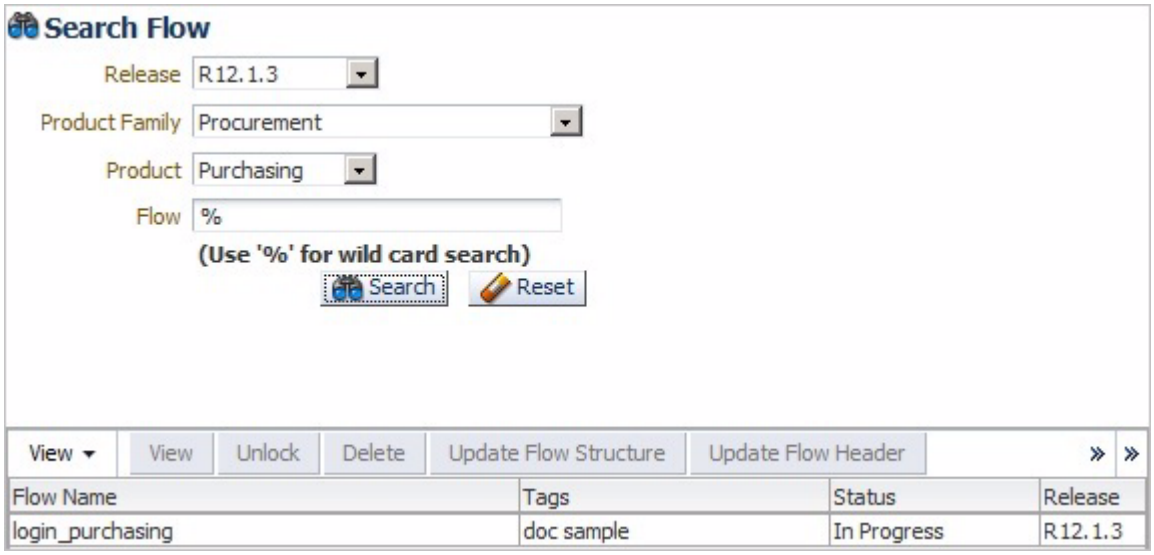
The Search pane of the Flow page lets you search for specific flows:

Figure 5–5 Search Flow Options



You can select the Release, Product Family, Product, and Flow to narrow the search criteria. You can also use the % wildcard character in the Flow field to narrow the search:

Figure 5–6 Search Flow Options with Search Criteria



5.2 Adding Flows

This section explains the procedure for adding flows to the Flows Tree. Flows can be added to the Flows Tree directly in the application UI.

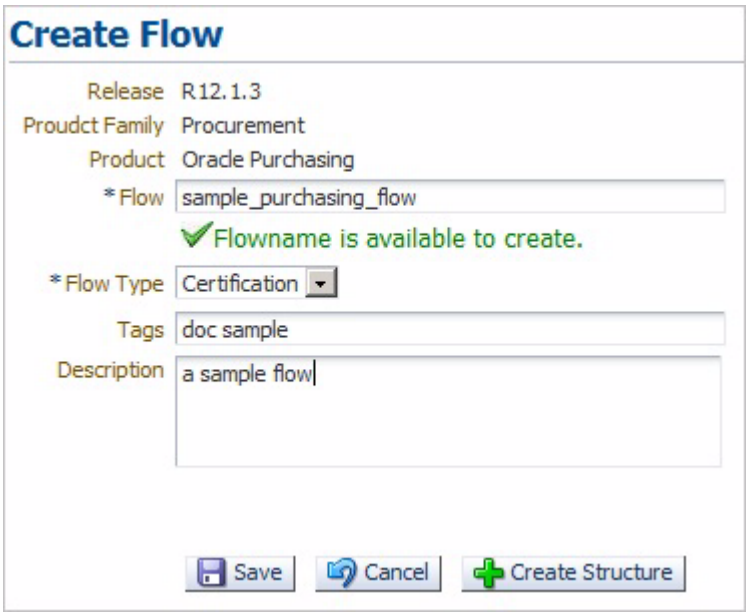
To add a flow to the Flows tree:

1. Expand the Flow Tree to the product feature where you want to add the flow.
2. Right-click the Feature name and select **Create Flow** from the shortcut menu

Figure 5–7 Create Flow Option of the Shortcut Menu

3. In the Create Flow pane, enter a Flow name, Flow Type, Tags and Description.

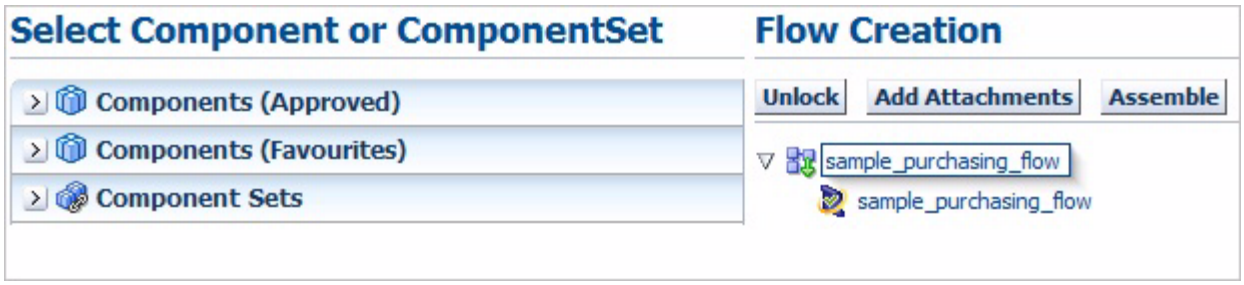
Figure 5–8 Create Flow Pane



The flow name should be between 5 and 30 characters. The *Flow name is available to create* check mark appears if the name does not currently exist in the database.

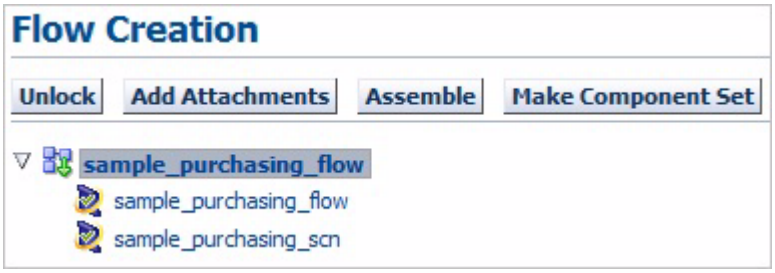
4. Click **Create Structure**. The Select Component or Components Set and the Flow Creation pane opens for specifying the components or component sets to add to the new flow and a new scenario is added to the flow tree. A scenario defines a specific OpenScript child script name within the overall flow. See Section 5.2.1, "Adding Scenarios to a Flow" for additional information about adding scenarios to a flow.

Figure 5–9 Flow Creation Pane



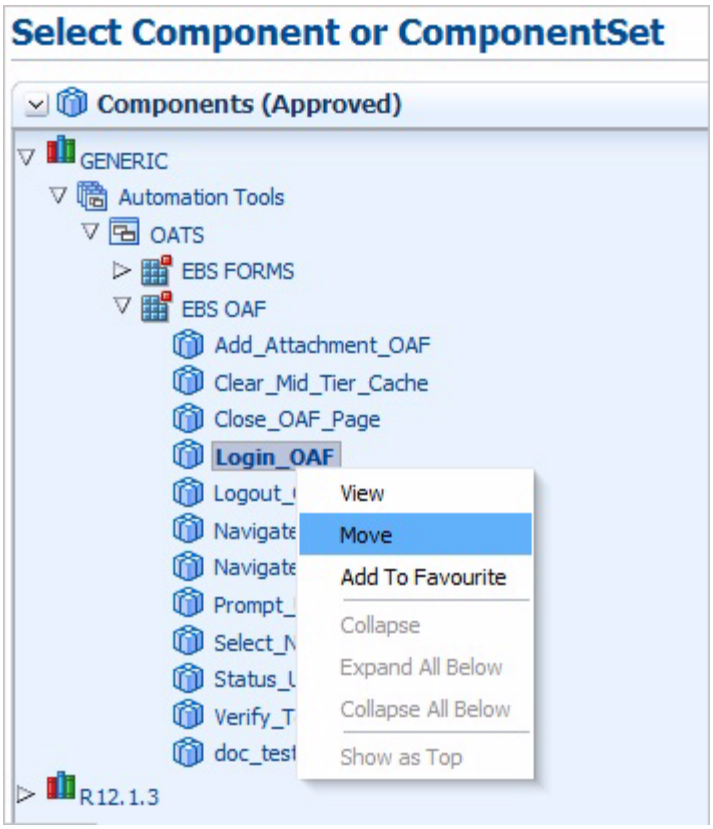
5. Expand the Flow tree and select the scenario where you want to add components or component sets.

Figure 5–10 Expanded Flow Creation Tree



6. Expand the Components (Approved), Components (Favourites), or Components Sets tree to view the available components or component sets.

Figure 5–11 Select Component Set Pane Shortcut Menu

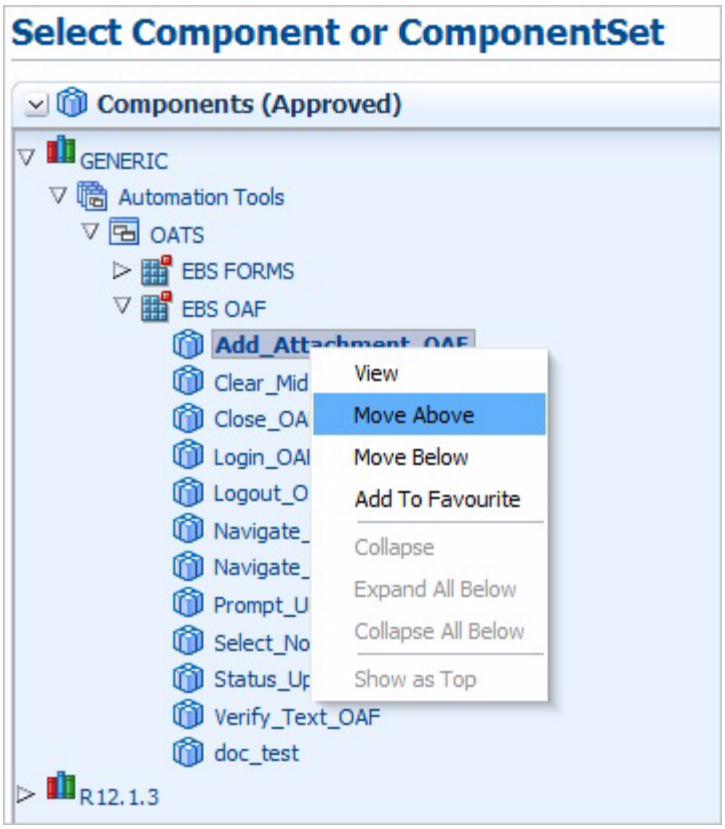


7. Right-click the component or component set to add to the Flow Creation tree and click **Move** (or click and drag the component or component set to the flow tree).

Figure 5–12 Flow Creation Pane with Components Added

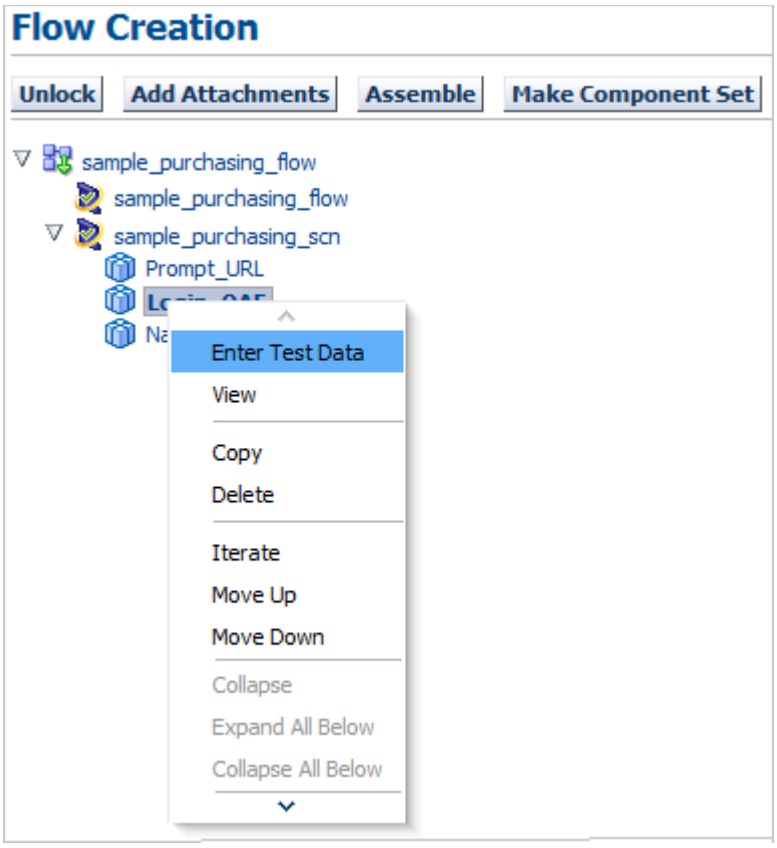
8. Repeat steps 5-7 to add additional components or component sets to the Flow Creation tree as needed. Select **Move Above** or **Move Below** to add the component or component set above or below the currently selected component or component set in the Flow Creation tree.

Figure 5–13 Select Component Pane Shortcut Menu



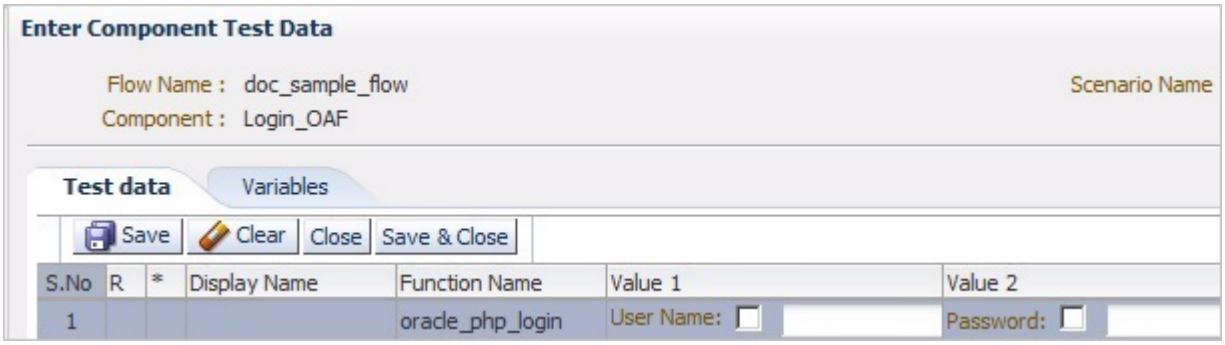
9. Right-click a component in the Flow and select **Enter Test Data** from the shortcut menu. You can also download a Test Data Excel file for the flow, enter the test data for the flow into the Excel file, then upload the Excel file. See Section 5.2.2, "Entering Test Data Using an Excel File" for additional information about entering test data using an Excel file.

Figure 5–14 Flow Creation Pane Enter Test Data Shortcut Menu Option



10. Enter the test data required for the component.

Figure 5–15 Enter Component Test Data Window



The following are guidelines for entering test data for component objects:

- **Rerunable**: For fields marked Rerunnable in the component code, the test data is entered as "?*Value*" in the Value column. For example, the test data entered for Username field would be ?RTUser. Rerunable fields are indicated with an "R" in the R (Rereunable) column in the test data entry screen.
- **Mandatory**: Test data is mandatory for all fields marked as Mandatory=Yes in the component code. Mandatory fields are indicated with an asterisk in the * (Mandatory) column in test data entry screen.

- **Value field checkbox**: Select the checkbox in value fields if the value is to be passed as a variable. When checked, the text field for the value changes to a list of values (LOV) listing all variable names from previous components in the flow. Variable s are appended with sequence number, for example, var1, var2, etc. If data is not provided to any field in "Enter Test Data", the code for that line would not be generated in the OpenScript script.
- **Button objects:** For button object clicks, select True/False. Test data input is required only if the button object keyword is enclosed in Optional tags.
- **Checkbox objects:** For checkbox object clicks, enter True to select the checkbox, False to clear the checkbox. Test data input is required only if the checkbox object keyword is enclosed in Optional tags.
- **Date fields**: For date fields, Test data is entered for different date requirements as follows:

  To empty the value in a Date field enter: #EMPTY

  To specify the date format as DD_MMM_YYYY enter: #RAND(2_3_4)

  To add n days to system date enter: #SYSDATE(+n) Ex. #SYSDATE(+2)

  To subtract n days to system date enter: #SYSDATE(-n) Ex. #SYSDATE(-2)

  To use the System Date enter: #SYSDATE(0)
- **Function calls:** For calls to library functions, the data entered is specific to the function called. See the library reference for additional details.
- **Link objects:** For Link object clicks, select True to click the link. Test data input is required only if the link object keyword is enclosed in Optional tags.
- **Menu selection:** For simple menu selections, enter the menu names separated by the pipe (|) character. For example, File|Open. Test data input is required only if the menu object keyword is enclosed in Optional tags.
- **Navigation (Forms)**: To handle navigation to a form from the forms navigator, enter data as follows in the Navigate component:

  Security,User,Define
- **Navigation (OAF)**: To handle navigation to a form or OAF page with multiple drill menus from the application home page such as: Purchasing, Vision Operations (USA)- Setup-Profile Management Configuration-Organization Encryption, enter data as follows in the Navigate component:

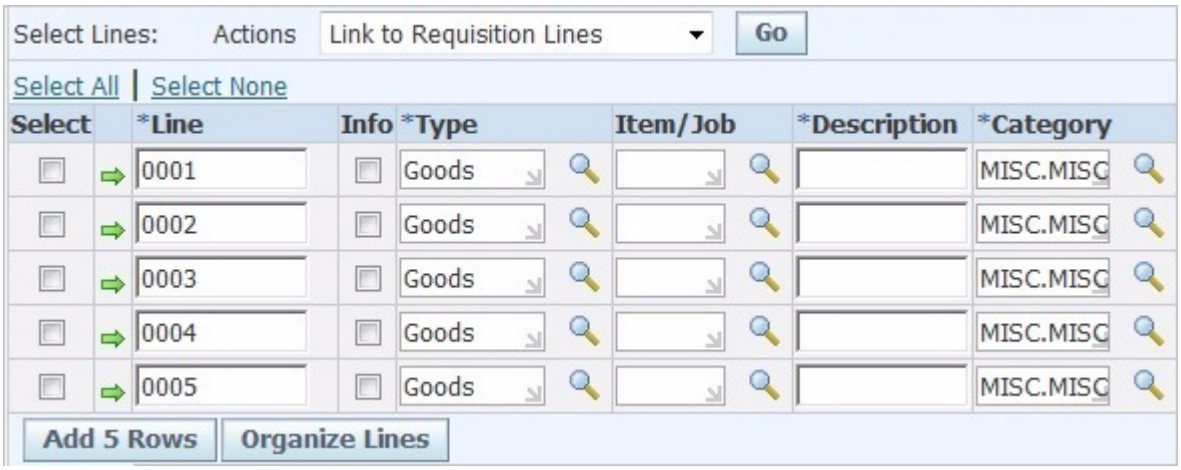
  Resp: Purchasing, Vision Operations (USA)

  Menu Path: Setup,Profile Management Configuration

  Menu Choice: Organization Encryption
- **Radio button objects:** For Radio button object clicks, enter True to select the Radio button, False to clear the checkbox. Test data input is required only if the Radio object keyword is enclosed in Optional tags.
- **Text/Text Area objects:** Enter the value to input into the text/textarea field.
- **Wait**: For Wait for all objects: No test data required. Wait uses the default wait time. For Wait > Normal: No test data required. Wait uses the default wait time.
- **Webtable/Forms table-Entry:** For components containing tables in forms / Web, enter test data for "Enter Line Number" or similar Display Name as the

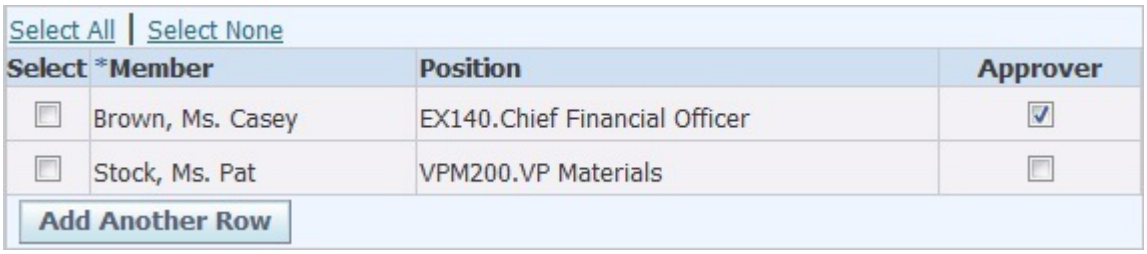
line number where values are to be entered. For example, to enter data for the 1st line in the form or table, enter 1 as the test data.

Figure 5–16 Test Data for Webtable/Forms-Entry



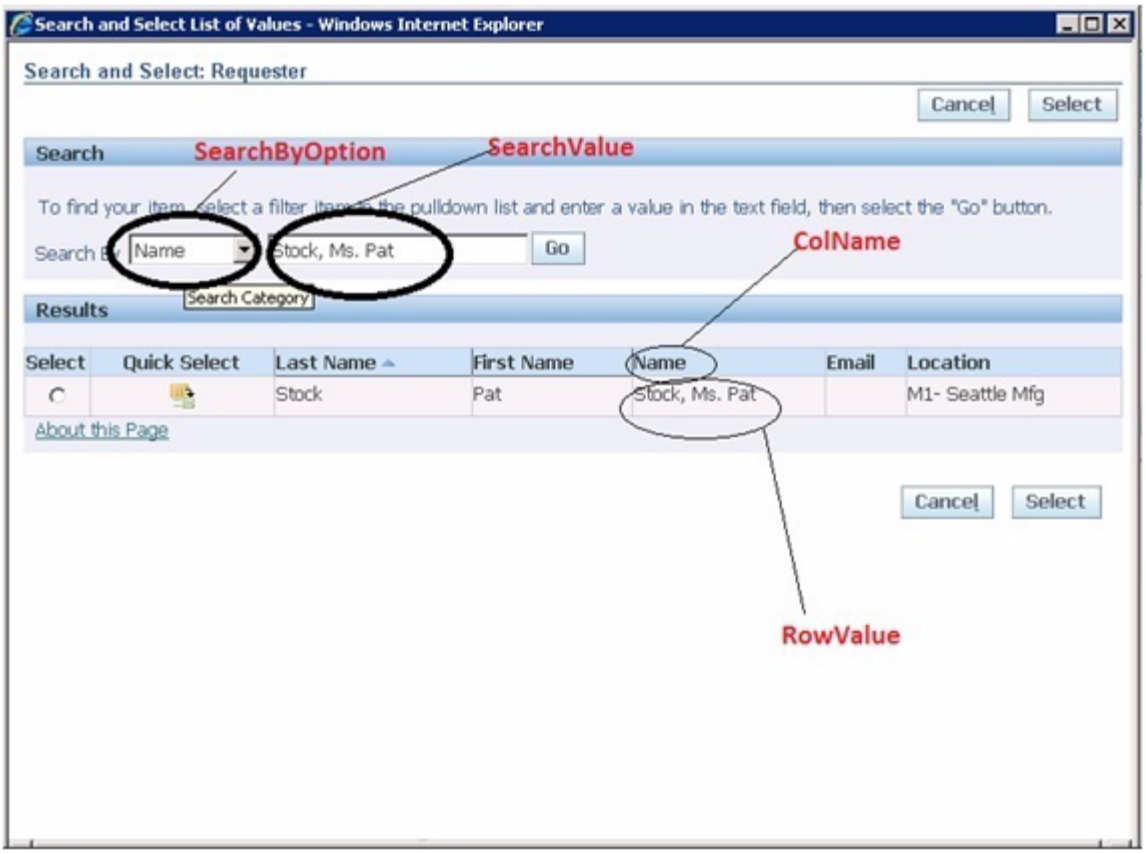
- **Webtable/Forms table-Verify/Update:** For components that require updating a line in a table or performing an action on any element in a table, the test data entered is the unique column name.For example, if the action to perform is check the checkbox in the "Select" column for *Member = Stock, Ms.Pat, then the test data to be entered for the Search column specified as "*Member" is "Stock, Ms Pat".

Figure 5–17 Test Data for Webtable/Forms-Verify/Update



- **WEBSELECTLOV Function**: Test data entry for the WEBSELECTLOV Function corresponds to the following:

Figure 5–18 Test Data Locations for WEBSELECTLOV Function



11. Click **Save and Close** when finished.
12. Repeat steps 9-11 to enter test data into additional components in the flow.
13. Set the status of the flow as necessary for the current state of development:
    - Click **Unlock** at any time when adding components and entering test data to save and exit the Flow Creation pane. The Search pane indicates *Published Flow Successfully.* The Flow is set to In Progress status and can be changed and updated as necessary.
    - Click **Assembled** when finished entering the test data for all the components and exit the Flow Creation pane. The Search pane indicates *Assembled Flow successfully.* The flow is set to Assembled status. When the flow is in Assembled status, the flow cannot be updated or changed. The flow status must be changed to Stabilizing to make any updates. Change the status to Stabilizing by right clicking on the flow name in the flow tree.

      When a flow is in Assembled status, it can be evaluated by an Automation Team member who changes the status to Stabilizing. The automation user can update the flow if required. When in the Stabilizing status, the automation user generates and executes the OpenScript scripts and tests if the scripts generated from the flow are executing properly. The automation user determines if changes are required to update the components in the flow and continue stabilizing the flow. Once the automation user stabilizes the flow completely, an Automation POC verifies the flow and OpenScript scripts and changes the flow status to Completed.

Once the flow is in Completed status, if any user clicks on Create / Update flow structure, the flow changes back to In Progress status.

5.2.1 Adding Scenarios to a Flow

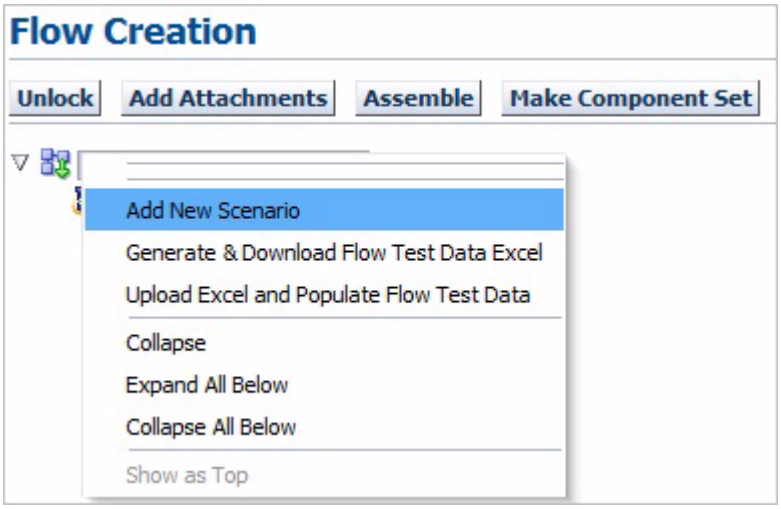
When you create a flow structure, a scenario name is automatically created to define the OpenScript child script name. Each scenario in a flow corresponds to an OpenScript child script. A flow may have only one scenario or it can have multiple scenarios depending upon the flow and the automation process the flow designer is developing.

- The number of scenarios in a flow depends upon the user/automation process the flow designer is developing.
- If the flow is small requiring only a few steps (components or component sets) to complete the flow, only one scenario may be required which will create the MASTERDRIVE script and one OpenScript child script. There is no need to add additional scenarios to the flow.
- If the purpose of the flow is to automate multiple functional scenarios in one flow, then the better practice is to break the flow into multiple scenarios (and create multiple OpenScript child scripts) under one flow.
- If a flow is to automate a lengthy end-to-end scenario, the better practice is to break the flow into multiple simple scenarios to make it easier to execute and troubleshoot the OpenScript child scripts, if necessary.
- OpenScript script code is contained in Java ".class" files. There is a technical limitation that a ".class" file cannot exceed a file size of 65536 Bytes. If a scenario in a flow is too large, it is possible for the generated OpenScript script ".class" file to exceed the file size limitation for one script. Use multiple scenarios to break up larger scenarios (and multiple generated OpenScript scripts) to avoid exceeding file size limitations.

To add scenarios to the flow tree:

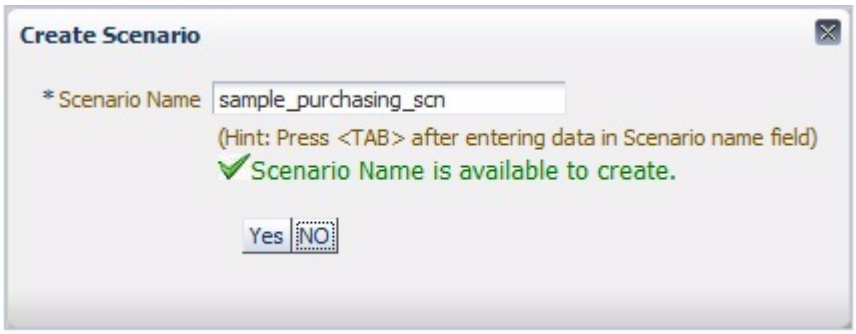
1. Open the flow tree and select the flow name.
2. Right-click the flow name and select **Add New Scenario** from the shortcut menu.

Figure 5–19 Add New Scenario Shortcut Menu



3. Enter a scenario name, then press the **Tab** key. The *Scenario Name is available to create* check mark appears if the name does not currently exist in the database.

Figure 5–20 Create Scenario Options



4. Click **Yes**.
5. Click **OK** to confirm.
6. Add components and/or component sets to the flow tree under the scenario name.

   When you generate the OpenScript scripts, each scenario name will correspond to an OpenScript child script along with the MASTERDRIVE script.

5.2.2 Entering Test Data Using an Excel File

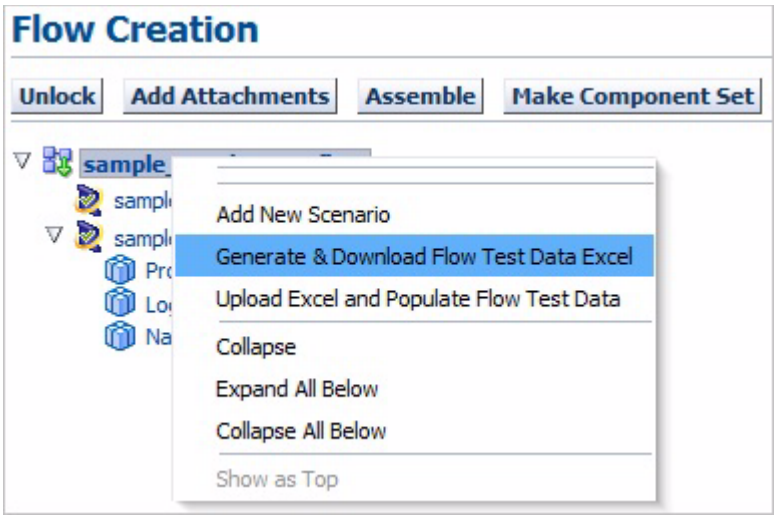
You can enter test data for an entire flow into an Excel file and then upload the Excel file to the flow. The basic steps are as follows:

- Download the test Data Excel file for the flow.
- Enter the test data into the Excel file.
- Upload the test data Excel file to the flow.
- Populate the flow with the test data.

To enter test data for a flow using an Excel file:

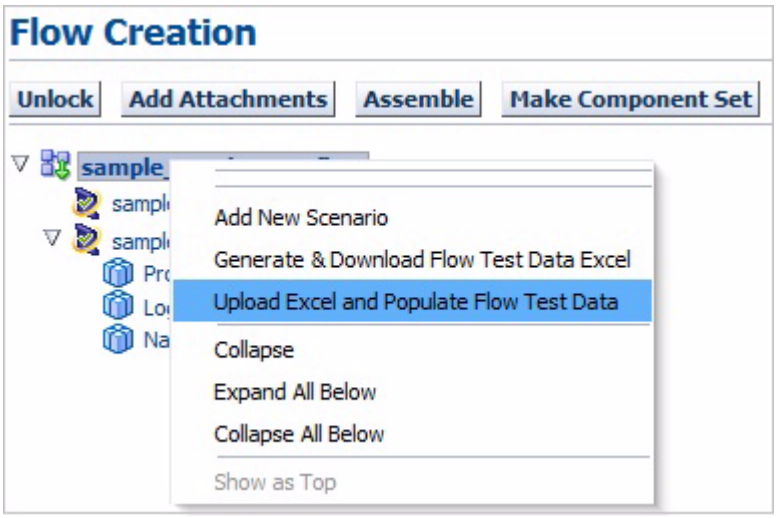
1. Create a new flow or search for an existing flow.
2. Select **Update Flow Structure**.
3. Right-click the flow name in the Flow Creation pane and select **Generate & Download Flow Test Data Excel** from the shortcut menu.

Figure 5–21 Flow Creation Pane Generate & Download Excel Shortcut Menu Option



4. Select **Save** and specify the location to save the Excel file.
5. Edit the Excel file to enter the test data for the flow and save the Excel file.
6. Right-click on the flow name in the Flow Creation pane and select **Upload Excel and Populate Flow Test Data** from the shortcut menu

Figure 5–22 Flow Creation Pane Upload Excel Shortcut Menu Option



7. Select the Excel file for the flow test data and click **Start**. Close the Browse dialog box after successfully uploading the file. The data will be populated to all components in the flow.

5.3 Updating Flows

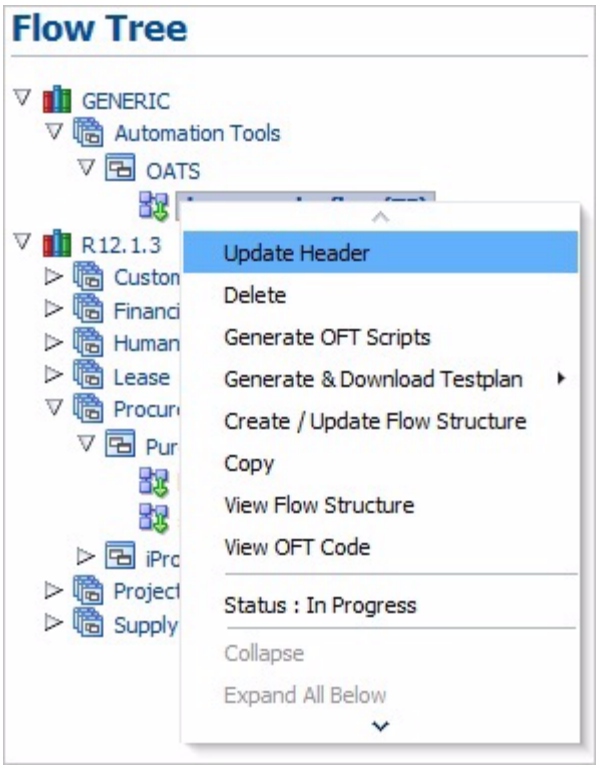
Existing flows can be updated to add or remove components or components sets from a flow or change test data.

5.3.1 Updating Flow Headers

To update a flow header:

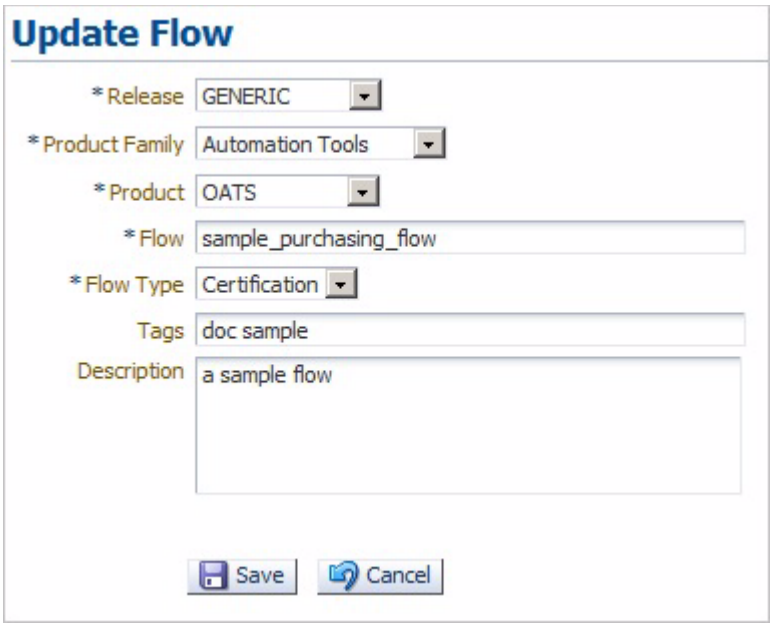
1. Expand the Flow Tree to the flow you want to update.
2. Right-click the flow name and select **Update Header** from the shortcut menu.

Figure 5–23 Update Header Option of the Flow Tree Shortcut Menu



3. In the Update Flow pane, edit the Release, Product Family, Product, Flow name, Flow Type, Tags and Description.

Figure 5–24 Update Flow Pane



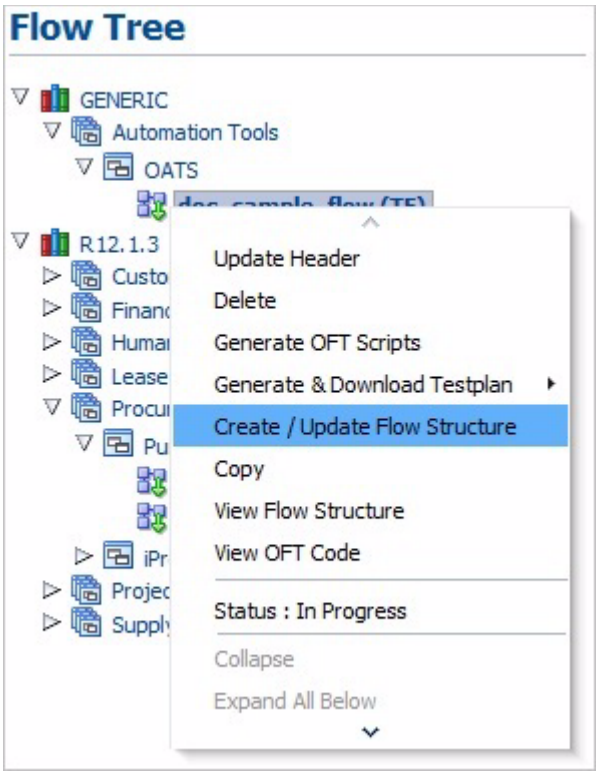
4. Click **Save** when finished to save and exit the Update Component Set pane. The Search pane indicates *Updated Flow <name> Successfully.*

5.3.2 Adding Components or Component Sets to an Existing Flow

To add a component or component set to an existing component set:

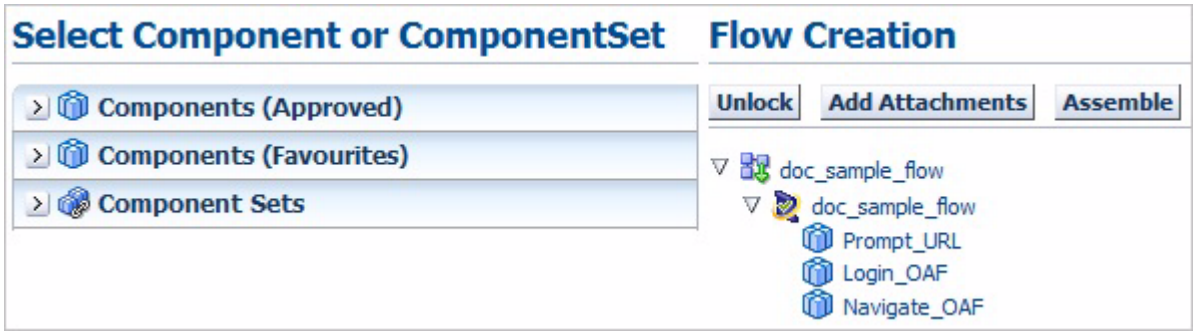
1. Expand the Flow Tree to the flow you want to update.
2. Right-click the Flow name and select **Create/ Update Flow Structure** from the shortcut menu.

Figure 5–25 Create/Update Flow Structure Option of the Flow Shortcut Menu



3. The Select Component or Components Set and the Flow Creation pane opens for specifying the components or component sets to add or update in the Flow.

Figure 5–26 Flow Creation Pane

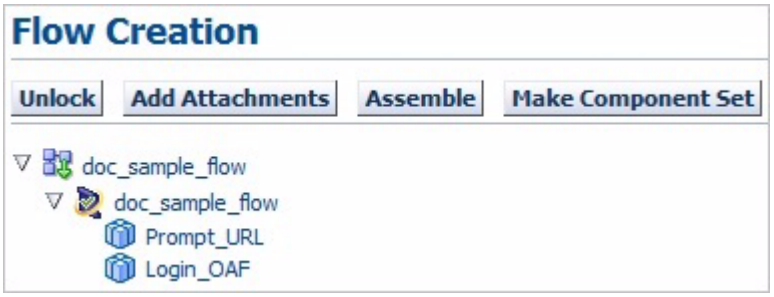


4. Expand the Components (Approved), Components (Favourites), or Components Sets tree to view the available components or component sets.

Figure 5–27 Select Component Set Pane with Expanded Components Tree

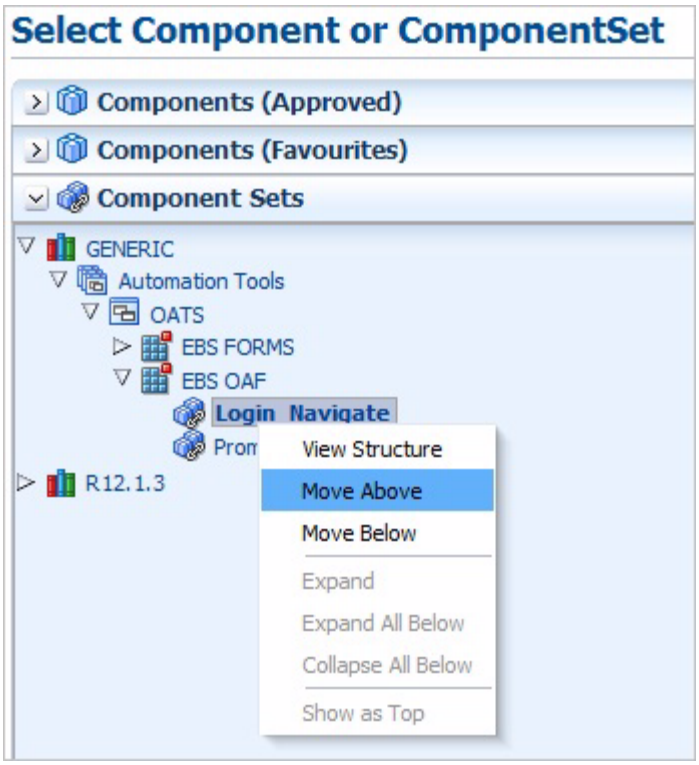
5. Expand the Flow Creation tree to view the existing components or component sets.

Figure 5–28 Flow Creation Pane with Components Added



6. Select a component or component set in the Flow Creation tree where you want to add the new component or component set.
7. Right-click the component or component set in the Select Component (Approved), Select Component (Favourites) or Component Set tree that you want to add to the Flow Creation tree and click **Move Above** or **Move Below**.

Figure 5–29 Move Options of the Shortcut Menu

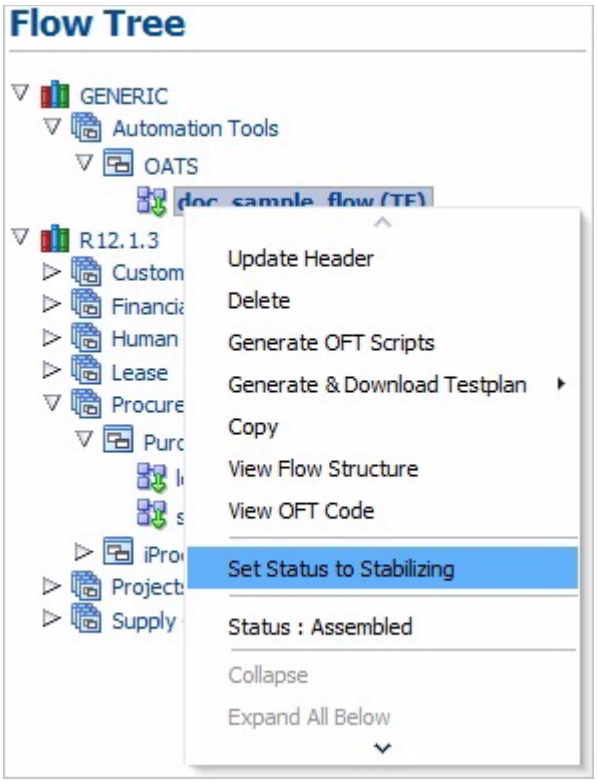


8. Repeat step 7 to add additional components or component sets to the Flow Creation tree Component Set as needed.

Figure 5–30 Flow Creation Pane with Additional Components Added

9. Enter Test Data into the components as needed.
10. Click **Assemble** when finished to change the flow status to Assembled and unlock the flow. The Search pane indicates *Assembled Flow Successfully.*
11. To change the flow to Stabilizing status, right-click the flow name in Flow Tree and select **Set Status to Stabilizing**.

Figure 5–31 Flow Tree Set Status to Stabilizing Shortcut Menu Option

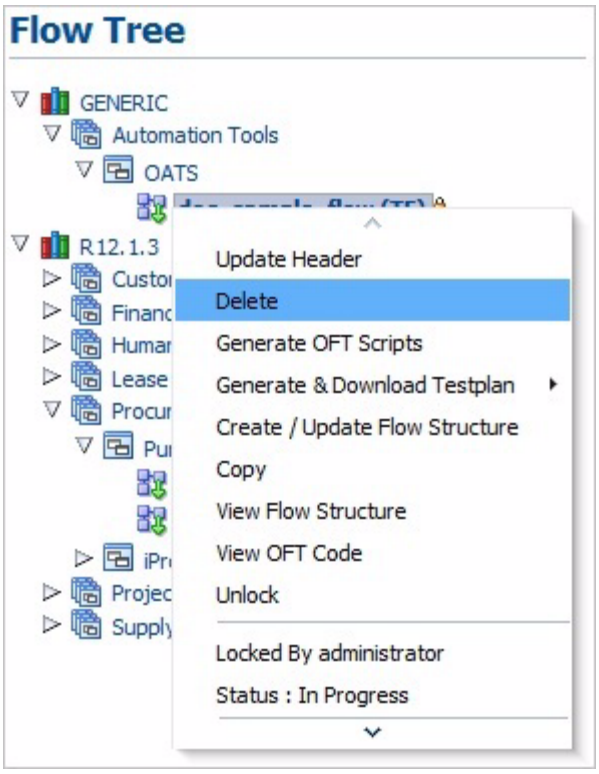


5.4 Deleting Flows

To delete flows from the Flow tree:

1. Expand the Flow Tree to the flow you want to remove from the flow tree.
2. Right-click the Flow name and select **Delete** from the shortcut menu.

Figure 5–32 Delete Flow Option of the Shortcut Menu



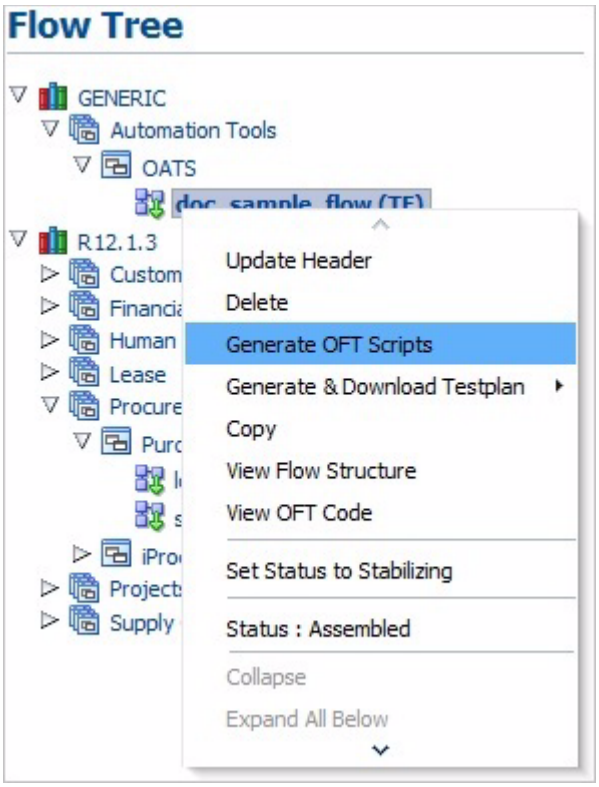
3. Click **Yes** to confirm.
4. Click **OK**.

5.5 Generating OpenScript Scripts

To generate OpenScript script code from an existing flow:

1. Expand the Flow Tree or use the search options to select the flow to use to generate OpenScript code.
2. Select **Generate OFT Scripts** from the Flow Tree shortcut menu or the search pane.

Figure 5–33 Generate OFT Scripts Option of the Shortcut Menu



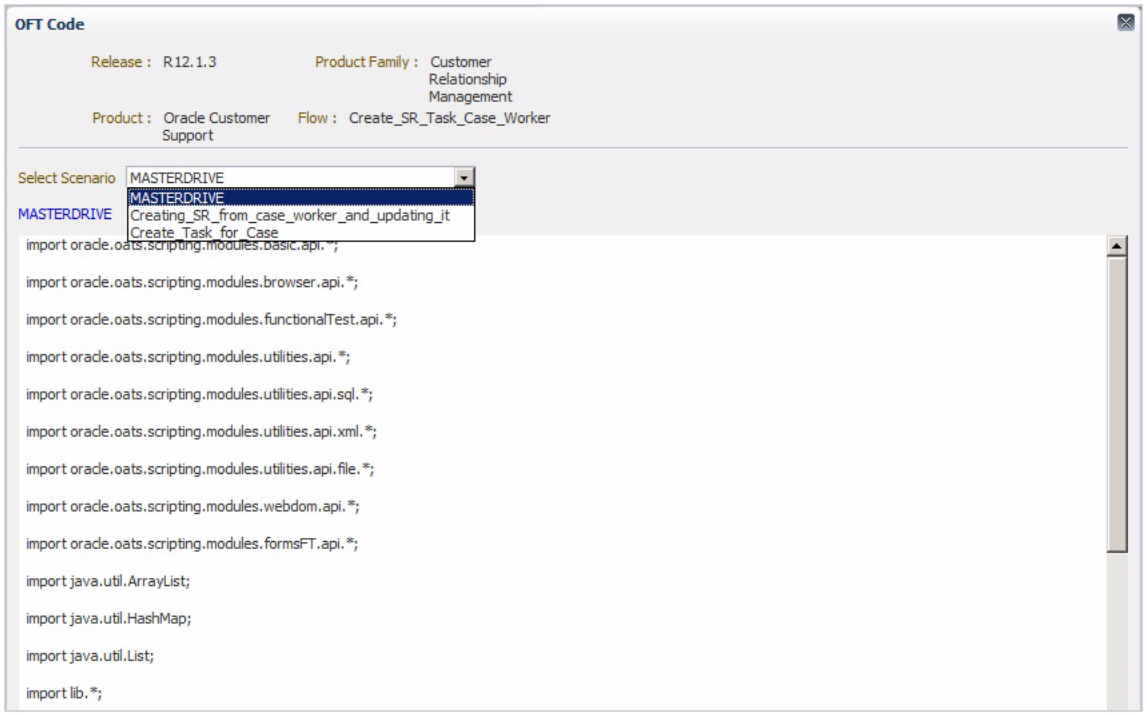
3. Enter the Execution location folder.
4. Enter the Normal wait time.
5. Enter the Page wait time.
6. Select if the script is rerunable or not.
7. If Rerunable, select the time zone, specify the Rerun type.
8. Select the time zone.
9. Click **Download Suite**.
10. Extract the zip file into the OpenScript repository.
11. Start OpenScript.
12. Open and playback the MASTERDRIVE script located in the *flowname* folder. The MASTERDRIVE script executes the script containing the OpenScript code for the flow. You can also open the script with the flow code in OpenScript to view the Tree View and Java Code.

5.6 Viewing OpenScript Code

To view OpenScript script code from an existing flow:

1. Expand the Flow Tree or use the search options to select the flow to use to view OpenScript code.
2. Select **View OFT Code** from the Flow Tree shortcut menu.

Figure 5–34 View Code Window Showing OpenScript Code for a Flow



3. Select the MASTERDRIVE or flow scenario name from the Select Scenario list to view the OpenScript code for the specific flow.
4. Click the **[X]** button to close the code view when finished.

6

Using Notifications

This chapter explains how to use the Notifications options in the Oracle Flow Builder application. This chapter contains the following sections:

6.1 Overview

Notifications provide status information to users and administrators using the Oracle Flow Builder application. Notifications consist of two types: **To Do** and **FYI**.

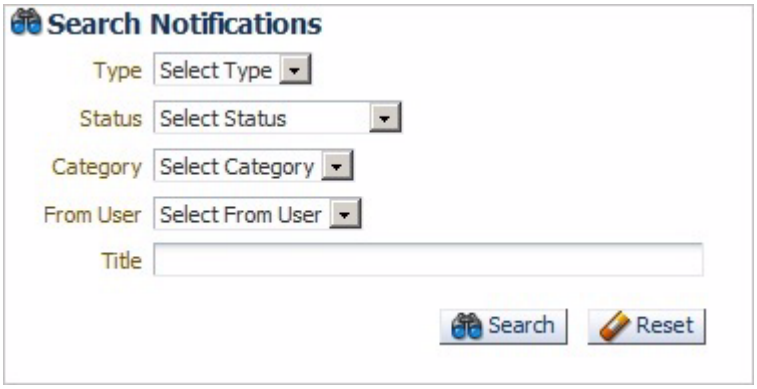
- To Do notifications: include notifications such as *Approved*, *Pending for Approval*, *Rejected*, and *Self Approved*. These notifications pertain to Components, User creation, or Product Family Access (PFAccess).
- FYI notifications: include notifications such as grant of Product Family Access, approval or rejection of components, and user approval.

Clicking the **Notifications** link at the top of the main window shows the Search Notifications options.

6.2 Searching Notifications

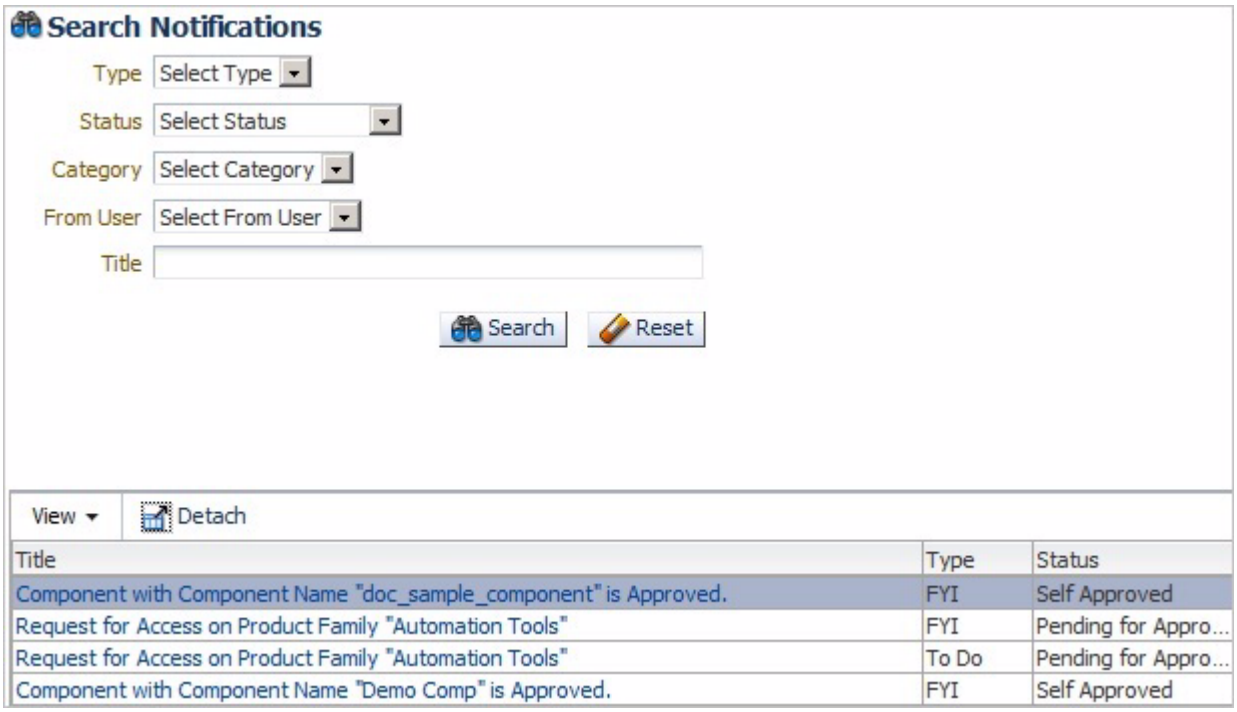
The Search pane of the Notifications page lets you search for specific notifications:

Figure 6–1 Search Notifications Options



Select the Release, Status, Category, User, and Title (or use % wildcard) to narrow the search criteria and click **Search** to view the notifications:

Figure 6–2 Search Notifications with Search Criteria



Click the notification title to show the details.

6.3 Viewing Notification Details

Notification details provide information about the notification.

To view notification details:

1. From the Notifications page, enter the search criteria and click **Search**.
2. Click the Notification title to view the details.
   - Component notifications: details show the Component Details and Component Code Details. Component Details shows the Release, Product Family, Product, Feature, and Component name information.

Figure 6–3 Component Details

For Component notifications in a *Pending for Approval* status, the Component Details include options to **Accept** or **Reject** the component and **Find Usages** to locate where a component is used in a flow.

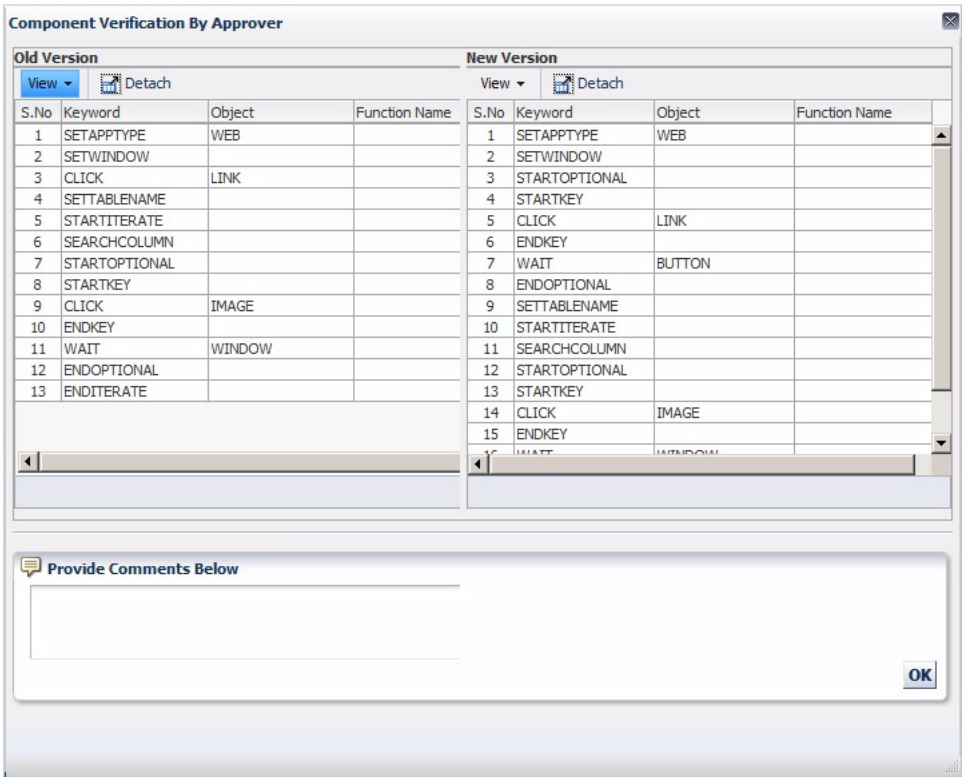
Figure 6–4 Component Details Actions

Component Code Details provides a link to open the component code.

Figure 6–5 Component Code Details

Clicking the link opens the Code Verification window showing the old and new versions of the component code. The approver reviews the code, enters any comments in the comments section and clicks **OK** to approve the code.

Figure 6–6 Component Verification Window



- PFAccess notifications: details show the Product Family Access Details including the User Name, Product Family, Email, and Application Role information and Product Family Role Details show the user role.

Figure 6–7 Product Family Access Details

For PFAcces notifications in a *Pending for Approval* status, the Product Family Access Details include options to **Accept** or **Reject** the access.

Figure 6–8 Product Family Access Details Actions

- User notifications: details show the User Registration Access Details including the User Name, Full Name, Email and User Application Role Details.

Figure 6–9 User Access Details

Select the Application Role.

For User Registration notifications in a *Pending for Approval* status, the User Registration Access Details include options to **Accept** or **Reject** the access.

Figure 6–10 User Registration Access Details Actions

Accept or reject the User Registration.

7

Using History

This chapter explains how to use the History options in the Oracle Flow Builder application. This chapter contains the following sections:

7.1 Overview

The History page lets you search the history of changes in components, component sets, flows and users.

Clicking the **History** link at the top of the main window shows the History categories and the Search History options.

Figure 7–1 History Options

Select the history category and specify the search criteria.

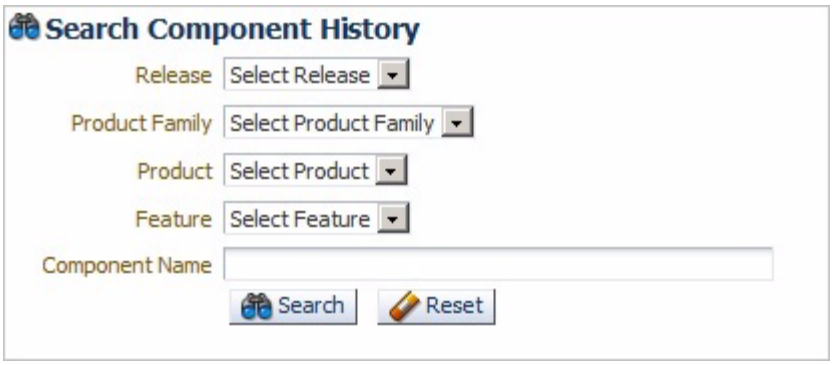
7.2 Searching and Viewing Component History

Selecting the Component category of the History page shows the Component search options. The Search Component History pane of the History page lets you search for the history of specific components:

To view Component history:

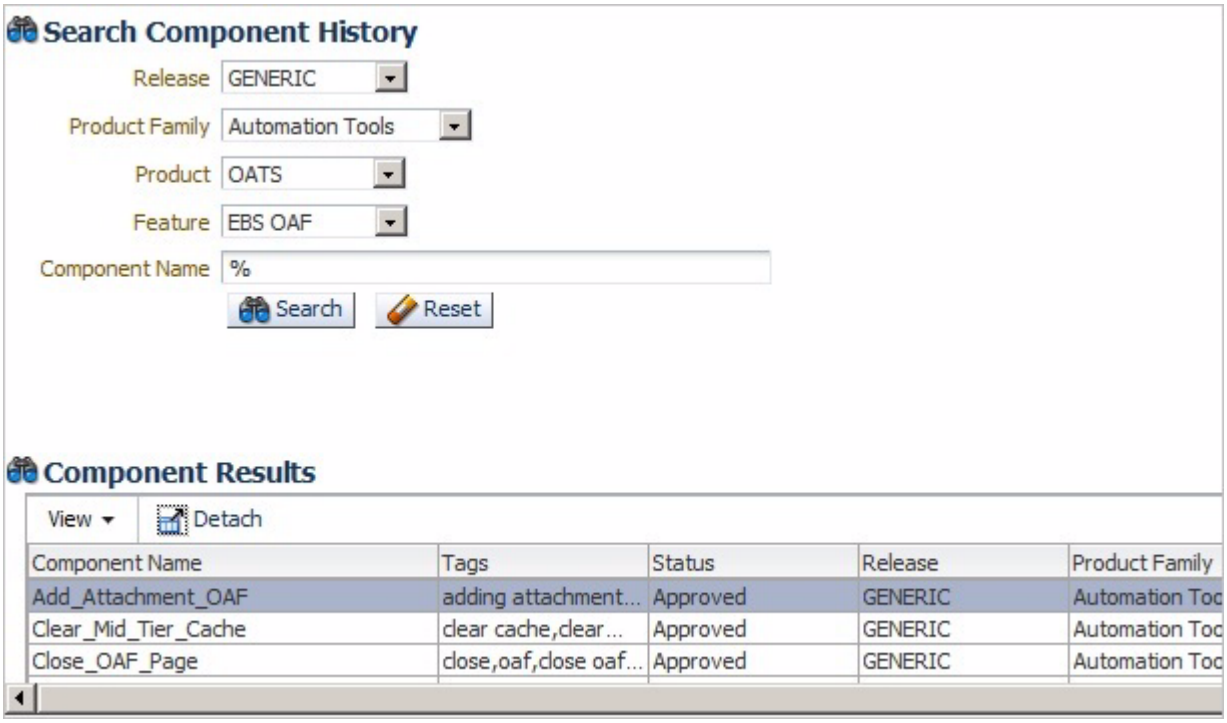
1. Log in and go to the History page.
2. From the **History** categories, select **Components**.

Figure 7–2 Search Component History Options



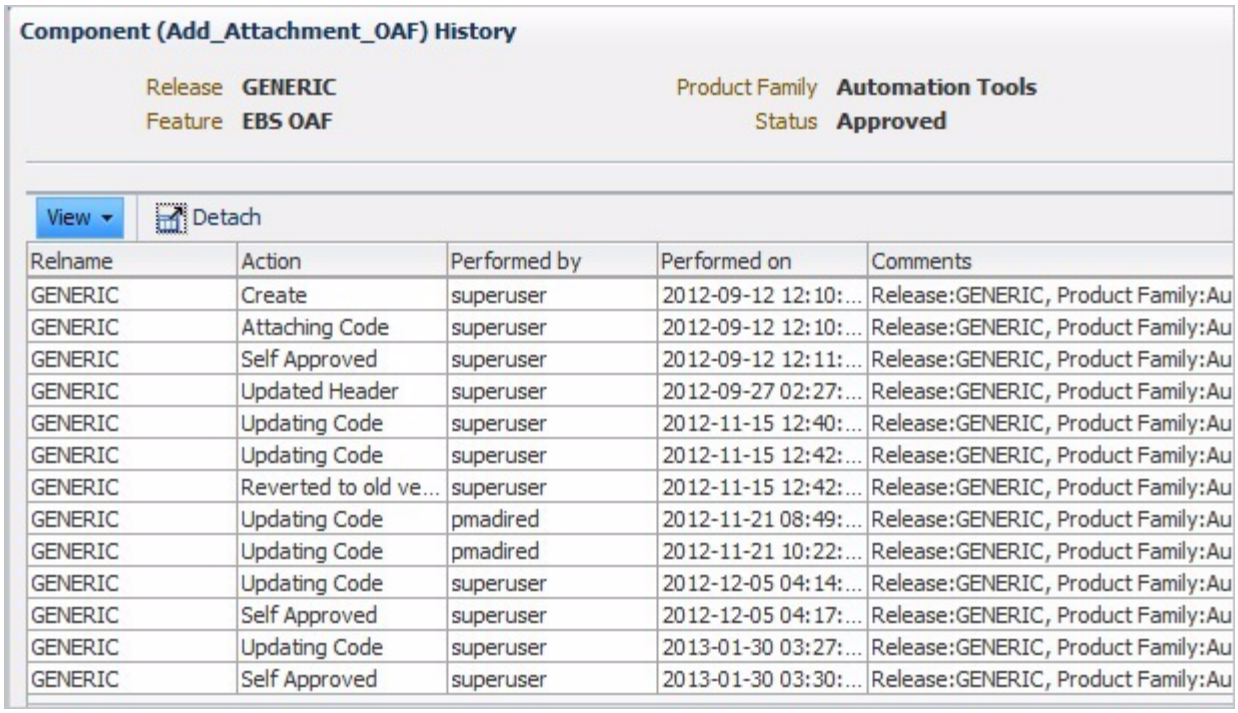
3. Select the Release, Product Family, Product, Feature, and Component name (or use % wildcard) to narrow the search criteria and click **Search** to view the history of the component:

Figure 7–3 Search Component History with Search Criteria



4. Click the **View History** link in the View History column to show the details.

Figure 7–4 Component History Details



Component (Add_Attachment_OAF) History

Release **GENERIC** Product Family **Automation Tools**
Feature **EBS OAF** Status **Approved**

View ▾ Detach

Relname	Action	Performed by	Performed on	Comments
GENERIC	Create	superuser	2012-09-12 12:10:...	Release:GENERIC, Product Family:Au
GENERIC	Attaching Code	superuser	2012-09-12 12:10:...	Release:GENERIC, Product Family:Au
GENERIC	Self Approved	superuser	2012-09-12 12:11:...	Release:GENERIC, Product Family:Au
GENERIC	Updated Header	superuser	2012-09-27 02:27:...	Release:GENERIC, Product Family:Au
GENERIC	Updating Code	superuser	2012-11-15 12:40:...	Release:GENERIC, Product Family:Au
GENERIC	Updating Code	superuser	2012-11-15 12:42:...	Release:GENERIC, Product Family:Au
GENERIC	Reverted to old ve...	superuser	2012-11-15 12:42:...	Release:GENERIC, Product Family:Au
GENERIC	Updating Code	pmadired	2012-11-21 08:49:...	Release:GENERIC, Product Family:Au
GENERIC	Updating Code	pmadired	2012-11-21 10:22:...	Release:GENERIC, Product Family:Au
GENERIC	Updating Code	superuser	2012-12-05 04:14:...	Release:GENERIC, Product Family:Au
GENERIC	Self Approved	superuser	2012-12-05 04:17:...	Release:GENERIC, Product Family:Au
GENERIC	Updating Code	superuser	2013-01-30 03:27:...	Release:GENERIC, Product Family:Au
GENERIC	Self Approved	superuser	2013-01-30 03:30:...	Release:GENERIC, Product Family:Au

List of actions that may appear in history of components are as follows:

- **Create**: indicates the component was created for the first time.
- **Updated Header**: indicates updating of the component header.
- **Update Code**: indicates updating of component code.
- **Reverted to Old Version**: indicates changes made but not submitted and the component keyword code was reverted to the previously approved component code.
- **Submit for Approval**: indicates component keyword code was submitted for approval by the Product Family Access owner.
- **Attached Code**: indicates that component code was attached after component creation.
- **Self Approved**: indicates self approval of the component (applicable for Product Family Access owner).
- **Rejected**: indicates rejection of the component (applicable for Product Family Access owner).
- **Delete**: indicates deletion of the component (applicable for Product Family Access owner).
- **Approved**: indicates approval of the component.
- **Saved And Unlocked**: indicates component code was saved and unlocked.

5. Click the Window close button when finished.

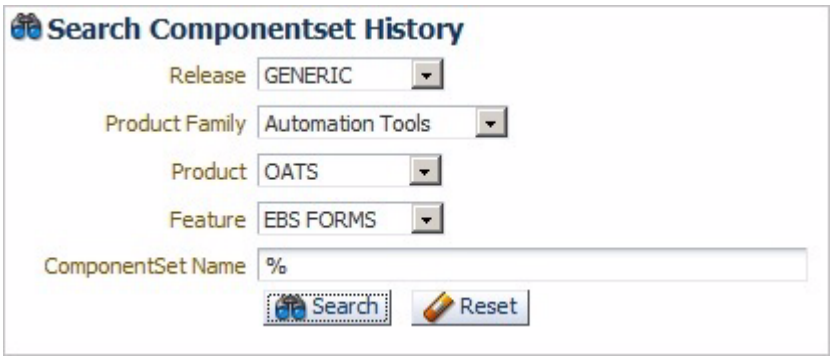
7.3 Searching and Viewing Component Set History

Selecting the Component Set category of the History page shows the Component Set search options. The Search Component Set History pane of the History page lets you search for the history of specific component sets:

To view Component Set history:

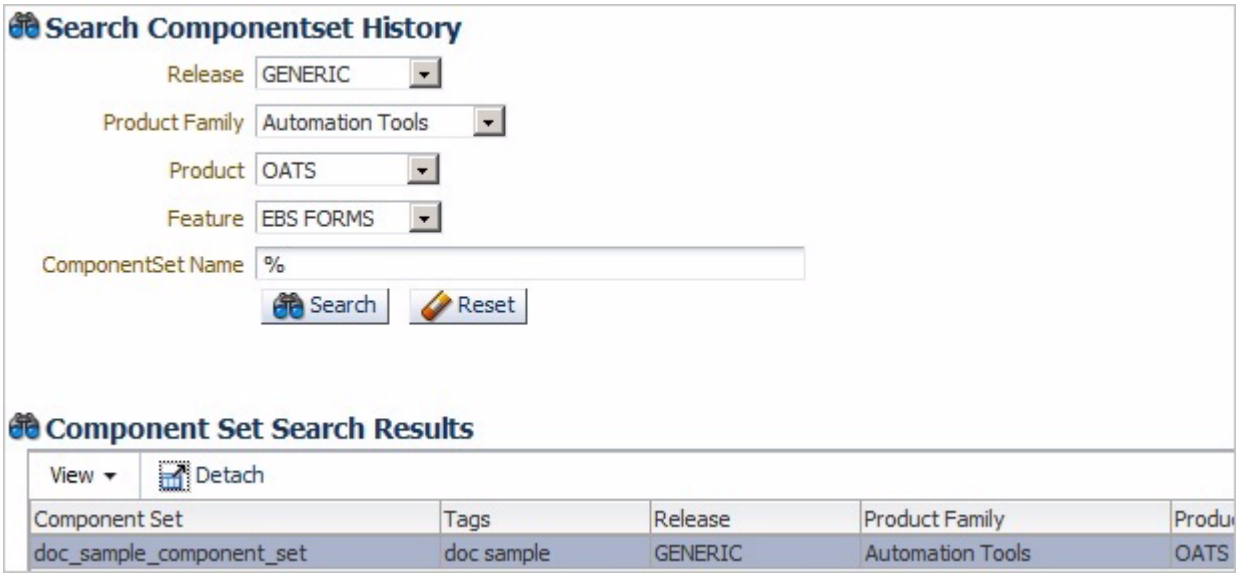
1. Log in and go to the History page.
2. From the **History** categories, select **Component Set**.

Figure 7–5 Search Component Set History Options



3. Select the Release, Product Family, Product, Feature, and Component Set name (or use % wildcard) to narrow the search criteria and click **Search** to view the history of the component set:

Figure 7–6 Search Component Set History with Search Criteria



4. Click the **View History** link in the View History column to show the details.

Figure 7–7 Component Set History Details

List of actions that may appear in history of component sets are as follows:

- **Create**: indicates the component set was created for the first time.
- **Updated Structure**: indicates modification of the component set structure.
- **Update**: indicates updating of component set header.

5. Click the Window close button when finished.

7.4 Searching and Viewing Flow History

Selecting the Flows category of the History page shows the Flow search options. The Search Flow History pane of the History page lets you search for the history of specific flows:

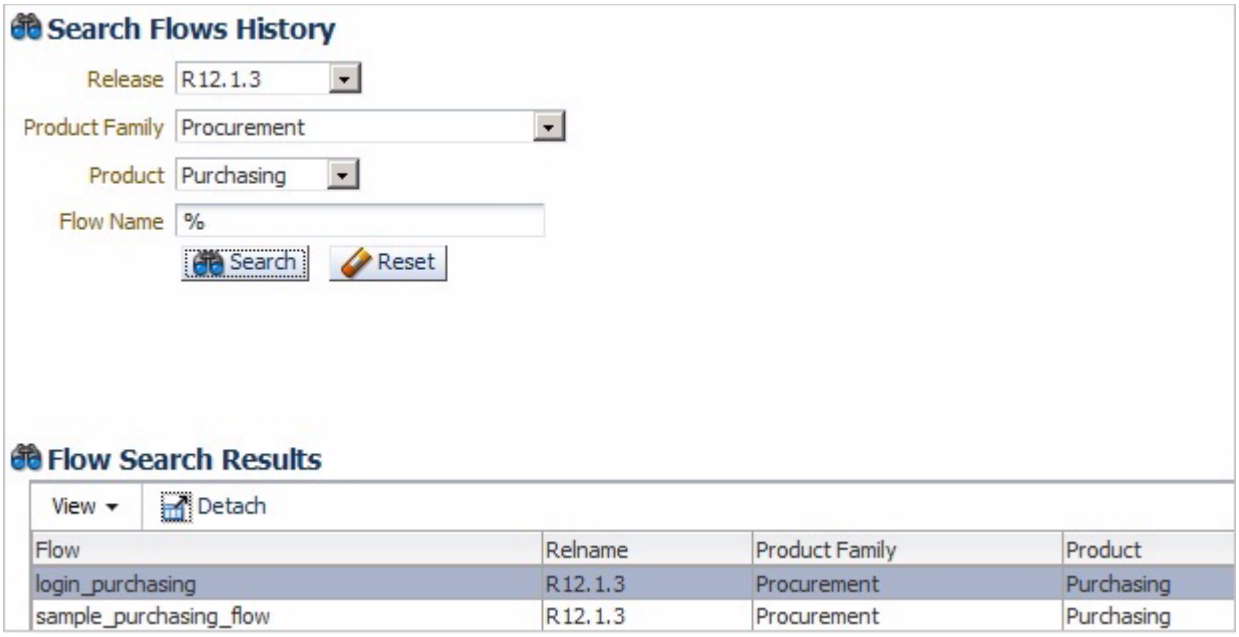
To view Flow history:

1. Log in and go to the History page.
2. From the **History** categories, select **Flows**.

Figure 7–8 Search Flows History Options

3. Select the Release, Product Family, Product, and Flow Name (or use % wildcard) to narrow the search criteria and click **Search** to view the history of the flow:

Figure 7–9 Search Flows History with Search Criteria



4. Click the **View History** link in the View History column to show the details.

Figure 7–10 Flow History Details

List of actions that may appear in history of flows are as follows:

- **Create**: indicates the flow was created for the first time.
- **Create Structure**: indicates the flow was created and the flow status is in progress.
- **Update Structure**: indicates updating of the flow structure.
- **Reverted to Old Version**: indicates changes made but not submitted and the component keyword code was reverted to the previously approved component code.
- **Assembled the Flow**: indicates the flow is completed and ready for stabilization.
- **Unlock**: indicates the flow is in progress and unlocked for other users to update.
- **Self Approved**: indicates self approval of the component (applicable for Product Family Access owner).

- **Delete**: indicates deletion of the flow (applicable for Product Family Access owner).
- **Stabilizing the Flow**: indicates the flow is in the process of stabilization.
- **Completed the Flow**: indicates completion of flow (applicable for Product Family Access owner).

5. Click the Window close button when finished.

7.5 Searching and Viewing User History

Selecting the User History category of the History page shows the User search options. The Search User History pane of the History page lets you search for the history of specific users:

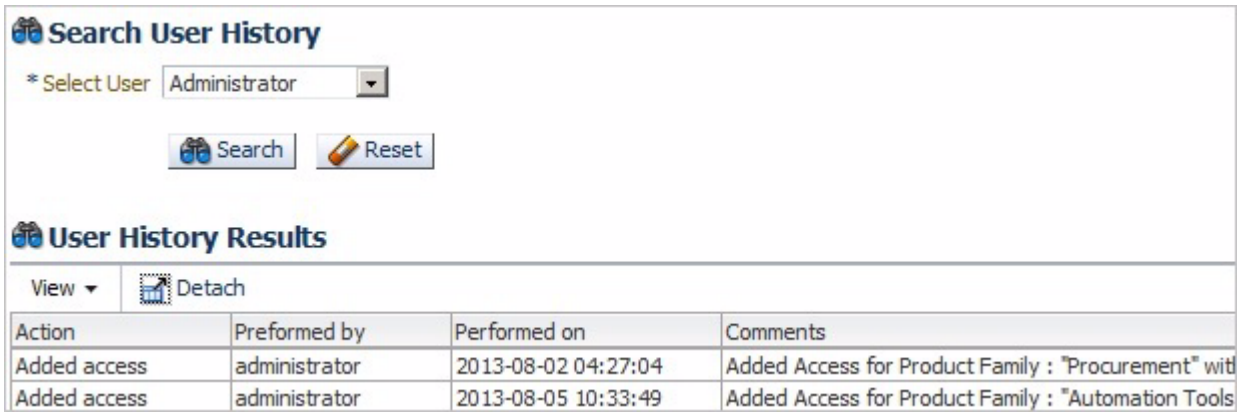
To view User history:

1. Log in and go to the History page.
2. From the **History** categories, select **User History**.

Figure 7–11 Search User History Options

3. Select the User to narrow the search criteria and click **Search** to view the history of the user:

Figure 7–12 Search Users History with Search Criteria



List of actions that may appear in history of users are as follows:

- **Create**: indicates the user was created for the first time.
- **Added access**: indicates adding a specific product family access.
- **Updated access**: indicates updating product family access.
- **Deleted access**: indicates deletion of access to a product family.
- **Update**: indicates updating of user details.

- **Approved**: indicates approval of user registration.

8

Using Reports

This chapter explains how to use the Report options in the Oracle Flow Builder application. This chapter contains the following sections:

8.1 Overview

Reports provide statistical information about components, component sets, and flows to users and administrators using the Oracle Flow Builder application. Reports consist of the following types:

- **Component Report**: shows the number of components by status in the selected Release, Product Family, Product, or Feature.
- **Component Sets Report**: shows the total number of components sets in the selected Release, Product Family, Product, or Feature.
- **Flows Report**: shows the number of flows by status and type in the selected Release, Product Family, Product, or Feature.

The data in generated reports can be viewed in table and graphical formats and the data can be exported to Excel spreadsheet files.

Clicking the **Reports** link at the top of the main window shows the Search Report options.

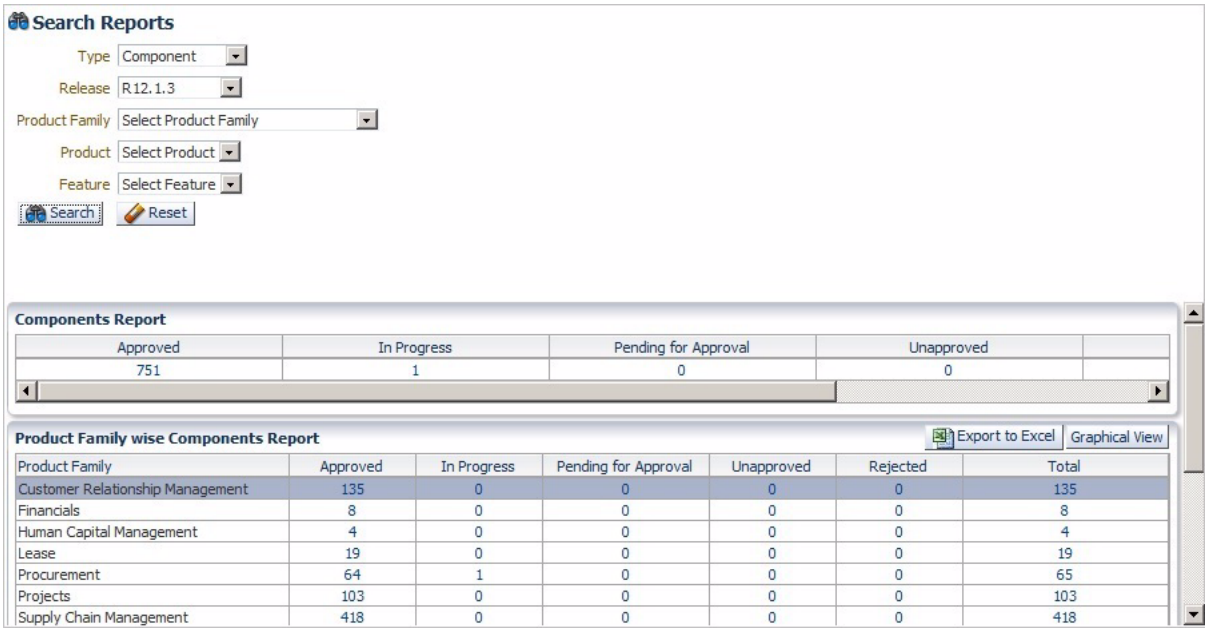
8.2 Searching and Generating Reports

The Search pane of the Reports page lets you search for and generate reports for specific components, component sets, and flows by Release, Product Family, Product, or Feature:

Figure 8–1 Search Report Options

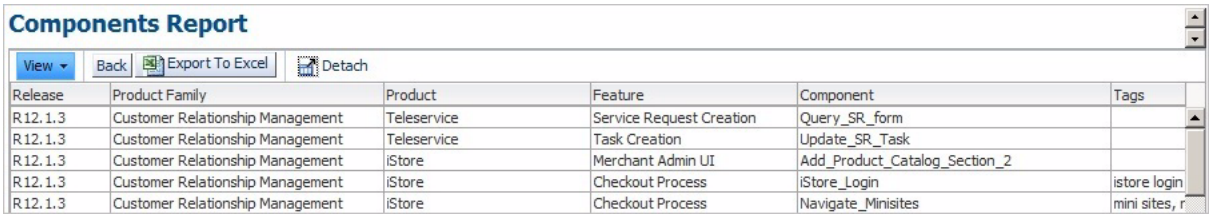
Select the Type, Release, Product Family, Product, and Feature to narrow the search criteria and click **Search** to generate the report:

Figure 8–2 Search Reports with Search Criteria and Report Data



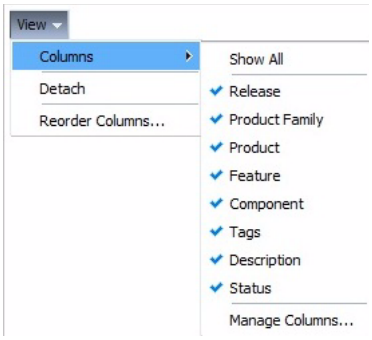
Click a value in a report table to show the report details in table format or click **Graphical View** to show the report details in graphical format. Figure 8–3 shows a sample components report in table format:

Figure 8–3 Sample Components Report in Table View



Use the **View** menu options to show or hide columns, detach the report to a separate view, or reorder the columns.

Figure 8–4 Report View Menu Options



Click **Back** to return to the search pane.

Figure 8–5 shows a sample Product Family report in graphical view:

Figure 8–5 Sample Components by Product Family Report in Graphical View

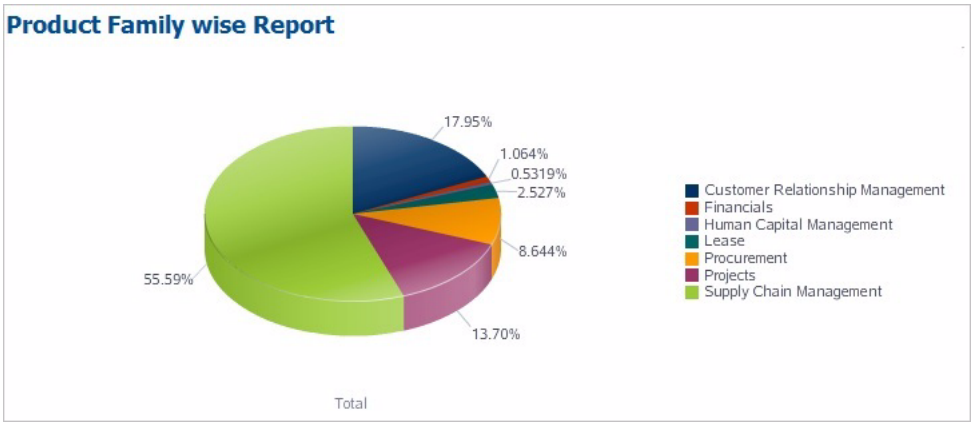


Figure 8–6 shows a sample Component Status report in graphical view:

Figure 8–6 Sample Component Status Report in Graphical View

Placing the mouse cursor over an element in the graph shows the data details.

8.3 Generating Component Reports

Component reports show the number of Approved, In Progress, Pending for Approval, Unapproved, and Rejected components and the total number of components in the selected Release, Product Family, Product, and Feature.

The following sections explain how to generate specific types of component reports:

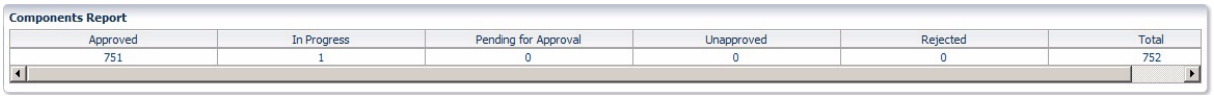
8.3.1 Generating a Component Totals Report

A component totals report shows the total number of components in the Oracle Flow Builder database.

To generate a component totals report:

1. Log in and go to the Reports page.
2. Select **Component** as the search **Type** setting.
3. Leave or select **Select Release** as the search **Release** setting.
4. Leave or select **Select Product Family** as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Component Report table shows the total number of components in the Oracle Flow Builder database and the number of components in each status:

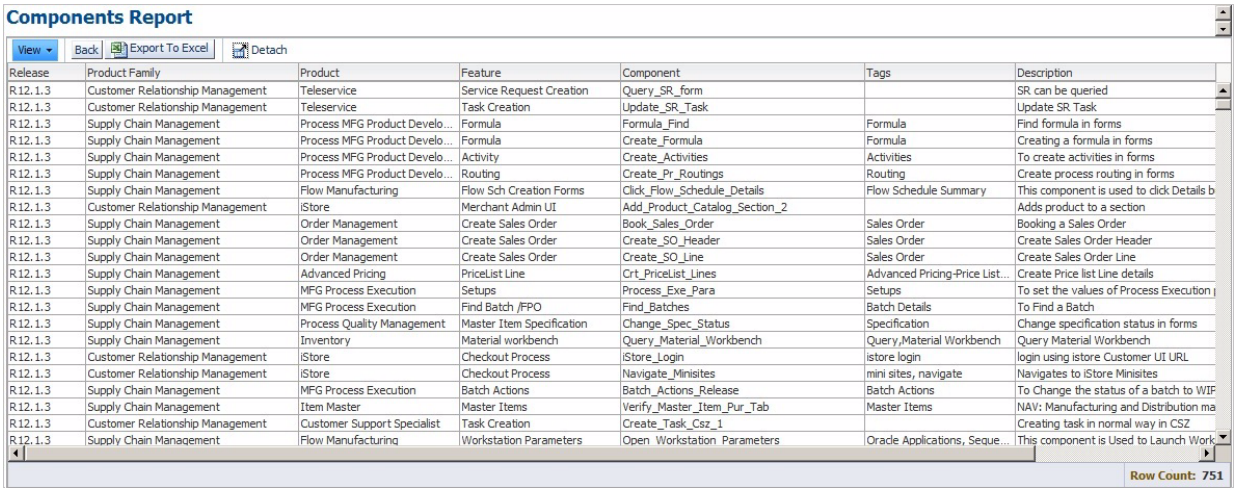
Figure 8–7 Component Totals Report in Table View



The Component Totals report is a table only report. There is no graphical view for this report.

8. Click a value in the report to view the component report for a status or the component totals.

Figure 8–8 Component Report Details in Table View



9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Back** to return to the search pane.

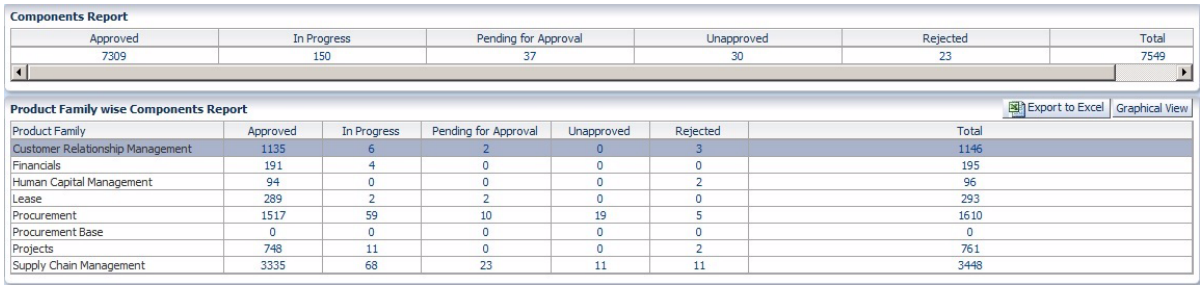
8.3.2 Generating a Component by Product Family Report

A components by product family report shows the number of components in each status for each of the Product Families in a Release.

To generate a component by product family report:

1. Log in and go to the Reports page.
2. Select **Component** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Leave or select **Select Product Family** as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Component Report tables show the total number of components in the selected Release and the number of components in each status for each Product Family in the Release:

Figure 8–9 Components by Product Family Report in Table View



Components Report

Approved	In Progress	Pending for Approval	Unapproved	Rejected	Total
7309	150	37	30	23	7549

Product Family wise Components Report — Export to Excel | Graphical View

Product Family	Approved	In Progress	Pending for Approval	Unapproved	Rejected	Total
Customer Relationship Management	1135	6	2	0	3	1146
Financials	191	4	0	0	0	195
Human Capital Management	94	0	0	0	2	96
Lease	289	2	2	0	0	293
Procurement	1517	59	10	19	5	1610
Procurement Base	0	0	0	0	0	0
Projects	748	11	0	0	2	761
Supply Chain Management	3335	68	23	11	11	3448

8. Click a value in the report to view the component report details for a product family by status or the component totals.
9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Graphical View** to view the data as graphs.

8.3.3 Generating a Component by Product Report

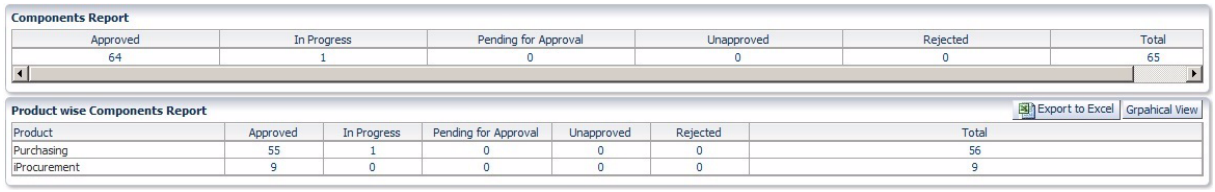
A components by product report shows the number of components in each status for each of the Products in a Product Family of a Release.

To generate a component by product report:

1. Log in and go to the Reports page.
2. Select **Component** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.

7. Click **Search**. The Component Report tables show the total number of components in the selected Product Family and the number of components in each status for each Product in the Product Family:

Figure 8–10 Components by Product Report in Table View



Components Report

Approved	In Progress	Pending for Approval	Unapproved	Rejected	Total
64	1	0	0	0	65

Product wise Components Report — Export to Excel | Grpahical View

Product	Approved	In Progress	Pending for Approval	Unapproved	Rejected	Total
Purchasing	55	1	0	0	0	56
iProcurement	9	0	0	0	0	9

8. Click a value in the report to view the component report details for a product by status or the component totals.
9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Graphical View** to view the data as graphs.

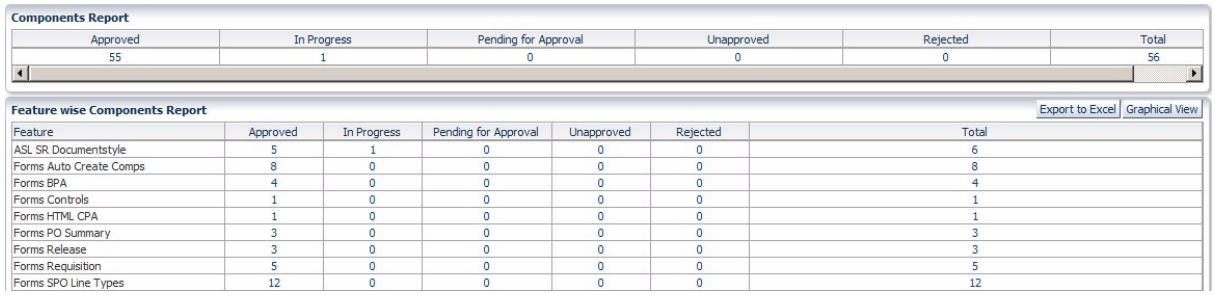
8.3.4 Generating a Component by Feature Report

A components by feature report shows the number of components in each status for each of the Features of a specific product.

To generate a component by feature report:

1. Log in and go to the Reports page.
2. Select **Component** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Select a product as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Component Report tables show the total number of components in the selected Product and the number of components in each status for each Feature in the Product:

Figure 8–11 Components by Feature Report in Table View



Components Report

Approved	In Progress	Pending for Approval	Unapproved	Rejected	Total
55	1	0	0	0	56

Feature wise Components Report — Export to Excel | Graphical View

Feature	Approved	In Progress	Pending for Approval	Unapproved	Rejected	Total
ASL SR Documentstyle	5	1	0	0	0	6
Forms Auto Create Comps	8	0	0	0	0	8
Forms BPA	4	0	0	0	0	4
Forms Controls	1	0	0	0	0	1
Forms HTML CPA	1	0	0	0	0	1
Forms PO Summary	3	0	0	0	0	3
Forms Release	3	0	0	0	0	3
Forms Requisition	5	0	0	0	0	5
Forms SPO Line Types	12	0	0	0	0	12

8. Click a value in the report to view the component report details for a feature by status or the component totals.
9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Graphical View** to view the data as graphs.

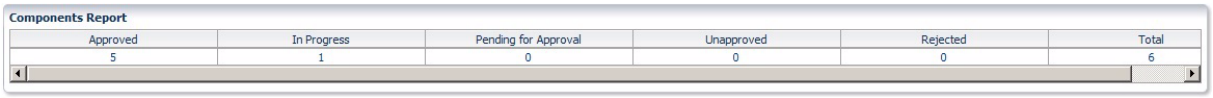
8.3.5 Generating a Component Report for a Specific Feature

A components for a specific feature report shows the number of components in each status for the selected Feature of a specific product.

To generate a component report for a specific feature:

1. Log in and go to the Reports page.
2. Select **Component** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Select a product as the search **Product** setting.
6. Select a feature as the search **Feature** setting.
7. Click **Search**. The Component Report table shows the total number of components in the selected Feature and the number of components in each status for the selected Feature:

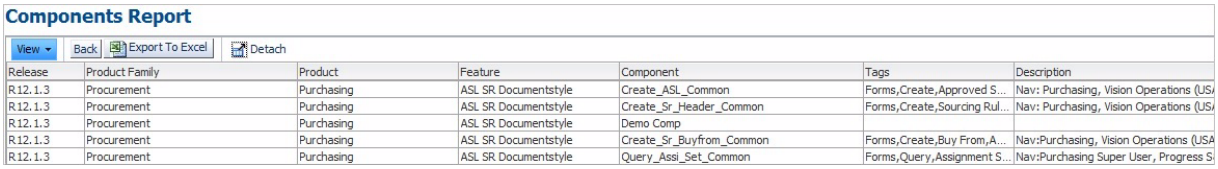
Figure 8–12 Component Report for a Specific Feature in Table View



The Component Report for a specific feature is a table only report. There is no graphical view for this report.

8. Click a value in the report to view the component report details for a feature by status or the component totals.

Figure 8–13 Component Report Details for a Specific Feature in Table View



9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Back** to return to the search view.

8.4 Generating Component Set Reports

Component Set reports show the total number of component sets in the selected Release, Product Family, Product, and Feature.

The following section explain how to generate specific types of component reports:

8.4.1 Generating a Component Set Totals Report

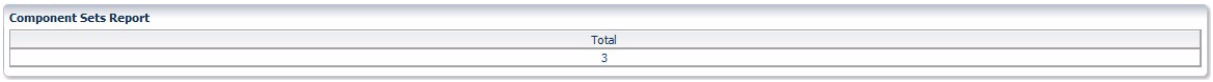
A component totals report shows the total number of components in the Oracle Flow Builder database.

To generate a component set report:

1. Log in and go to the Reports page.

2. Select **Component Set** as the search **Type** setting.
3. Leave or select **Select Release** as the search **Release** setting.
4. Leave or select **Select Product Family** as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Component Set Report table shows the total number of components sets in the Oracle Flow Builder database:

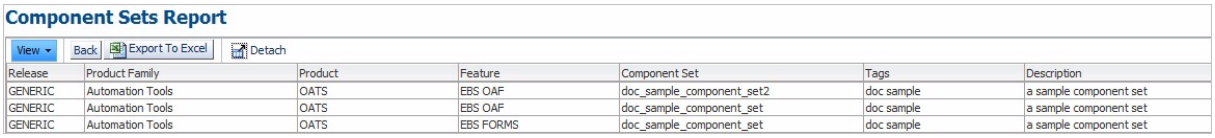
Figure 8–14 Component Set Totals Report in Table View



The Component Set Totals report is a table only report. There is no graphical view for this report.

8. Click the value in the report to view the component set report.

Figure 8–15 Component Set Report Details in Table View



9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Back** to return to the search pane.

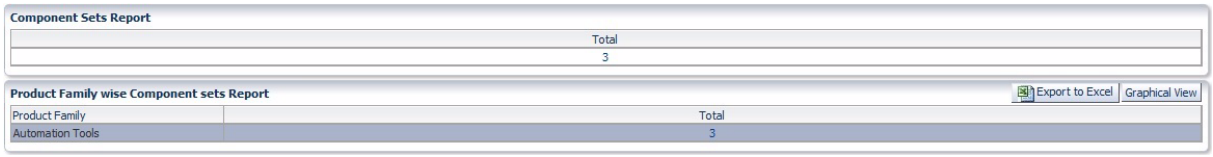
8.4.2 Generating a Component Set by Product Family Report

A component sets by product family report shows the number of component sets in each of the Product Families in a Release.

To generate a component set by product family report:

1. Log in and go to the Reports page.
2. Select **Component Set** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Leave or select **Select Product Family** as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Component Set Report tables show the total number of component sets in the selected Release for each Product Family in the Release:

Figure 8–16 Component Sets by Product Family Report in Table View



8. Click a value in the report to view the component set report details for a product family.
9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Graphical View** to view the data as graphs.

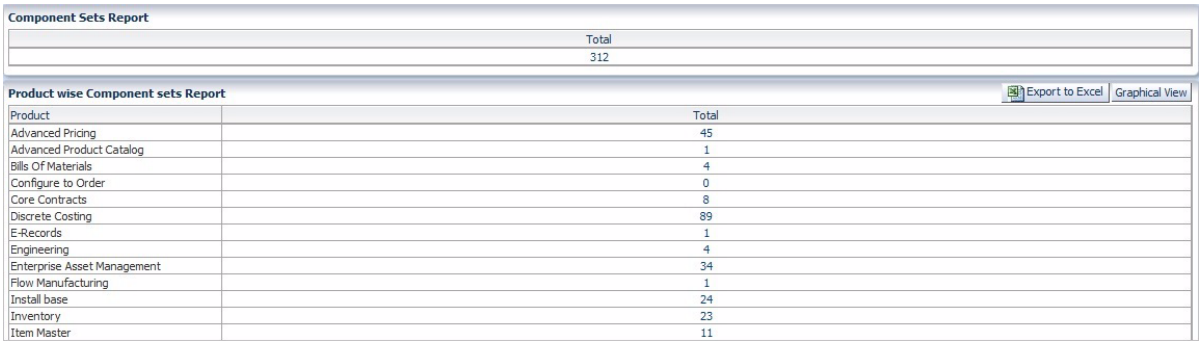
8.4.3 Generating a Component Set by Product Report

A component sets by product report shows the number of component sets for each of the Products in a Product Family of a Release.

To generate a component sets by product report:

1. Log in and go to the Reports page.
2. Select **Component** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Component Set Report tables show the total number of component sets in the selected Product Family and the number of component sets in each Product in the Product Family:

Figure 8–17 Component Sets by Product Report in Table View



8. Click a value in the report to view the component set report details for a product.
9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Graphical View** to view the data as graphs.

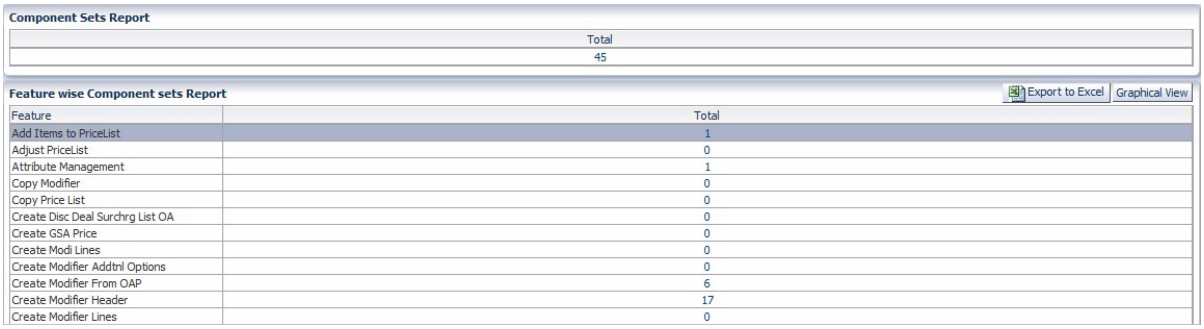
8.4.4 Generating a Component Set by Feature Report

A component sets by feature report shows the number of component sets in each of the Features of a specific product.

To generate a component set by feature report:

1. Log in and go to the Reports page.
2. Select **Component** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Select a product as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Component Set Report tables show the total number of component sets in the selected Product and the number of component sets in each Feature in the Product:

Figure 8–18 Component Sets by Feature Report in Table View



Component Sets Report

Total
45

Feature wise Component sets Report — Export to Excel | Graphical View

Feature	Total
Add Items to PriceList	1
Adjust PriceList	0
Attribute Management	1
Copy Modifier	0
Copy Price List	0
Create Disc Deal Surchrg List OA	0
Create GSA Price	0
Create Modi Lines	0
Create Modifier Addtnl Options	0
Create Modifier From OAP	6
Create Modifier Header	17
Create Modifier Lines	0

8. Click a value in the report to view the component set report details for a feature.
9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Graphical View** to view the data as graphs.

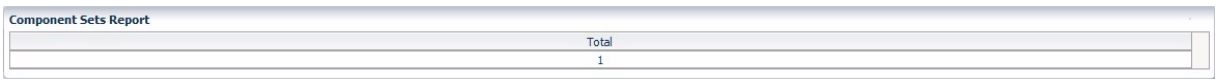
8.4.5 Generating a Component Set Report for a Specific Feature

A component sets for a specific feature report shows the number of component sets for the selected Feature of a specific product.

To generate a component set report for a specific feature:

1. Log in and go to the Reports page.
2. Select **Component** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Select a product as the search **Product** setting.
6. Select a feature as the search **Feature** setting.
7. Click **Search**. The Component Set Report table shows the total number of component sets in the selected Feature:

Figure 8–19 Component Set Report for a Specific Feature in Table View



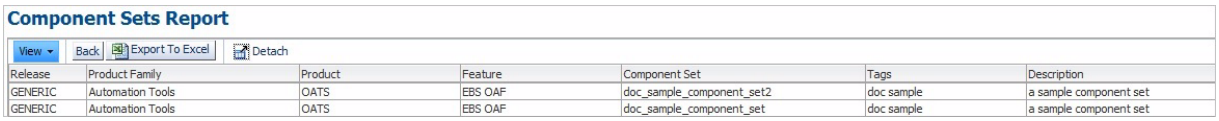
Component Sets Report

Total
1

The Component Set Report for a specific feature is a table only report. There is no graphical view for this report.

8. Click a value in the report to view the component set report details for a feature.

Figure 8–20 Component Set Report Details for a Specific Feature in Table View



Component Sets Report

View | Back | Export To Excel | Detach

Release	Product Family	Product	Feature	Component Set	Tags	Description
GENERIC	Automation Tools	OATS	EBS OAF	doc_sample_component_set2	doc sample	a sample component set
GENERIC	Automation Tools	OATS	EBS OAF	doc_sample_component_set	doc sample	a sample component set

9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Back** to return to the search view.

8.5 Generating Flow Reports

Flow reports show the number and status of Certification, Very High, High, Low, and Test flow types and the total number of flows in the selected Release, Product Family, Product, and Feature.

The following sections explain how to generate specific types of flow reports:

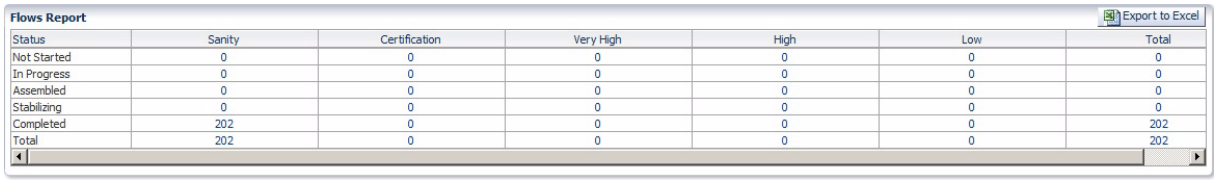
8.5.1 Generating a Flow Totals Report

A flow totals report shows the status and number of flow in the Oracle Flow Builder database.

To generate a component totals report:

1. Log in and go to the Reports page.
2. Select **Flow** as the search **Type** setting.
3. Leave or select **Select Release** as the search **Release** setting.
4. Leave or select **Select Product Family** as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Flow Report table shows the total number of flows in the Oracle Flow Builder database and the number of flows in each status:

Figure 8–21 Flows Totals Report in Table View



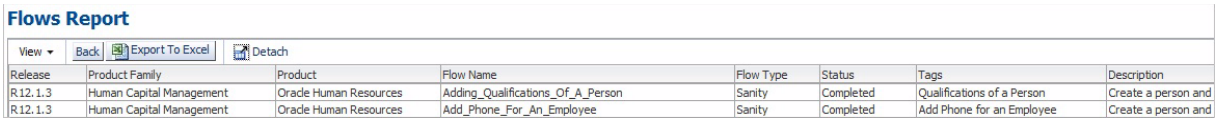
Flows Report — Export to Excel

Status	Sanity	Certification	Very High	High	Low	Total
Not Started	0	0	0	0	0	0
In Progress	0	0	0	0	0	0
Assembled	0	0	0	0	0	0
Stabilizing	0	0	0	0	0	0
Completed	202	0	0	0	0	202
Total	202	0	0	0	0	202

The Flows Totals report is a table only report. There is no graphical view for this report.

8. Click a value in the report to view the flow report for a status or the flow totals.

Figure 8–22 Flows Report Details in Table View



Flows Report

View | Back | Export To Excel | Detach

Release	Product Family	Product	Flow Name	Flow Type	Status	Tags	Description
R12.1.3	Human Capital Management	Oracle Human Resources	Adding_Qualifications_Of_A_Person	Sanity	Completed	Qualifications of a Person	Create a person and
R12.1.3	Human Capital Management	Oracle Human Resources	Add_Phone_For_An_Employee	Sanity	Completed	Add Phone for an Employee	Create a person and

9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Back** to return to the search pane.

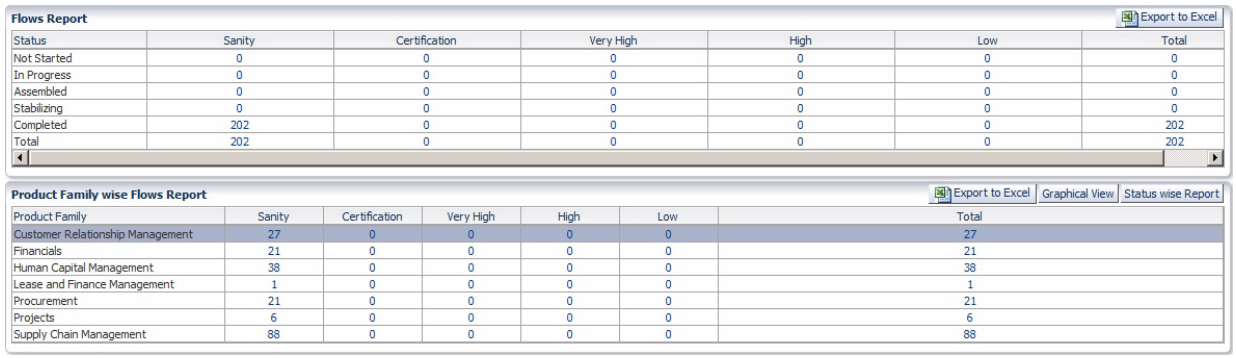
8.5.2 Generating a Flows by Product Family Report

A flows by product family report shows the number of flows in each status for each of the Product Families in a Release.

To generate a flows by product family report:

1. Log in and go to the Reports page.
2. Select **Flow** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Leave or select **Select Product Family** as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Flows Report tables show the total number of components in the selected Release and the number of flows or each type for each Product Family in the Release:

Figure 8–23 Flows by Product Family Report in Table View



Flows Report — Export to Excel

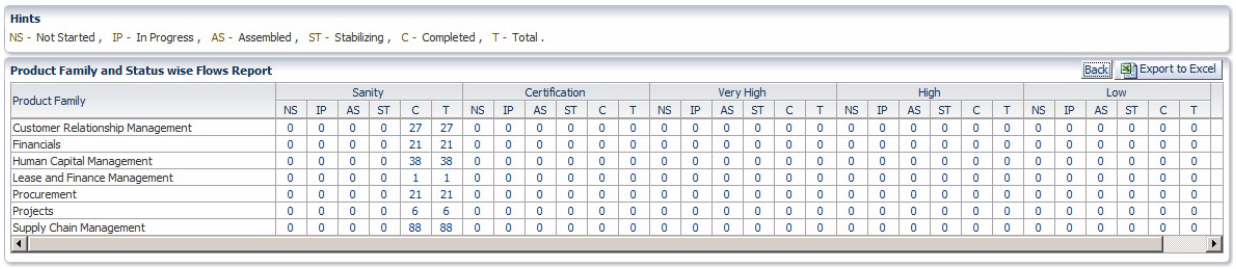
Status	Sanity	Certification	Very High	High	Low	Total
Not Started	0	0	0	0	0	0
In Progress	0	0	0	0	0	0
Assembled	0	0	0	0	0	0
Stabilizing	0	0	0	0	0	0
Completed	202	0	0	0	0	202
Total	202	0	0	0	0	202

Product Family wise Flows Report — Export to Excel | Graphical View | Status wise Report

Product Family	Sanity	Certification	Very High	High	Low	Total
Customer Relationship Management	27	0	0	0	0	27
Financials	21	0	0	0	0	21
Human Capital Management	38	0	0	0	0	38
Lease and Finance Management	1	0	0	0	0	1
Procurement	21	0	0	0	0	21
Projects	6	0	0	0	0	6
Supply Chain Management	88	0	0	0	0	88

8. Click a value in the report to view the flow report details for a product family by status or the component totals.
9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Graphical View** to view the data as graphs.
11. Click **Status wise Report** to view the status data for each type of flow in each product family.

Figure 8–24 Status of Flows by Product Family Report in Table View



Hints

NS - Not Started , IP - In Progress , AS - Assembled , ST - Stabilizing , C - Completed , T - Total .

Product Family and Status wise Flows Report

Back | Export to Excel

Product Family	Sanity						Certification						Very High						High						Low					
	NS	IP	AS	ST	C	T	NS	IP	AS	ST	C	T	NS	IP	AS	ST	C	T	NS	IP	AS	ST	C	T	NS	IP	AS	ST	C	T
Customer Relationship Management	0	0	0	0	27	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Financials	0	0	0	0	21	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Human Capital Management	0	0	0	0	38	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lease and Finance Management	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Procurement	0	0	0	0	21	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Projects	0	0	0	0	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Supply Chain Management	0	0	0	0	88	88	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The Product Family and Status Report uses the following abbreviations to represent the status of flows of each type in each product family:

- **NS** - Not Started
- **IP** - In Progress
- **AS** - Assembled
- **ST** - Stabilizing
- **C** - Completed
- **T** - Total

12. Click a value in the report to view the flow report details for a product family by status or the flow totals.
13. Click **Export to Excel** to save the data to an Excel spreadsheet file.
14. Click **Back** to return to the previous report view or the search view.

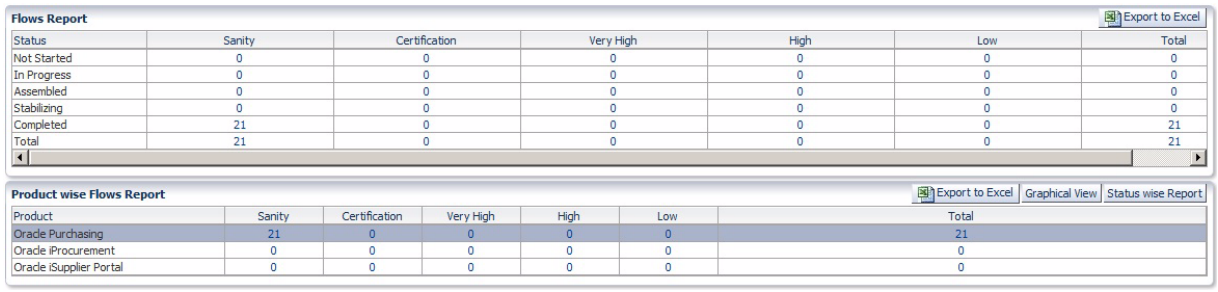
8.5.3 Generating a Flows by Product Report

A flows by product report shows the number of flows in each status for each of the Products in a Product Family of a Release.

To generate a flows by product report:

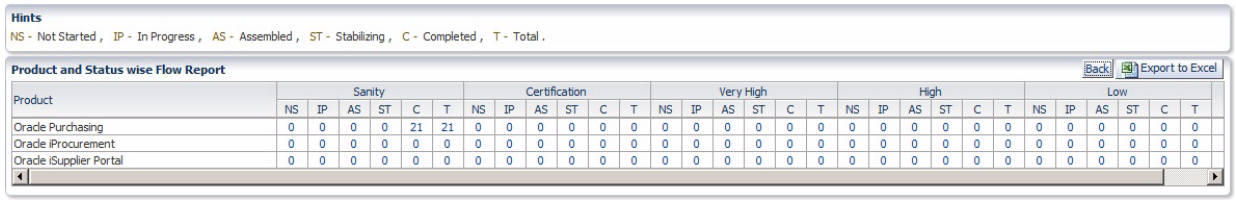
1. Log in and go to the Reports page.
2. Select **Flow** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Flows Report tables show the total number of flows in the selected Product Family and the number of flows for each Product in the Product Family:

Figure 8–25 Flows by Product Report in Table View



8. Click a value in the report to view the flow report details for a product by status or the component totals.
9. Click **Export to Excel** to save the data to an Excel spreadsheet file.
10. Click **Graphical View** to view the data as graphs.
11. Click **Status wise Report** to view the status data for each type of flow in each product.

Figure 8–26 Status of Flows by Product Report in Table View



The Product and Status Report uses the following abbreviations to represent the status of flows of each type in each product:

- **NS** - Not Started
- **IP** - In Progress
- **AS** - Assembled
- **ST** - Stabilizing
- **C** - Completed
- **T** - Total

12. Click a value in the report to view the flow report details for a product by status or the flow totals.
13. Click **Export to Excel** to save the data to an Excel spreadsheet file.
14. Click **Back** to return to the previous report view or the search view.

8.5.4 Generating a Flows by Feature Report

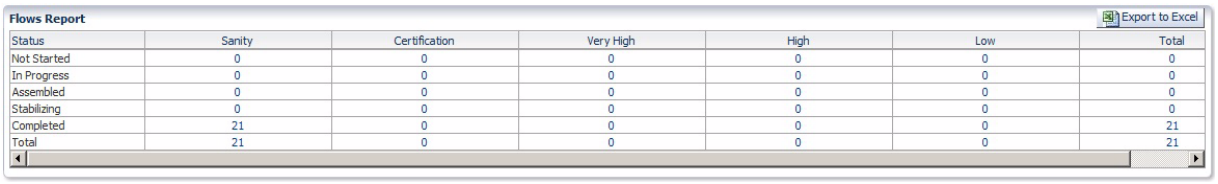
A flow by feature report shows the number of flows in each status for the flows in a specific product.

To generate a flows by feature report:

1. Log in and go to the Reports page.
2. Select **Flows** as the search **Type** setting.

3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Select a product as the search **Product** setting.
6. Leave or select **Select Feature** as the search **Feature** setting.
7. Click **Search**. The Flows Report tables shows the total number of flows in the selected Product and the number of flows in each status in the Product:

Figure 8–27 Flows Report in Table View



Flows Report

Export to Excel

Status	Sanity	Certification	Very High	High	Low	Total
Not Started	0	0	0	0	0	0
In Progress	0	0	0	0	0	0
Assembled	0	0	0	0	0	0
Stabilizing	0	0	0	0	0	0
Completed	21	0	0	0	0	21
Total	21	0	0	0	0	21

8. Click a value in the report to view the flows report details.
9. Click **Export to Excel** to save the data to an Excel spreadsheet file.

9

Administering Oracle Flow Builder

This chapter explains how to perform administrative tasks within the Oracle Flow Builder application. This chapter contains the following sections:

9.1 Overview

This section provides an overview of the administrative tasks with in the Oracle Flow Builder application. Administrator tasks can be performed by any user with administrator user privileges. However, the administrator user defined during setup of the application performs the initial administrative tasks. The administrative tasks include the following tasks:

- Setup - define the Release, Product Family, Product, and Features hierarchy of the Component Tree. Define user roles and users. Users can also request registration from the Oracle Flow Builder login pane.
- Product Family Access Controls - manage user access to specific product families.

Follow these steps to access the administrative options within the Oracle Flow Builder application:

1. Log in to Oracle Flow Builder using Administrator credentials.
2. Click **Administration** at the top of the Home page.

Figure 9–1 Administration Options

The links on the left side of the Administration page provide access to the specific administrative tasks. The following sections explain the specific administrative tasks.

Table 9–1 Administrator Tasks

Step	Tasks	Role
1	Setting Up Oracle Flow Builder See Section 9.2, "Setting Up Oracle Flow Builder"	Administrator
2	Managing Product Family Access See Section 9.3, "Managing Product Family Access"	Administrator

9.2 Setting Up Oracle Flow Builder

An Administrator defines and manages the component tree hierarchy and users and user roles set up. The following tasks are performed by an administrator:

- Setting Up Releases
- Setting Up Product Families
- Setting Up Products
- Setting Up Features
- Setting Up Roles
- Setting Up Users
- Setting Up Function Libraries
- Setting Up Email

9.2.1 Setting Up Releases

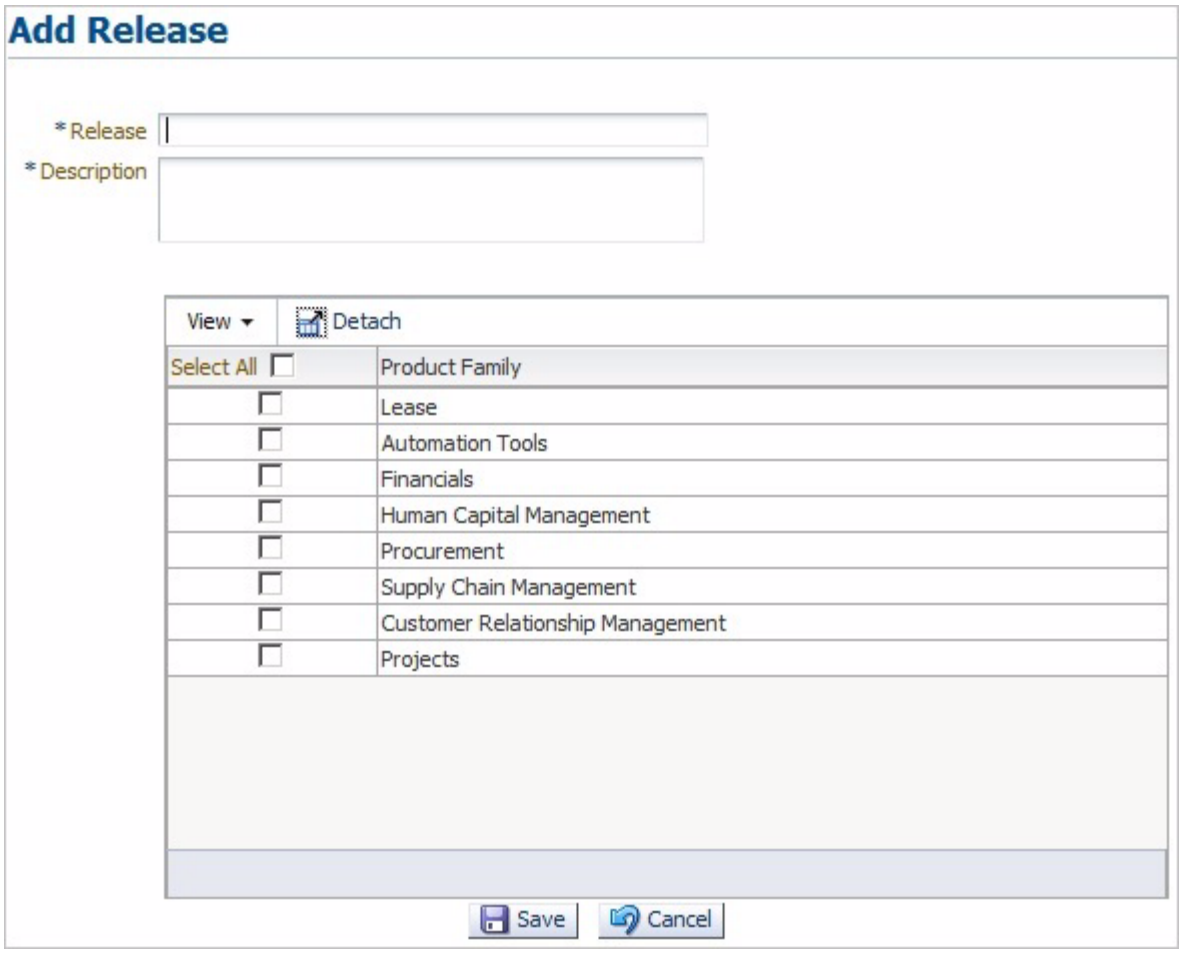
This section explains the procedures for administering the Releases within Oracle Flow Builder. Releases are the top level of the Component Tree hierarchy within the Oracle Flow Builder application.

9.2.1.1 Adding Releases

Follow these steps to add a new Release to the Component Tree:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Releases**.
3. Click **Add** to define the name and description of the Release to make available in the Component Tree.

Figure 9–2 Add Release Options



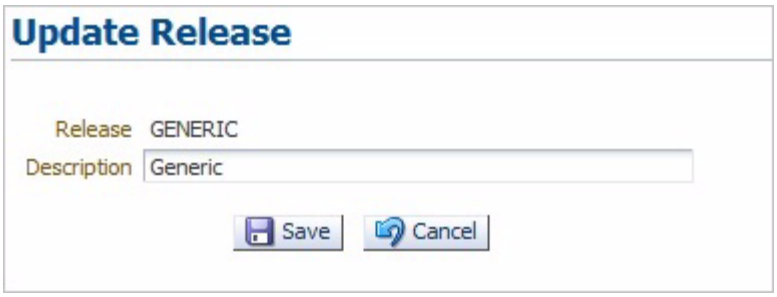
4. Define the Release name and description.
5. Select the product family or families to include under the Release in the Component Tree hierarchy and click **Save**.

9.2.1.2 Updating Releases

Follow these steps to update a Release in the Component Tree:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Releases**.
3. Enter the name of the Release (or use % wildcard) and click **Search** to list currently defined Releases.
4. Click the **Update** icon to view the Release information.

Figure 9–3 Update Release Options



5. Edit the Release information and click **Save**.
6. Select the product family or families to include in the Component Tree hierarchy and click **Save**.

9.2.2 Setting Up Product Families

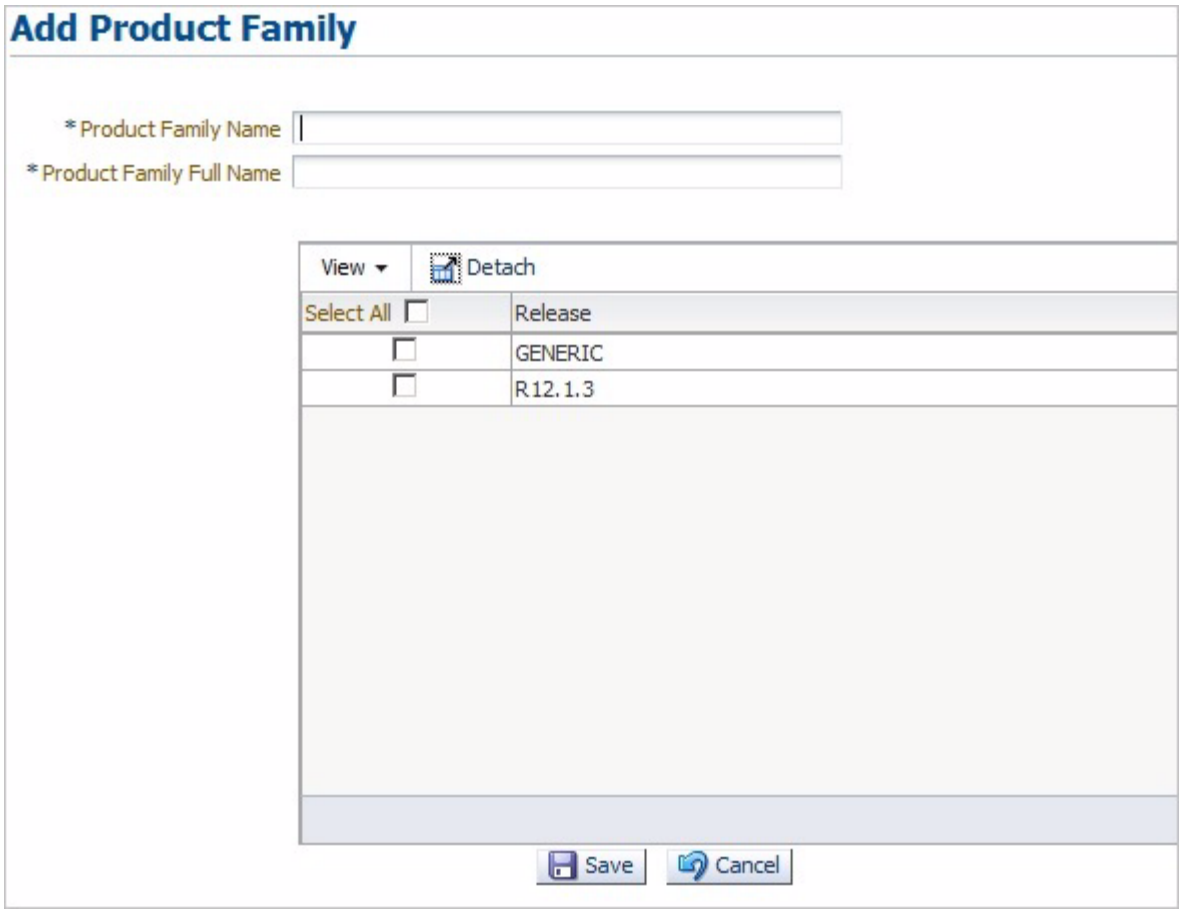
This section explains the procedures for administering the Product Families within Oracle Flow Builder. Product Families are defined to categorize the products and features in a Release hierarchy within the Component Tree.

9.2.2.1 Adding Product Families

Follow these steps to add a new Product Family to a Release in the Component Tree:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Product Families**.
3. Click **Add** to define the Product Family to make available in the Component Tree.

Figure 9–4 Add Product Family Options



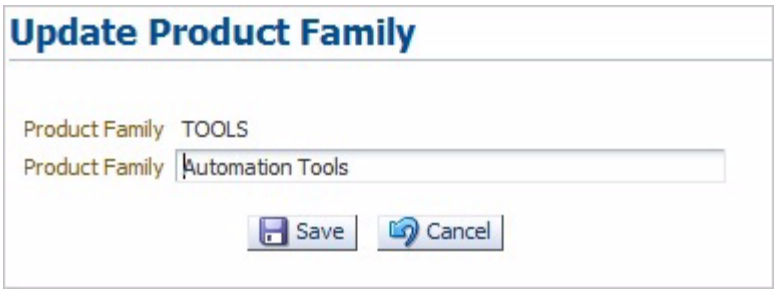
4. Define the Product Family Name and Product Family Full Name.
5. Select the Release(s) and click **Save**.

9.2.2.2 Updating Product Families

Follow these steps to update a Product Family in the Component Tree:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Releases**.
3. Enter the name of the Product Family (or use % wildcard) and click **Search** to list currently defined Product Families.
4. Click the **Update** icon to view the Product Family information.

Figure 9–5 Update Product Family Options



5. Edit the Product Family information and click **Save**.

9.2.3 Setting Up Products

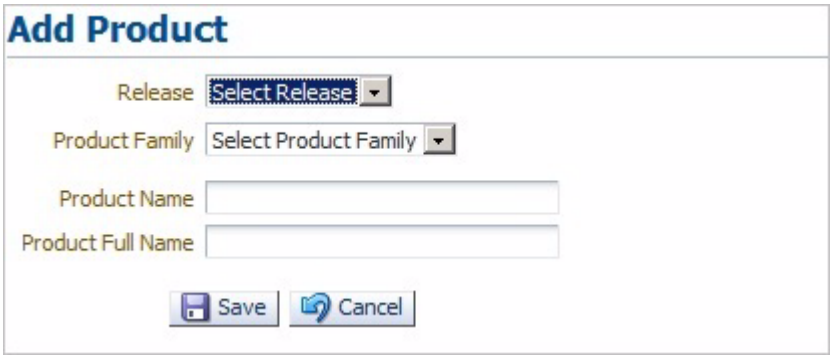
This section explains the procedures for administering the Products within Oracle Flow Builder. Products are defined to categorize the features in a Product Family hierarchy within the Component Tree.

9.2.3.1 Adding Products

Follow these steps to add a new Product to a Product Family in the Component Tree:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Products**.
3. Click **Add** to define the Product to make available in the Component Tree.

Figure 9–6 Add Product Options



4. Select the Release.
5. Select the Product Family.
6. Define the Product Name and Product Full Name and click **Save**.

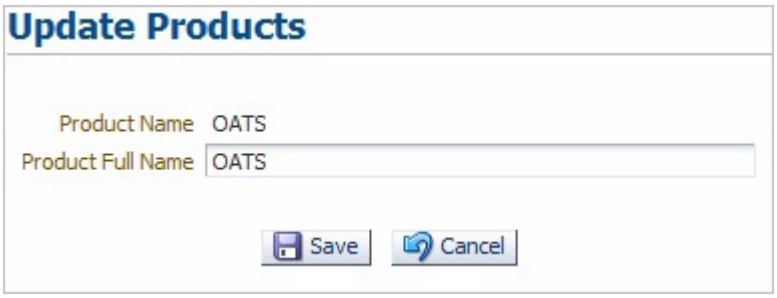
9.2.3.2 Updating Products

Follow these steps to update a Product in the Component Tree:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Products**.
3. Select the Release.
4. Select the Product Family.

5. Enter the name of the Product (or use % wildcard) and click **Search** to list currently defined Products.
6. Click the **Update** icon to view the Product information.

Figure 9–7 Update Product Options



7. Edit the Product information and click **Save**.

9.2.4 Setting Up Features

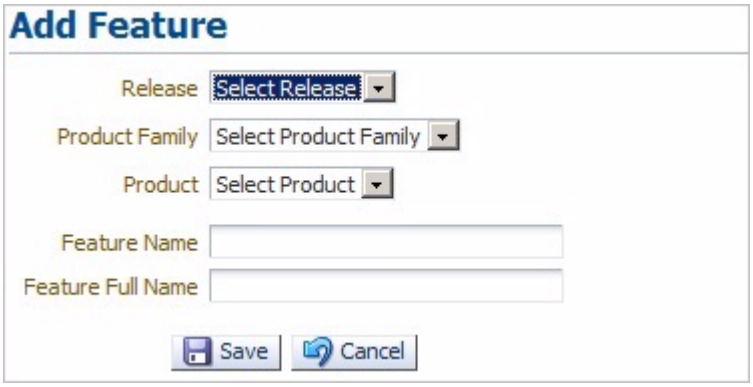
This section explains the procedures for administering the Features within Oracle Flow Builder. Features define specific features of a Product to test within the Component Tree.

9.2.4.1 Adding Features

Follow these steps to add a new Feature to a Product in the Component Tree:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Features**.
3. Click **Add** to define the Feature to make available in the Component Tree.

Figure 9–8 Add Feature Options



4. Select the Release.
5. Select the Product Family.
6. Select the Product.
7. Define the Feature Name and Feature Full Name and click **Save**.

9.2.4.2 Updating Features

Follow these steps to update a Feature in the Component Tree:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Features**.
3. Select the Release.
4. Select the Product Family.
5. Select the Product.
6. Enter the name of the Feature (or use % wildcard) and click **Search** to list currently defined Products.
7. Click the **Update** icon to view the Feature information.

Figure 9–9 Update Feature Options

8. Edit the Feature information and click **Save**.

9.2.5 Setting Up Roles

This section explains the procedures for administering the user roles within Oracle Flow Builder. User Roles specify the categories of users and the specific permissions assigned to each user role.

9.2.5.1 Adding Roles

Follow these steps to add a new user role to the Oracle Flow Builder application:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Roles**.
3. Click **Add** to define the user role name and role actions assigned to that role.

Figure 9–10 Add Role Options

4. Define the Role name.
5. Select the Role Actions to assign to the user role.

Table 9–2 User Role Actions

Role Action	Definition
Upload	Permission to upload files.
Unlock	Permission to unlock components.
Change PF Access	Permission to change access to Product Families.
Create	Permission to create components and flows.
Delete	Permission to delete components and flows.
Approve	Permission to approve component changes.
UCC	Permission to Update Component Code.
UCH	Permission to Update Component Headers.
UFS	Permission to Update Flow Structures.
UFH	Permission to Update Flow Headers.
UCSS	Permission to Update Component Set Structures.
UCSH	Permission to Update Component Set Headers.
SF	Permission to Stabilizing Flows.

Table 9–2 (Cont.) User Role Actions

Role Action	Definition
CF	Permission to Complete Flows.

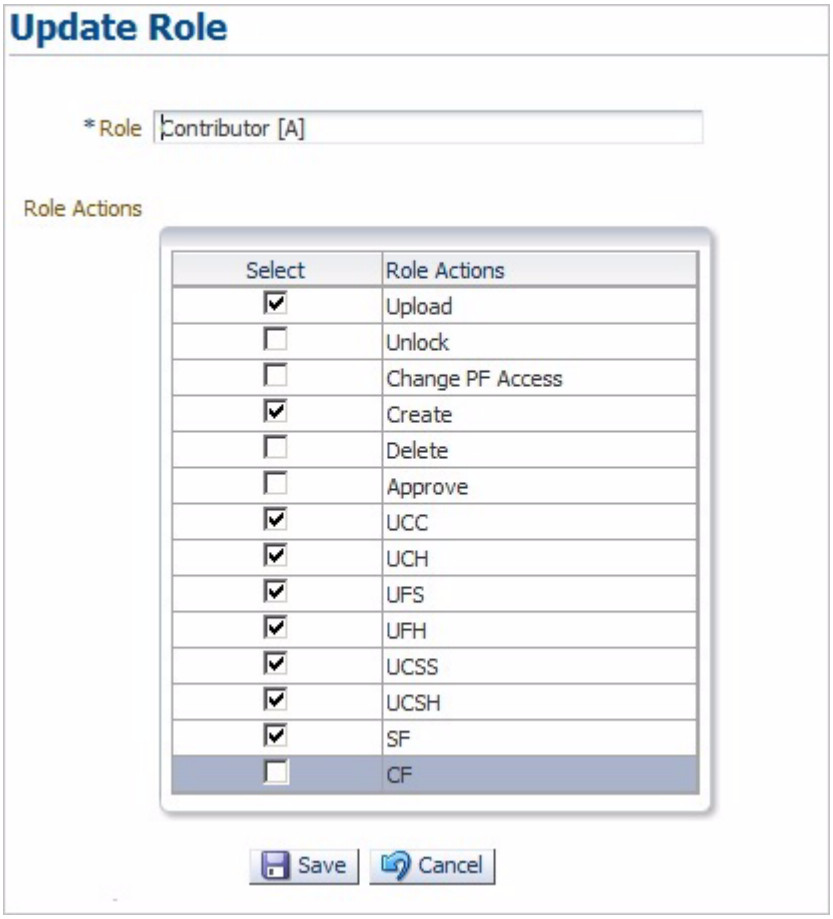
6. Click **Save** to add the user role.

9.2.5.2 Updating Roles

Follow these steps to update user role in the Oracle Flow Builder application:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Roles**.
3. Enter the name of the Role (or use % wildcard) and click **Search** to list currently defined Roles.
4. Click the **Update** icon to view the Role information.

Figure 9–11 Update Role Options



5. Edit the Role information and click **Save**.

9.2.6 Setting Up Users

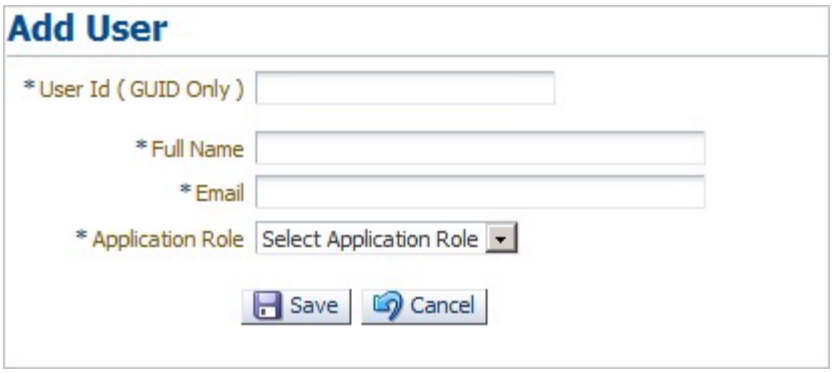
This section explains the procedures for administering the users within Oracle Flow Builder.

9.2.6.1 Adding Users

Follow these steps to add a new user to the Oracle Flow Builder application:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Users**.
3. Click **Add** to define the user information.

Figure 9–12 Add User Options



4. Define the User ID, full name, and email.

> **Note:** Make sure the email server has been set up before adding users. An email notification will be sent to newly added users specifying the initial password to use to log in to the application. See Section 9.2.8, "Setting Up Email" for additional information.

5. Select the Application Role to assign to the user.

Table 9–3 Application Roles

Role Action	Definition
Admin	The user has an Administrator role.
Approver	The user has an Approver role.
Contributor	The user has a Contributor role.
Member	The user has a Member role.

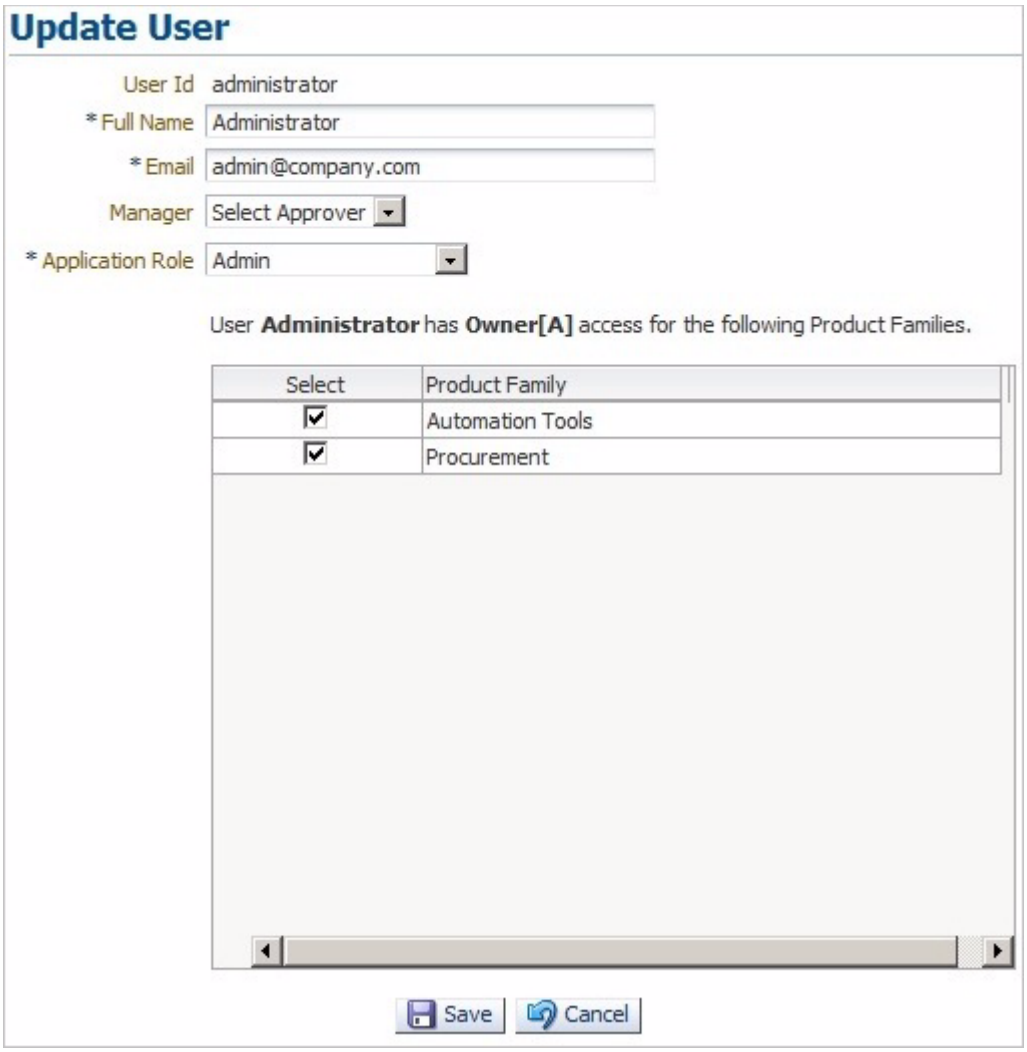
6. Click **Save** to add the user.

9.2.6.2 Updating Users

Follow these steps to update users in the Oracle Flow Builder application:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Users**.
3. Enter the ID or name of the User (or use % wildcard) and click **Search** to list currently defined Users.
4. Click the **Update** icon to view the User information.

Figure 9–13 Update User Options



5. Edit the User information and click **Save**.

9.2.7 Setting Up Function Libraries

This section explains the procedures for administering function libraries within Oracle Flow Builder. Function libraries define the built in and custom functions used with the FUNCTIONCALL keyword to perform specific tasks within component code.

The Oracle Flow Builder application includes a set of function libraries that can be used for specific testing purposes. The following table lists the default function libraries included with the application:

Table 9–4 Function Libraries Installed with Oracle Flow Builder

Library	Related Application/Description
cRMLIB	Default function library for Customer Relationship Management application components.
eBSLibrary	Default function library for E-Business Suite application components.
gENLIB	Default function library for generic application components.

Table 9–4 (Cont.) Function Libraries Installed with Oracle Flow Builder

Library	Related Application/Description
pRJTBLVERIFYLIB	Default function library for verification of Projects application components.
pROCLIB	Default function library for Procurement application components.
pROJLIB	Default function library for Projects application components.
sCMLIB	Default function library for Supply Chain Management application components.
tELNETLIB	Default function library for Telnet application components (requires a third-party Java library to be added to the OpenScript repository where Oracle Flow Builder-generated scripts will be executed. See Adding a Telnet Function Library for additional information).
wEBTABLELIB	Default function library for Web table application components.

See Appendix B, "Function Library Reference" for details about the functions available in each function library.

9.2.7.1 Setting Up a Function Library Repository in OpenScript

The Oracle Flow Builder function libraries must be added to the OpenScript installation that users will use to playback functional testing scripts generated by Oracle Flow Builder. The function libraries are added to the repository where the generated Oracle Flow Builder script files will unzipped and executed. This is a one-time setup procedure required for each and any OpenScript installation that will be used to execute Oracle Flow Builder generated functional test scripts in OpenScript.

> **Note:** Function libraries shipped with Oracle Flow Builder are included in the product download zip file in the following location:
>
> `<OFB-install-files>/common/install/static/libs/function-libs.zip`

To set up a function library repository in OpenScript:

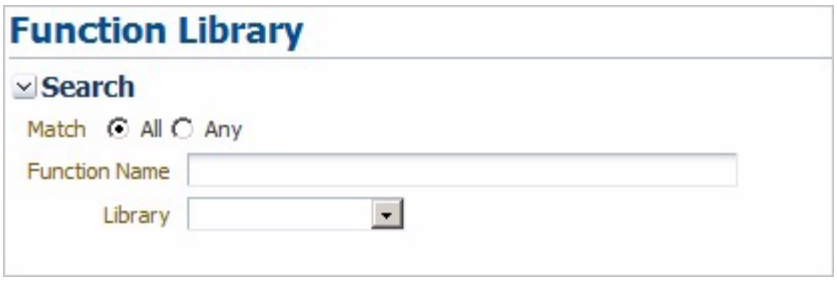
1. Start OpenScript.
2. From the **Tools** menu, select **Manage Repositories**.
3. Click **Add**.
4. Enter `OATS` as the **Name**.
5. Enter *`<any-location-on-disk>`* as the **Location**.
6. Click **OK**.
7. Copy the function libraries (`function-libs.zip`) included with the Oracle Flow Builder download zip to the OATS Repository folder and unzip the file.

9.2.7.2 Searching Function Libraries

Follow these steps to search function libraries used with the Oracle Flow Builder application:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Function Library**.

Figure 9–14 Function Library Search Options



3. Select the Match type: **All** or **Any**.
4. Select a function library or leave blank to search all libraries.
5. Enter all or part of a function name (or use % wildcard) and click **Search** to list functions within the library.

9.2.7.3 Creating a Function Library Script

Creating a function library requires that you have the OpenScript component of the Oracle Application Testing Suite installed on a Windows machine. This section provides the basic steps for creating a custom dedicated function library script for use in the Oracle Flow Builder application. See the *Oracle Functional Testing OpenScript User's Guide* for additional information about creating function library scripts and adding functions.

To create a dedicated function library script:

1. Start OpenScript.
2. Select **New** from the **File** menu.
3. Select the project type and click **Next**.
4. Enter a script name for the function library (for example, myScriptLib).
5. Select **Create script as a Function Library**.
6. Click **Next**. The script wizard opens the Create Function Library options:

   Package: Specifies a unique Package name for the function library. Package must be a valid Java Package name that matches A-Z, a-z, 0-9, _. It must not contain spaces or Double-Byte Character Sets (DBCS). The initial default value is `myCompany.myTeam`. Subsequently, the default value will be set to the last value specified.

   Class: Specifies a unique alias to use as the name (typically the Class name) for the function library script. **Class** must be a valid Java Class name that matches A-Z, a-z, 0-9, _. It must not contain spaces or Double-Byte Character Sets (DBCS). The Class Name should be:

   - meaningful and provide context as to the purpose of the library,
   - clear and concise so it is easy to read and type in scripts,
   - something unique so it is not confused with other function libraries.

7. Enter a unique **Package** name for the function library in the form:

   ```
   orgName.groupName.subgroupName
   ```

   For example:

```
oracle.oats.dev
```

8. Enter a unique alias to use as the **Class** name for the function library to identify the library. For example:

   ```
   WebFunctLib
   ```

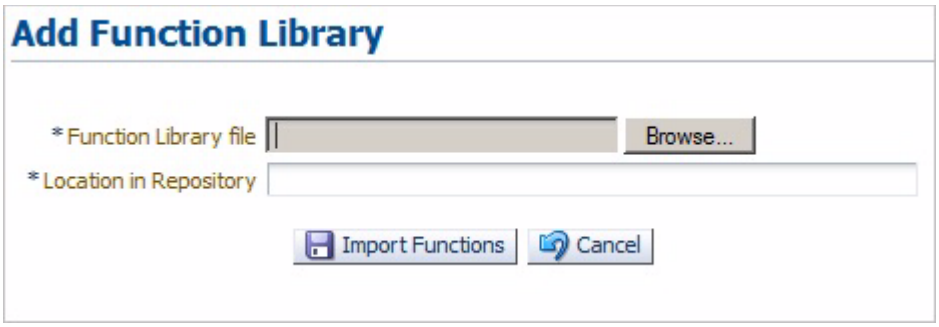
9. Click **Finish**.
10. Add your custom code to the function library script.
    - Use the script recorder to record steps.
    - Switch to the Java Code view and edit the code in the functions.
    - Functions declared public will be imported into Oracle Flow Builder.
    - Supporting functions should be declared private otherwise they will be imported into Oracle Flow Builder if declared public.
11. Save the function library script.
12. Select **Export** from the **File** menu.
13. Specify a file name for the zip file.
14. Clear the **Create self-contained zip file** option.
15. Under **Additional Files to export**, clear the **Recorded Data**, **Playback Results**, and **Error Log** options.
16. Click **OK** to save the file. This is the file you use to add the function library to the Oracle Flow Builder application. See Section 9.2.7.4, "Adding Function Libraries" for additional information about adding function libraries.

9.2.7.4 Adding Function Libraries

Follow these steps to add a new function library to the Oracle Flow Builder application:

1. Make sure you (or a function library developer) have created the function library script and exported the zip file from your OpenScript installation. See Section 9.2.7.3, "Creating a Function Library Script" for additional information.
2. Make sure you (or an Oracle Flow Builder administrator) have set up the function library repository in the OpenScript installation that will be used to execute the Oracle Flow Builder generated scripts. See Section 9.2.7.1, "Setting Up a Function Library Repository in OpenScript" for additional information.
3. Log in using Administrator credentials and go to the Administration page.
4. From **Setup**, click **Function Library**.
5. Click **Add New Library**.

Figure 9–15 Add Function Library Options



6. Click **Browse** and select the function library file.
7. Enter the target location of the function library relative to the main Oracle Application Testing Suite repository. This should be the same location specified in Section 9.2.7.1, "Setting Up a Function Library Repository in OpenScript".
8. Click **Import Functions**.

   The Parameter data is automatically created in Oracle Flow Builder when a new function library is imported. However, if you wish to include Comments and Test Plan description information for the functions in the library after the import has completed you must enter the data manually. See Section 9.2.7.5, "Modifying Functions" for information about modifying functions to add Comments and Test Plan description information.
9. Add the new function library to the `ebs-function-libs` folder in the `OATS` repository specified in the OpenScript installation that will be used to play back the functional test scripts that use the new library. See Section 9.2.7.1, "Setting Up a Function Library Repository in OpenScript" for additional information.
10. Modify the function library in the Orale Flow Builder application to add Comments and Test Plan description information for each new function added from the new function library. See Section 9.2.7.5, "Modifying Functions" for additional information.

9.2.7.5 Modifying Functions

To modify functions in a function library:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Function Library**.
3. Select the Match type: **All** or **Any**.
4. Select a function library or leave blank to search all libraries.
5. Enter the all or part of a function name (or use % wildcard) and click **Search** to list functions within the library.
6. Click the **Modify** link for the function.
7. Enter Comments and Test Plan description information for the function.
8. Click **Submit**.

9.2.7.6 Adding a Telnet Function Library

The Telnet function library requires a third-party Java library to be added to the OpenScript installation that will be used to playback functional scripts.

To add the Telenet function llibrary:

1. Browse to Java Telnet website at http://javatelnet.org/space/download.
2. Choose the Binary Release jta26.jar (250k) Executable Jar file.
3. Replace the empty stub jta26.jar inside the OATS-Repository/TELNETLIB/jar with the downloaded file.

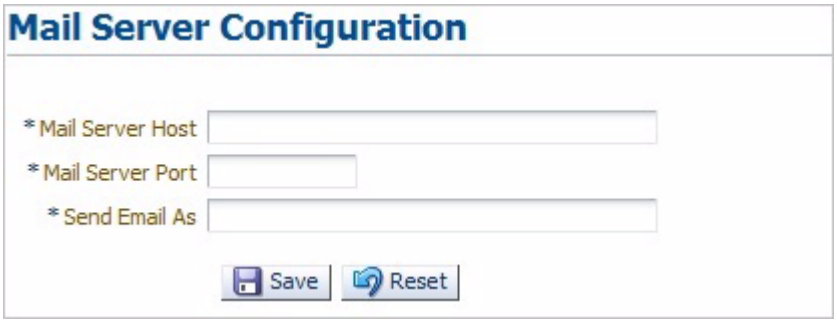
9.2.8 Setting Up Email

This section explains the procedures for administering the mail server within Oracle Flow Builder. The Notifications feature of Oracle Flow Builder uses Email to send notifications to users and administrators. The SMTP mail server must be defined in the Mail Server setup before Email notifications are activated.

Follow these steps to specify the mail server in the Oracle Flow Builder application:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Setup**, click **Mail Server**.

Figure 9–16 Mail Server Configuration Options



3. Enter the name of the mail server host. The mail server host is required for sending Email notifications.
4. Enter a numeric SMTP (Mail) Server Port value. The standard SMTP port is 25, unless some other specific port is in use.
5. Enter an Email address to use as the sent by address. Email Notifications will appear to have been be sent from this address.
6. Click **Save**.
7. Verify the Email notifications are working correctly by performing an action that will trigger an Email notification. See Chapter 6, "Using Notifications" for additional information.

9.3 Managing Product Family Access

An Administrator sets the access and role users are given for specific product families defined in the Component Tree. The following tasks are performed by an the Product Family owner administrator:

- Adding User Access to a Product Family
- Updating User Access Role for a Product Family
- Removing User Access

9.3.1 Adding User Access to a Product Family

Users can request access to one or more Product Families using the **Request for Access** option on the Oracle Flow Builder **Home** page. The request will be sent to the respective Product Family owner. The Product Family owner uses the Product Family Access Controls on the Administration page to add user access to a Product Family. Upon the approval by the Product Family owner, the user will be notified via an email.

To add user access to a Product Family:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Product Family Access Controls**, click **Users**.
3. Click **Add**.

Figure 9–17 Add Access Control Options

4. Select the user.
5. Select the Product Family.
6. Select the Role to assign to the user for the Product Family and click **Save**.

9.3.2 Updating User Access Role for a Product Family

An administrator can update a user's access to Product Families using Product Family Access Controls on the Administration page.

To update user access to a Product Family:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Product Family Access Controls**, click **Users**.
3. Select the Product Family.
4. Enter the ID of the user (or use % wildcard).
5. Click **Search** to list users matching the search criteria.
6. Click the pencil icon in the Update column of the user row.

Figure 9–18 Update Access Role Options

7. Select the Role and click **Save** to update the user access role.

9.3.3 Removing User Access

An administrator can remove a user's access to Product Families using Product Family Access Controls on the Administration page.

To remove user access to a Product Family:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Product Family Access Controls**, click **Users**.
3. Select the Product Family.
4. Enter the ID of the user (or use % wildcard).
5. Select the check box in the **Select** column next to the user to remove.
6. Click **Remove**.

A

Keyword Reference

This appendix lists the keywords and objects used to specify component code in Oracle Flow Builder. It contains the following sections:

A.1 Keywords and Objects

The following table lists the Keywords and valid objects available to use to specify component code.

Table A–1 Keywords and Objects Reference

Keywords	Description	Valid Objects
ACTIVATE	Activate the specified object.	The following objects are valid: ■ ALERT ■ CHOICEBOX ■ WINDOW
APPROVE	Approve the specified object.	The following objects are valid: ■ ALERT ■ CHOICEBOX ■ WINDOW
CALGETDATE	Get the Calendar date of the object.	The following objects are valid: ■ MISC
CALSETDATE	Set the Calendar date of the object.	The following objects are valid: ■ MISC
CANCEL	Cancel the specified object.	The following objects are valid: ■ ALERT ■ CHOICEBOX ■ FLEXWINDOW
CHECK	Check a checkbox object.	The following objects are valid: ■ CHECKBOX

Table A–1 (Cont.) Keywords and Objects Reference

Keywords	Description	Valid Objects
CLICK	Click the specified object.	The following objects are valid: ■ ALERTBUTTON ■ BUTTON ■ CHOICEBOXBUTTON ■ EDIT ■ FLEXCANCEL ■ FLEXCOMBINATION ■ FLEXOK ■ IMAGE ■ LINK ■ TAB ■ TOOLBAR
CLOSE	Close the specified window object.	The following objects are valid: ■ WINDOW
COLLAPSE	Collapse the specified tree object.	The following objects are valid: ■ TREE
COLLAPSENODE	Close the specified tree node object.	The following objects are valid: ■ TREE ■ TREELIST
ENDCATCH	Specify the end of a Catch Keyword set. Used with STARTCATCH.	No objects required.
ENDGROUP	Specify the end of a Group Keyword set. Used with STARTGROUP.	No objects required.
ENDITERATE	Specify the end of an iterate Keyword set. Used with STARTITERATE.	No objects required.
ENDKEY	Specify the end of a Key Keyword set. Used with STARTKEY.	No objects required.
ENDOPTIONAL	Specify the end of an Optional Keyword set. Used with STARTOPTIONAL.	No objects required.
ENDRECOVERY	Specify the end of a Recovery Keyword set. Used with STARTRECOVERY.	No objects required.
ENDTAB	Specify the end of a Tab Keyword set. Used with STARTTAB.	No objects required.
ENDXLTBLVERIFY	Specify the end of an Excel table verify Keyword set. Used with STARTXLTBLVERIFY.	No objects required.
ENDXLVERIFY	Specify the end of an Excel verify Keyword set. Used with STARTXLVERIFY.	No objects required.

Table A–1 (Cont.) Keywords and Objects Reference

Keywords	Description	Valid Objects
EXISTS	Check if the specified object exists.	The following objects are valid: ■ ALERT ■ BUTTON ■ CHOICEBOX ■ IMAGE ■ LINK ■ LIST ■ LOV ■ RADIOBUTTON ■ SPREADTABLE ■ TEXTAREA ■ WINDOW
EXPANDNODE	Expand the specified tree node object.	The following objects are valid: ■ TREE ■ TREELIST
FIREEVENTBLUR	Fire the Blur event on the specified object.	The following objects are valid: ■ CHECKBOX ■ EDIT ■ LIST ■ LISTBOX ■ RADIOBUTTON ■ TEXTAREA
FIREEVENTONCHANGE	Fire the OnChange event on the specified object.	The following objects are valid: ■ CHECKBOX ■ EDIT ■ LIST ■ LISTBOX ■ RADIOBUTTON ■ TEXTAREA
FUNCTIONCALL	Call a function from the specified function library.	The following libraries are valid: ■ cRMLIB ■ eBSLibrary ■ fINLIB ■ gENLIB ■ hRMSLIB ■ pRJTBLVERIFYLIB ■ pROCLIB ■ pROJLIB ■ sCMLIB ■ tELNETLIB ■ wEBTABLELIB

Table A–1 (Cont.) Keywords and Objects Reference

Keywords	Description	Valid Objects
GET	Get the specified object.	The following objects are valid: ■ ALERT ■ CHOICEBOX ■ EDIT ■ FIELD ■ LINK ■ LIST ■ LISTBOX ■ SPREADCELL ■ STATUS ■ TEXT ■ TEXTAREA
GETATTRIBUTE	Get the attributes of the specified object.	The following objects are valid: ■ BUTTON ■ EDIT ■ CHECKBOX ■ LIST ■ LOV ■ RADIOBUTTON ■ WINDOW
GETCELLDATA	Get the cell data of the specified table object.	The following objects are valid: ■ TABLE
GETITEMVALUE	Get the value of the specified list object.	The following objects are valid: ■ LIST
INVOKESOFTKEY	Invoke the soft key on the specified object.	The following objects are valid: ■ EDIT ■ SPREADTABLE
LAUNCH	Launch the specified browser object.	The following objects are valid: ■ BROWSER
MAXIMIZE	Maximize the specified window object.	The following objects are valid: ■ WINDOW
MAXVISIBLELINES	Set the maximum number of visible lines in a table object.	No objects required.
MENUSELECT	Select the specified menu object.	The following objects are valid: ■ CONTEXTMENU ■ MAINMENU ■ TREE
MINIMIZE	Minimize the specified window object.	The following objects are valid: ■ WINDOW
NAVIGATE	Navigate to the specified URL.	No objects required.

Table A–1 (Cont.) Keywords and Objects Reference

Keywords	Description	Valid Objects
PRESSTABKEY	Perform a tab key press on the specified object.	The following objects are valid: ■ EDIT
SEARCHBYDYNAMICCOLUMN	Search by dynamic column.	No objects required.
SEARCHCOLUMN	Search by column.	No objects required.
SEARCHEMPTY	Search for an empty column.	No objects required.
SELECT	Perform a select action on the specified object.	The following objects are valid: ■ EDIT ■ LIST ■ LISTBOX ■ RADIOBUTTON ■ TREELIST
SELECTALLROWS	Perform a select all rows action on the specified spreadtable object.	The following objects are valid: ■ SPREADTABLE
SELECTLOV	Perform a select action on the LOV object.	No objects required.
SELECTNODE	Perform a select node action on the specified tree object.	The following objects are valid: ■ TREE
SELECTROW	Perform a select row action on the specified spreadtable object.	The following objects are valid: ■ SPREADTABLE
SENDKEY	Perform a send key action on the specified edit object.	The following objects are valid: ■ EDIT
SETAPPTYPE	Set the application type. This is typically the first keyword specified in the component code.	The following application types are valid: ■ FORMFLEX ■ FORMS ■ JTT ■ TELNET ■ WEB
SETCURRENTROW	Set the current row.	No objects required.
SETFOCUS	Set the focus on the specified object.	The following objects are valid: ■ EDIT ■ FIRSTRECORD ■ TEXTAREA ■ TREELIST
SETLINE	Set the current line.	No objects required.
SETSPREADTABLE	Set the spreadtable.	No objects required.
SETTABLENAME	Set the table name.	No objects required.

Table A–1 (Cont.) Keywords and Objects Reference

Keywords	Description	Valid Objects
SETTEXT	Set the text on the specified object.	The following objects are valid: ■ DYNAMICEDIT ■ EDIT ■ FIELD ■ PASSWORD ■ TEXTAREA
SETWINDOW	Set the window.	No objects required.
STARTCATCH	Specify the start of a Catch Keyword set. Used with ENDCATCH.	No objects required.
STARTGROUP	Specify the start of a Group Keyword set. Used with ENDGROUP.	No objects required.
STARTITERATE	Specify the start of an Iterate Keyword set. Used with ENDITERATE.	No objects required.
STARTKEY	Specify the start of a Key Keyword set. Used with ENDKEY.	No objects required.
STARTOPTIONAL	Specify the start of an Optional Keyword set. Used with ENDOPTIONAL.	No objects required.
STARTRECOVERY	Specify the start of a Recovery Keyword set. Used with ENDRECOVERY.	No objects required.
STARTTAB	Specify the start of a Tab Keyword set. Used with ENDTAB.	No objects required.
STARTXLTBLVERIFY	Specify the start of an Excel table verify Keyword set. Used with ENDXLTBLVERIFY.	No objects required.
STARTXLVERIFY	Specify the start of an Excel verify Keyword set. Used with ENDXLVERIFY.	No objects required.

Table A–1 (Cont.) Keywords and Objects Reference

Keywords	Description	Valid Objects
UNCHECK	Uncheck the specified checkbox object.	The following objects are valid: ■ CHECKBOX
VERIFY	Verify the specified object.	The following objects are valid: ■ CHECKBOX ■ CHOICEBOX ■ EDIT ■ IMAGE ■ LINK ■ LIST ■ LISTBOX ■ RADIOBUTTON ■ SPREADCELL ■ STATUSBAR ■ TEXT ■ TEXTAREA
WAIT	Wait for the specified object.	The following objects are valid: ■ BUTTON ■ EDIT ■ IMAGE ■ LINK ■ LIST ■ LISTBOX ■ NORMAL ■ TEXTAREA ■ WINDOW

B

Function Library Reference

This appendix lists the function libraries and functions available to specify component code in Oracle Flow Builder. The function libraries and functions are used with the FUNCTIONCALL Keyword in the component code and related the test data in flows.

B.1 Installed Function Libraries

The following table lists the function libraries available to use to specify component code.

Table B–1 Function Libraries Installed with Oracle Flow Builder

Library	Related Application/Description
cRMLIB	Customer Relationship Management
eBSLibrary	EBS Forms applications
gENLIB	Generic Library
pRJTBLVERIFYLIB	Projects
pROCLIB	Procurement
pROJLIB	Projects
sCMLIB	Supply Chain Management
tELNETLIB	Telnet (requires a third-party Java library to be added to the OpenScript repository where Oracle Flow Builder-generated scripts will be executed. See Adding a Telnet Function Library for additional information).
wEBTABLELIB	Web Tables

The following sections provide details about the functions in each library.

B.1.1 Function Parameters

Various functions in the libraries use the following parameters:

Table B–2 List of Parameter Types used with Functions

Parameter Label	Description
@after	Specifies that the value after the specified text is used. The @after text is specified in the Attribute Value column of the component code. For example, @after='Child Labor Laws Compliance',@uitype='checkbox'.

Table B–2 (Cont.) List of Parameter Types used with Functions

Parameter Label	Description
@before	Specifies that the value before the specified text is used. The @before text is specified in the Attribute Value column of the component code. For example, @before='Child Labor Laws Compliance',@uitype='checkbox'.
@caption	Specifies the caption used to identify an object. The @caption is specified in the Display Name column of the component code. For example, @caption='Title'.
@logical	Specifies a True/False value. The @logical values is specified in the Attribute Value column in the component code. For example, @logical='True'.
@param#	Specifies the Test Data parameter(s) required for the function. Test Data is specified in the Flow test data.
@uitype	Specifies the type of UI component. The @uitype is specified in the Attribute Value column of the component code corresponding to the component type of the UI component. Valid values for @uitype are: select, textarea, input, checkbox.
@window	Refers to current window that is in context.

B.2 cRMLIB Function Library

The cRMLIB function library is used to develop component code and flows for Customer Relationship Management applications.

B.2.1 addToCartItemDetails

Clicks on the Item Details icon and Add to cart button for the specified item and quantity.

Test Data

The addToCartItemDetails function requires the following Test Data:

@param1, @param2 as Item label, Quantity.

B.2.2 cartCheckout

Click on Checkout for the specified cart type.

Test Data

The cartCheckout function requires the following Test Data:

@param1 as Cart Type.

B.2.3 checkImageCheckBox

Checks the specified Checkbox.

Test Data

The checkImageCheckBox function requires the following Test Data:

@logical, @param1 as Check (True / False).

B.2.4 clickAddToCart

Clicks on AddtoCart for the specified Item Name.

Test Data

The clickAddToCart function requires the following Test Data:

@param1 as Item Name.

B.2.5 clickConfigure

Clicks on Configure for the specified Item Name.

Test Data

The clickConfigure function requires the following Test Data:

@param1 as Item Name.

B.2.6 clickExpressCheckOut

Clicks on Express Checkout for the specified Item Name.

Test Data

The clickExpressCheckOut function requires the following Test Data:

@param1 as Item Name.

B.2.7 clickImageInInnerNavigationTable

Click on the image in the table with inner navigation.

Test Data

The clickImageInInnerNavigationTable function requires the following Test Data:

@param1, @param2, @param3, @param4, @param5 as Table Name, Navigation Column, Search Column, Search Value, Target Column.

B.2.8 clickLOVBasedOnLabel

Clicks on the Torch icon for the specified field.

Test Data

The clickLOVBasedOnLabel function requires the following Test Data:

@param1 as Label Name.

B.2.9 clickSiteLink

Clicks on the specified SiteLink in iStore.

Test Data

The clickSiteLink function requires the following Test Data:

@param1 as Site Name to Click.

B.2.10 clickTableImage

Clicks the image in the specified table.

Test Data

The clickTableImage function requires the following Test Data:

@caption, @param1, @param2, @param3 as Column Number To Verify , Search text, ColumnName To Select.

B.2.11 crmWebSelectLOV

Selects the specified values from the Search and Select list of values for the specified field.

Test Data

The crmWebSelectLOV function requires the following Test Data:

@param1, @param2, @param3, @param4, @param5 as LovName, SearchByOption, SearchValue, ColName, RowValue.

B.2.12 expandBasedOnLabel

Expands the specified label name.

Test Data

The expandBasedOnLabel function does not require Test Data.

B.2.13 expandCollapseBasedOnLabel

Expands or collapses the specified label.

Test Data

The expandCollapseBasedOnLabel function does not require Test Data.

B.2.14 getRequestStatus

Click the refresh button until the request status is completed with Request ID to get the Refresh the status of the request.

Test Data

The getRequestStatus function requires the following Test Data:

@param1 as Request ID.

B.2.15 jttLogin

Login to the wireless application with the specified username and password.

Test Data

The jttLogin function requires the following Test Data:

@param1, @param2 as Username, Password.

B.2.16 refreshWebItem

Click on the specified button until the status changes to the required value.

Test Data

The refreshWebItem function requires the following Test Data:

@param1, @param2, @param3, @param4, @param5, @param6 as Button Name, Table Name, Source Column, Source Column Value, Target Column, Target Column Value.

B.2.17 searchEditableRow

Searches For Editable row

Test Data

The searchEditableRow function does not require Test Data.

B.2.18 selectAddress

Selects the specified address.

Test Data

The selectAddress function requires the following Test Data:

@param1 as Address.

B.2.19 selectCartAction

Select the action to be performed on the specified cart type.

Test Data

The selectCartAction function requires the following Test Data:

@param1, @param2 as Cart Type, Item To Select.

B.2.20 selectCustomer

Selects the specified customer by value and account number

Test Data

The selectCustomer function requires the following Test Data:

@param1, @param2, @param3 as Search For, Search For Value, Account Number.

B.2.21 selectDisplayTemplate

Selects the radio button with the specified label name.

Test Data

The selectDisplayTemplate function requires the following Test Data:

@param1 as Label Name.

B.2.22 selectFormsSingleColValues

Selects the multiple (comma separated) resources.

Test Data

The selectFormsSingleColValues function requires the following Test Data:

@logical, @param1 as comma separated resources.

B.2.23 selectImageRadiobutton

Selects the specified Radio Button.

Test Data

The selectImageRadiobutton function requires the following Test Data:

@logical, @param1 as Select [True / False].

B.2.24 selectMediaContent

Select the specified value from Search and Select list of values.

Test Data

The selectMediaContent function requires the following Test Data:

@param1, @param2 as Search For, Value to Select.

B.2.25 setCartQuantity

Sets the cart quantity for the specified Item name.

Test Data

The setCartQuantity function requires the following Test Data:

@param1, @param2 as Item Name, Quantity.

B.2.26 setSearchParams

Sets the search parameters to handle the search criteria for different values in the listbox.

Test Data

The setSearchParams function requires the following Test Data:

@param1 as SearchForName, OperatorValue, ValueToSet.

B.2.27 verifyBatchStatus

Verifies the batch status.

Test Data

The verifyBatchStatus function requires the following Test Data:

@param1 as Batch Status.

B.2.28 verifyDateBasedOnMonday

Verify if the date is Monday.

Test Data

The verifyDateBasedOnMonday function requires the following Test Data:

@logical, @param1, @param2, @param3 as time, dateFormat, minutesToBeAdded.

B.2.29 verifyJobStatus

Verifies the job status.

Test Data

The verifyJobStatus function requires the following Test Data:

@param1 as Job Status.

B.2.30 webClickDynamicLink

Clicks on the specified Link Name.

Test Data

The webClickDynamicLink function requires the following Test Data:

@param1 as LinkName.

B.3 eBSLibrary Function Library

The eBSLibrary function library is used to develop component code and flows for EBS Forms applications.

B.3.1 addFailedResult

Adds a "Failed" result to the Oracle Application Testing Suite results file for the specified Step Name.

Test Data

The addFailedResult function requires the following Test Data:

@param1, @param2 as Step Name, Comment.

B.3.2 addPassedResult

Adds a "Pass" result to the Oracle Application Testing Suite results file for the specified Step Name.

Test Data

The addPassedResult function requires the following Test Data:

@param1, @param2 as Step Name, Comment.

B.3.3 oracle_close_all_browsers

Closes all open browsers and EBS Forms.

Test Data

The oracle_close_all_browsers function does not require Test Data.

B.3.4 oracle_date_manipulation

Returns a date value based on the format and manipulations specified as input.

Test Data

The oracle_date_manipulation function requires the following Test Data:

@param1, @param2, @param3, @param4 as Date Format, Days, Months, Years.

B.3.5 oracle_exit_app

Exits the EBS Forms application and closes all browsers.

Test Data

The oracle_exit_app function does not require Test Data.

B.3.6 oracle_form_initial_condition

Sets the initial state of the EBS Forms Navigator window.

Test Data

The oracle_form_initial_condition function does not require Test Data.

B.3.7 oracle_formWindow_close

Closes the specified EBS Forms window.

Test Data

The oracle_formWindow_close function requires the following Test Data:

@param1 as Title.

B.3.8 oracle_homepage_nav

Navigates to the specified responsibility, menu path, and function in an EBS Application home page.

Test Data

The oracle_homepage_nav function requires the following Test Data:

@param1, @param2, @param3 as Resp, Menu Path, Menu Choice.

B.3.9 oracle_input_dialog

Prompts the user for an input using the specified message title.

Test Data

The oracle_input_dialog function requires the following Test Data:

@param1 as Prompt Message Title.

B.3.10 oracle_launch_istore_url

Launches an iStore URL.

Test Data

The oracle_launch_istore_url function does not require Test Data.

B.3.11 oracle_launch_jsp_url

Launches a JSP URL.

Test Data

The oracle_launch_jsp_url function does not require Test Data.

B.3.12 oracle_launch_php_url

Launches a PHP URL.

Test Data

The oracle_launch_php_url function does not require Test Data.

B.3.13 oracle_menu_select

Selects the specified menu path in an EBS Forms window.

Test Data

The oracle_menu_select function requires the following Test Data:

@param1 as Menu Path.

B.3.14 oracle_navigation_menu

Navigates to the specified menu and function in the EBS Application home page.

Test Data

The oracle_navigation_menu function requires the following Test Data:

@param1, @param2 as Menu Path, Menu Function.

B.3.15 oracle_navigator_select

Selects the navigation path in the EBS Forms Navigator window.

Test Data

The oracle_navigator_select function requires the following Test Data:

@param1 as Navigation Path.

B.3.16 oracle_php_login

Logs in to a PHP URL with the specified user credentials.

Test Data

The oracle_php_login function requires the following Test Data:

@param1, @param2 as User Name, Password.

B.3.17 oracle_php_signon

Logs in to a PHP URL with the specified user credentials and clicks on the specified responsibility of an EBS Application.

Test Data

The oracle_php_signon function requires the following Test Data:

@param1, @param2, @param3 as User, Password, Resp.

B.3.18 oracle_prompt_sql

Prompts the user to provide a SQL URL.

Test Data

The oracle_prompt_sql function does not require Test Data.

B.3.19 oracle_prompt_url

Prompts the user to provide instance URLs.

Test Data

The oracle_prompt_url function does not require Test Data.

B.3.20 oracle_statusbar_msgck

Checks the EBS Forms status bar message against the specified expected value.

Test Data

The oracle_statusbar_msgck function requires the following Test Data:

@param1 as Expected Status Bar Message.

B.3.21 oracle_switch_responsibility

Switches to the specified responsibility in an EBS Form.

Test Data

The oracle_switch_responsibility function requires the following Test Data:

@param1 as Responsibility Name.

B.3.22 oracle_table_objClick

Clicks an image in the specified table.

Test Data

The oracle_table_objClick function requires the following Test Data:

@param1, @param2 as UniqueIdentifier, Column, State.

B.4 gENLIB Function Library

The gENLIB function library is used to develop component code and flows for General applications.

B.4.1 actOnAssignment

Clicks on update or correct button based on the specified input.

Test Data

The actOnAssignment function requires the following Test Data:

@param1 as Click Update / Correction.

B.4.2 addPassFailResult

Adds pass or fail result to the OATS result file based on input.

Test Data

The addPassFailResult function requires the following Test Data:

@param1, @param2 as Step Name, Comment.

B.4.3 alterEffectiveDate

Sets effective date to the specified value.

Test Data

The alterEffectiveDate function requires the following Test Data:

@param1 as Date.

B.4.4 clickFlexOK

Clicks the Ok button on a flex window.

Test Data

The clickFlexOK function does not require Test Data.

B.4.5 clickHide

Clicks the hide link.

Test Data

The clickHide function does not require Test Data.

B.4.6 closeForm

Closes the current form.

Test Data

The closeForm function does not require Test Data.

B.4.7 closeForms

Closes all open forms.

Test Data

The closeForms function does not require Test Data.

B.4.8 closeWebPage

Close the specified web page.

Test Data

The closeWebPage function requires the following Test Data:

@param1 as Title Name.

B.4.9 expandAndSelectNode

Expands and selects tree nodes in EBS Form window.

Test Data

The expandAndSelectNode function requires the following Test Data:

@logical, @param1 as Navigation Path.

B.4.10 expandNodes

Expands the specified tree nodes in EBS Form window.

Test Data

The expandNodes function requires the following Test Data:

@logical, @param1 as Navigation Path.

B.4.11 extractNumber

Extracts a number from a specified string.

Test Data

The extractNumber function requires the following Test Data:

@param1 as Text.

B.4.12 extractZipFile

Extracts the zip file to the specified location.

Test Data

The extractZipFile function requires the following Test Data:

@param1 as File Name.

B.4.13 formHideField

Hides the current field in an EBS Form window.

Test Data

The formHideField function requires the following Test Data:

@window, @logical as FieldName.

B.4.14 formMenuSelect

Selects the specified menu option in an EBS Form window.

Test Data

The formMenuSelect function requires the following Test Data:

@param1 as Main Menu Path.

B.4.15 formsChoiceWindow

Clicks the Ok button in an EBS Forms decision box.

Test Data

The formsChoiceWindow function does not require Test Data.

B.4.16 formsConfirmDialog

Clicks the Yes button in an EBS Forms alert dialog.

Test Data

The formsConfirmDialog function does not require Test Data.

B.4.17 formSelectDate

Selects the specified date from an EBS Forms calendar.

Test Data

The formSelectDate function requires the following Test Data:

@logical, @param1 as DateValue.

B.4.18 formSelectLOV

Selects a value from a forms select list of values window.

Test Data

The formSelectLOV function requires the following Test Data:

@logical, @param1 as Value To Select.

B.4.19 formSetValueInDynamicColumn

Sets a value in a dynamic column.

Test Data

The formSetValueInDynamicColumn function requires the following Test Data:

@param1, @param2, @param3 as Forms Dynamic Column Name, Row Number, Value To Set.

B.4.20 formShowField

Shows a specified field in an EBS Form window.

Test Data

The formShowField function requires the following Test Data:

@window, @param1 as | separated field name and value or field name to show.

B.4.21 formsSelectColor

Selects color from an EBS Forms color picker window.

Test Data

The formsSelectColor function requires the following Test Data:

@logical, @param1 as Color Value.

B.4.22 formsVerifyTextArea

Verifies the value of a textarea located in an EBS Form window.

Test Data

The formsVerifyTextArea function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.23 formVerifyCheckBox

Verifies the status of an EBS Forms check box.

Test Data

The formVerifyCheckBox function requires the following Test Data:

@logical, @param1 as CheckBox Status.

B.4.24 formVerifyEdit

Verifies text field values on an EBS Form window.

Test Data

The formVerifyEdit function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.25 formVerifyList

Verifies a list value in an EBS Form window.

Test Data

The formVerifyList function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.26 formVerifyListBox

Verifies a list box value in an EBS Form window.

Test Data

The formVerifyListBox function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.27 formVerifyListBoxValues

Verifies if the specified values exists in an EBS Forms list box.

Test Data

The formVerifyListBoxValues function requires the following Test Data:

@param1 as expectedValue1, expectedValue2, expectedValue3[...].

B.4.28 formVerifyRadioButton

Verifies the status of an EBS Forms radio button.

Test Data

The formVerifyRadioButton function requires the following Test Data:

@logical, @param1 as Radio Button Status.

B.4.29 formVerifyStatus

Verifies an EBS Forms status bar message.

Test Data

The formVerifyStatus function requires the following Test Data:

@param1 as Message.

B.4.30 getNumbersFromStr

Extracts numbers from a specified string.

Test Data

The getNumbersFromStr function requires the following Test Data:

@param1, @param2, @param3, @param4 as Message, before, after, index.

B.4.31 getRandomNumber

Returns a random number.

Test Data

The getRandomNumber function requires the following Test Data:

@param1 as MaxRange.

B.4.32 getSysDate

Gets the current date based on the specified format.

Test Data

The getSysDate function requires the following Test Data:

@param1, @param2 as Format, Numberofdays[0/+ve/-ve].

B.4.33 getSysDateTime

Gets the current date and time based on the specified format.

Test Data

The getSysDateTime function requires the following Test Data:

@param1, @param2 as Format, Numberofdays[0/+ve/-ve].

B.4.34 getValueBasedonLabelAfterUIComponent

Gets a value from a field based on the label present after the field.

Test Data

The getValueBasedonLabelAfterUIComponent function does not require Test Data.

B.4.35 getValueBasedonLabelBeforeUIComponent

Gets a value from a field based on the label present before the field.

Test Data

The getValueBasedonLabelBeforeUIComponent function does not require Test Data.

B.4.36 handleDialog

Approves or rejects the dialog window based on the input provided.

Test Data

The handleDialog function requires the following Test Data:

@param1 as Action to Perform (TRUE / FALSE).

B.4.37 handleMicrosoftAlert

Handle the Microsoft alerts.

Test Data

The handleMicrosoftAlert function does not require Test Data.

B.4.38 handleSSL

Handles the ssl alerts.

Test Data

The handleSSL function does not require Test Data.

B.4.39 navigateToHome

Navigates to the EBS Application home page.

Test Data

The navigateToHome function does not require Test Data.

B.4.40 openInventoryPeriod

Opens inventory periods for specified list of periods.

Test Data

The openInventoryPeriod function requires the following Test Data:

@param1 as Comma separated periods.

B.4.41 oracle_prompt_jtt_url

Prompts the user to provide a jtt URL.

Test Data

The oracle_prompt_jtt_url function does not require Test Data.

B.4.42 saveDialog

Saves the downloadable file to the specified location.

Test Data

The saveDialog function requires the following Test Data:

@param1 as Loction.

B.4.43 selectFile

Browses and selects a file in EBS OAF / web page.

Test Data

The selectFile function requires the following Test Data:

@logical, @param1 as File Path.

B.4.44 selectListMultiValues

Selects multiple values in a list box.

Test Data

The selectListMultiValues function requires the following Test Data:

@logical, @param1 as Comma separated values to Select.

B.4.45 setEditValueBasedOnLabel

Sets a value in an edit field based on the specified label.

Test Data

The setEditValueBasedOnLabel function requires the following Test Data:

@param1, @param2 as Label Name, Value to set.

B.4.46 setFlexText

Sets a value in text field of a flex window.

Test Data

The setFlexText function requires the following Test Data:

@param1, @param2 as Text Field Name, Value.

B.4.47 setFormsText

Sets the specified value in EBS Forms text field.

Test Data

The setFormsText function requires the following Test Data:

@logical, @param1 as Value To Set.

B.4.48 setPayablePeriods

Sets payable periods to a status as per the specified input.

Test Data

The setPayablePeriods function requires the following Test Data:

@param1, @param2, @param3, @param4 as Ledger, Operating Unit, Comma separated Periods, Comma separated status.

B.4.49 setPurchasingPeriod

Sets the specified purchasing periods to the specified status respectively.

Test Data

The setPurchasingPeriod function requires the following Test Data:

@param1, @param2 as Comma separated periods, Comma separated status.

B.4.50 setRadioValueBasedonLabelAfterUIComponent

Selects a radio button based on the label present after the button.

Test Data

The setRadioValueBasedonLabelAfterUIComponent function requires the following Test Data:

@param1 as After Text For Radio.

B.4.51 setRadioValueBasedonLabelBeforeUIComponent

Selects a radio button based on the label present before the button.

Test Data

The setRadioValueBasedonLabelBeforeUIComponent function requires the following Test Data:

@param1 as Before Text For Radio.

B.4.52 setValueBasedonLabelAfterUIComponent

Sets or selects a value in a field based on the label present after the field.

Test Data

The setValueBasedonLabelAfterUIComponent function requires the following Test Data:

@after, @uitype, @param1 as Value to set.

B.4.53 setValueBasedonLabelBeforeUIComponent

Sets or selects a value in a field based on the label present before the field.

Test Data

The setValueBasedonLabelBeforeUIComponent function requires the following Test Data:

@before, @uitype, @param1 as Label before UI component, UI Component type, Value to set.

B.4.54 SHOWALLFIELDS

Shows all fields in an EBS Form window.

Test Data

The SHOWALLFIELDS function does not require Test Data.

B.4.55 switchResponsibility

Switches to the specified responsibility in EBS Forms.

Test Data

The switchResponsibility function requires the following Test Data:

@param1 as Value To Select.

B.4.56 uploadFile

Uploads the specified file.

Test Data

The uploadFile function requires the following Test Data:

@param1 as File Name.

B.4.57 verifyAndClosePopup

Verifies a message on a popup and closes the popup.

Test Data

The verifyAndClosePopup function requires the following Test Data:

@param1, @logical as Popup message to verify.

B.4.58 verifyParenetChildReqs

Verifies the parent and child request status based on the specified input.

Test Data

The verifyParenetChildReqs function requires the following Test Data:

@param1, @param2, @param3, @param4, @param5 as Parent ReqID, Parent ReqIndex, Parent Status, comma separated Child Request Names, comma separated Child Req Statuses.

B.4.59 verifyRequestStatus

Verifies the request status with the specified value.

Test Data

The verifyRequestStatus function requires the following Test Data:

@param1 as Expected Value.

B.4.60 verifyValueBasedonLabelAfterUIComponent

Verifies the value of a field based on the label present after the field.

Test Data

The verifyValueBasedonLabelAfterUIComponent function requires the following Test Data:

@after, @uitype, @param1 as Value to verify.

B.4.61 verifyValueBasedonLabelBeforeUIComponent

Verifies the value of a field based on the label present before the field.

Test Data

The verifyValueBasedonLabelBeforeUIComponent function requires the following Test Data:

@before, @uitype, @param1 as Value to verify.

B.4.62 verifyValueInDynamicColumn

Verifies a value in the specified column of web table.

Test Data

The verifyValueInDynamicColumn function requires the following Test Data:

@param1, @param2, @param3 as Column Name, Row Number, Expected Value.

B.4.63 webClickButton

Searches a specified button name in an EBS OAF / web page and clicks the button.

Test Data

The webClickButton function does not require Test Data.

B.4.64 webClickDynamicLink

Searches the specified link name on an EBS OAF / web page and clicks the link.

Test Data

The webClickDynamicLink function requires the following Test Data:

@param1 as LinkName.

B.4.65 webClickImage

Clicks an image on an EBS OAF / web page.

Test Data

The webClickImage function does not require Test Data.

B.4.66 webClickLink

Clicks a link in a EBS OAF / web page.

Test Data

The webClickLink function does not require Test Data.

B.4.67 webGetTextBasedOnLabel

Gets plain text from a web page which is present after the specified text.

Test Data

The webGetTextBasedOnLabel function does not require Test Data.

B.4.68 webLogout

Logs out from the EBS Application.

Test Data

The webLogout function does not require Test Data.

B.4.69 webSelectDate

Selects the specified date value from an EBS OAF calendar.

Test Data

The webSelectDate function requires the following Test Data:

@logical, @param1 as DateValue.

B.4.70 webSelectListBox

Selects a value in a list box.

Test Data

The webSelectListBox function requires the following Test Data:

@logical, @param1 as Value to Select.

B.4.71 webSelectLOV

Selects a value from Search and Select list of values.

Test Data

The webSelectLOV function requires the following Test Data:

@logical, @param1, @param2, @param3, @param4 as SearchByOption, SearchValue, ColName, RowValue.

B.4.72 webSetTextBasedOnLabel

Sets text in an OAF text field based on the label specified.

Test Data

The webSetTextBasedOnLabel function requires the following Test Data:

@caption, @param1 as Value To Set.

B.4.73 webSetTextWithLOV

Enters value in a text field and if a search and select list of values window appears, function will try to select the value from the new window.

Test Data

The webSetTextWithLOV function requires the following Test Data:

@logical,@param1,@param2,@param3,@param4 as searchByOption, searchValue, colName, rowValue.

B.4.74 webVerifyCheckBox

Verifies if the EBS OAF / web check box is checked or unchecked as per the specified input.

Test Data

The webVerifyCheckBox function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.75 webVerifyEdit

Verifies text field values on an EBS OAF / web page.

Test Data

The webVerifyEdit function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.76 webVerifyLinkBasedOnLabel

Verifies a link from a web page which is present after the specified text.

Test Data

The webVerifyLinkBasedOnLabel function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.77 webVerifyList

Verifies a value in list object.

Test Data

The webVerifyList function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.78 webVerifyListBoxValues

Verifies if values are present in EBS OAF / web list box.

Test Data

The webVerifyListBoxValues function requires the following Test Data:

@logical, @param1 as expectedValue1, expectedValue2, expectedValue3[...].

B.4.79 webVerifyListValues

Verifies if values are present in an EBS OAF / web list.

Test Data

The webVerifyListValues function requires the following Test Data:

@logical, @param1 as expectedValue1, expectedValue2, expectedValue3[...].

B.4.80 webVerifyRadioButton

Verifies the status of a radio button status in an EBS OAF / web page.

Test Data

The webVerifyRadioButton function requires the following Test Data:

@logical, @param1 as Radio Button Status.

B.4.81 webVerifyText

Verifies plain text on an EBS OAF / web page.

Test Data

The webVerifyText function requires the following Test Data:

@param1 as Text.

B.4.82 webVerifyTextArea

Verifies if the EBS OAF / web text area has the specified value.

Test Data

The webVerifyTextArea function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.83 webVerifyTextBasedOnLabel

Verifies plain text in a web page which is present after the specified text.

Test Data

The webVerifyTextBasedOnLabel function requires the following Test Data:

@logical, @param1 as Expected Value.

B.4.84 webVerifyTextWithAfter

Verifies the text on a EBS OAF / web page which is present after the specified string.

Test Data

The webVerifyTextWithAfter function requires the following Test Data:

@logical, @param1 as Expected value to verify.

B.4.85 webVerifyTextWithBefore

Verifies the text on a EBS OAF / web page which is present before the specified string.

Test Data

The webVerifyTextWithBefore function requires the following Test Data:

@logical, @param1 as Expected value to verify.

B.4.86 webVerifyTextWithBeforeAfter

Verifies text on an EBS OAF / web page which is present between two specified strings.

Test Data

The webVerifyTextWithBeforeAfter function requires the following Test Data:

@before, @after, @param1 as Expected value to verify.

B.5 pRJTBLVERIFYLIB Function Library

The pRJTBLVERIFYLIB function library is used to develop component code and flows for Projects applications.

B.5.1 setTableContext

Verify if the values present in web table match the values present in the specified Excel file.

Test Data

The setTableContext function requires the following Test Data:

@param1, @param2, @param3 as Table Name, Excel Name, Sheet Name.

B.6 pROCLIB Function Library

The pROCLIB function library is used to develop component code and flows for Procurement applications.

B.6.1 addAttachments

Adds attachments.

Test Data

The addAttachments function requires the following Test Data:

@param1, @param2, @param3 as Array of Field Labels, Array of Field Values, Action.

B.6.2 addIDVStructuretoCart

Adds the specified IDV structure to the cart based on the Item name and IDV number.

Test Data

The addIDVStructuretoCart function requires the following Test Data:

@param1, @param2, @param3 as Item Name, Enter IDV Number, enter price.

B.6.3 addItemFromFavToDocument

Adds the specified Item from the favorites list to the cart.

Test Data

The addItemFromFavToDocument function requires the following Test Data:

@param1, @param2 as Enter Item, Enter IDV Number (Optional).

B.6.4 addItemPricingDetails

Sets specified Item pricing detail to specified value.

Test Data

The addItemPricingDetails function requires the following Test Data:

@param1, @param2 as Array of Field Names, Array of Field Values.

B.6.5 addToCartBasedOnSource

Adds an item to the cart based on the Item name and Source document number.

Test Data

The addToCartBasedOnSource function requires the following Test Data:

@param1, @param2 as Item Name, Enter Source.

B.6.6 Award_Bid_To_Supplier

Fills in details in the Award By Bid page for the specified supplier and specified option.

Test Data

The Award_Bid_To_Supplier function requires the following Test Data:

@param1, @param2, @param3, @param4, @param5, @param6 as Supplier Name , Award, Award Option, Value To Enter, Internal Note , Note To Supplier.

B.6.7 awardTableAction

Enters text or selects the Award option of a Supplier in the Awards table.

Test Data

The awardTableAction function requires the following Test Data:

@param1, @param2, @param3, @param4 as Supplier name, Label value of line, Enter object type among:checkbox/radiobutton/textarea/textbox, Enter value.

B.6.8 carWebSelectLOV

Selects values from a List of Values (LOV) in the Car creation page, which does not have OAF UI.

Test Data

The carWebSelectLOV function requires the following Test Data:

@logical, @param1, @param2, @param3, @param4 as SearchByOption, SearchValue, ColName, RowValue.

B.6.9 clearShopppingCart

Clears the existing items in the iProcurement Shopping cart.

Test Data

The clearShopppingCart function does not require Test Data.

B.6.10 clickBidinAwardBid

Clicks on the Bid of a mentioned Supplier in the Award By Bid page.

Test Data

The clickBidinAwardBid function requires the following Test Data:

@param1 as Supplier Name .

B.6.11 editRequisitionNumber

Edits the Requisition number auto-generated to a user defined Requisition number.

Test Data

The editRequisitionNumber function requires the following Test Data:

@param1, @param2, @param3, @param4 as Enter Prefix, Enter Agency Identifier, Enter Allowed Range, Enter serial Number.

B.6.12 encryptURL

Generates a Encrypted URL for a specified operating unit for a Supplier to register to that Operating unit.

Test Data

The encryptURL function requires the following Test Data:

@param1 as Url to Encrypt.

B.6.13 formsSetChargeAccount

Sets the charge account for a distribution Line in the Distributions window.

Test Data

The formsSetChargeAccount function requires the following Test Data:

@param1, @param2 as chargeAccount, distributionLineNumber.

B.6.14 getAwardOption

Gets the specified Award option of the specified Supplier name in the Award by Bid table.

Test Data

The getAwardOption function requires the following Test Data:

@param1, @param2 as Supplier name, Label value of line.

B.6.15 getCheckboxValueBasedOnLabel

Gets the state of the checkbox next to the specified label.

Test Data

The getCheckboxValueBasedOnLabel function requires the following Test Data:

@param1 as Enter label name.

B.6.16 getEditValueBasedOnLabel

Gets the text in the textbox next to the specified label.

Test Data

The getEditValueBasedOnLabel function requires the following Test Data:

@param1 as Enter label name.

B.6.17 getOfferReceiveTime

Calculates Offer receive time to be used while creating Surrogate bid based on Negotiation open time.

Test Data

The getOfferReceiveTime function requires the following Test Data:

@param1, @param2 as Enter Date format of OpenTime (Optional), Enter the open time.

B.6.18 getSelectValueBasedOnLabel

Gets the selected option in the Selectbox next to the specified label.

Test Data

The getSelectValueBasedOnLabel function requires the following Test Data:

@param1 as Enter label name.

B.6.19 getTextAreaValueBasedOnLabel

Gets the text in the textarea next to the specified label.

Test Data

The getTextAreaValueBasedOnLabel function requires the following Test Data:

@param1 as Enter label name.

B.6.20 getValueBasedOnLabel

Gets a value in the specified type of component next to the label specified.

Test Data

The getValueBasedOnLabel function requires the following Test Data:

@caption, @param1 as Component Type.

B.6.21 handleEditDocumentNumber

Edits the Document number auto generated to a user defined Document number.

Test Data

The handleEditDocumentNumber function requires the following Test Data:

@param1, @param2, @param3, @param4 as DODAAC, Instrument Type, Allowed Range, Serial Number.

B.6.22 handleWebTermsWindow

Handles Terms and Accept condition window and Click on specified button in Terms and Accept condition window.

Test Data

The handleWebTermsWindow function requires the following Test Data:

@param1 as Button Name (Accept / Cancel).

B.6.23 launchCustomURL

Navigates to a generated custom URL.

Test Data

The launchCustomURL function requires the following Test Data:

@param1 as Enter custom URL.

B.6.24 selectApproverInManageApprovals

Selects a specific Approver from the Approver's list.

Test Data

The selectApproverInManageApprovals function requires the following Test Data:

@param1 as approver.

B.6.25 selectFirstOptionInSelectBoxBasedOnLabel

Selects the first available option in the Selectbox next to the specified label.

Test Data

The selectFirstOptionInSelectBoxBasedOnLabel function requires the following Test Data:

@param1 as Label name.

B.6.26 selectFirstValueFromLOV

Selects the First available value in a List Of Values (LOV).

Test Data

The selectFirstValueFromLOV function requires the following Test Data:

@param1 as Enter LOV Name.

B.6.27 selectRadiobuttonBasedonLabel

Selects the Radio button next to the specified label.

Test Data

The selectRadiobuttonBasedonLabel function requires the following Test Data:

@param1 as Select Label.

B.6.28 selectSearchRadioOption

Selects specified Show table data Radio button in Search criteria of Search window.

Test Data

The selectSearchRadioOption function requires the following Test Data:

@param1 as Search Radio Label.

B.6.29 setCheckboxValueBasedOnLabel

Checks or unchecks the checkbox next to the specified label.

Test Data

The setCheckboxValueBasedOnLabel function requires the following Test Data:

@param1, @param2 as Enter label name, Enter true or false to Check or uncheck.

B.6.30 setEditValueBasedOnLabel

Enters text in the textbox next to the specified label.

Test Data

The setEditValueBasedOnLabel function requires the following Test Data:

@param1, @param2 as Enter label name, Value to enter.

B.6.31 setNextRandomNumber

Enters a Random number in a field until that Random number is unique and not present in the Application database.

Test Data

The setNextRandomNumber function does not require Test Data.

B.6.32 setSelectValueBasedOnLabel

Selects an option in the Selectbox next to the specified label.

Test Data

The setSelectValueBasedOnLabel function requires the following Test Data:

@param1, @param2 as Enter label name, Value to select.

B.6.33 setTextAreaValueBasedOnLabel

Enters text in the textarea next to the specified label.

Test Data

The setTextAreaValueBasedOnLabel function requires the following Test Data:

@param1, @param2 as Enter label name, Value to enter.

B.6.34 setValueBasedOnLabel

Sets a value in the specified type of component next to the label specified.

Test Data

The setValueBasedOnLabel function requires the following Test Data:

@caption, @param1, @param2 as Type of the Component, Value to Set.

B.6.35 Verify_AwardBid_Details_Supplier

Verifies the details in the Award By Bid page for the specified supplier and specified option.

Test Data

The Verify_AwardBid_Details_Supplier function requires the following Test Data:

@param1, @param2, @param3 as Supplier Name , Award Option, ValuevTo VerifY.

B.6.36 verifyAwardOption

Verifies that the specified Award option of the specified Supplier name in the Award by Bid table matches the Expected value.

Test Data

The verifyAwardOption function requires the following Test Data:

@param1, @param2, @param3 as Enter Supplier name, Enter Label value of line, value to verify.

B.6.37 verifyCheckBoxImageBasedOnLabel

Verifies that the image ALT tag next to the specified label matches with ON/OFF checkbox image ALT tag.

Test Data

The verifyCheckBoxImageBasedOnLabel function requires the following Test Data:

@param1, @param2 as Label Name, Enter true/false for checked/unchecked status.

B.6.38 verifyCheckboxValueBasedOnLabel

Verifies the state of the checkbox next to the specified label .

Test Data

The verifyCheckboxValueBasedOnLabel function requires the following Test Data:

@param1, @param2 as Enter label name, Enter true or false to Check or uncheck.

B.6.39 verifyDocumentNumberDetails

Verifies whether or not the document number fields have expected values.

Test Data

The verifyDocumentNumberDetails function requires the following Test Data:

@param1, @param2 as Enter Label Name, Enter value.

B.6.40 verifyEditValueBasedOnLabel

Verifies the text in the textbox next to the specified label.

Test Data

The verifyEditValueBasedOnLabel function requires the following Test Data:

@param1, @param2 as Enter label name, Value to verify.

B.6.41 verifyItemPricingDetails

Verifies the Pricing details of an Item matches with expected value.

Test Data

The verifyItemPricingDetails function requires the following Test Data:

@param1, @param2 as Enter Label of Pricing Detail, Enter pricing Value.

B.6.42 verifyRequisitionNumber

Verifies whether Requisition number is in the expected format.

Test Data

The verifyRequisitionNumber function requires the following Test Data:

@param1, @param2, @param3, @param4 as Enter Prefix, Enter agencyIdentifier, Enter allowedRange, Enter delimiter.

B.6.43 verifySelectValueBasedOnLabel

Verifies the selected option in the Selectbox next to the specified label.

Test Data

The verifySelectValueBasedOnLabel function requires the following Test Data:

@param1, @param2 as Enter label name, Value to verify.

B.6.44 verifyTextAreaValueBasedOnLabel

Verifies the text in the textarea next to the specified label.

Test Data

The verifyTextAreaValueBasedOnLabel function requires the following Test Data:

@param1, @param2 as Enter label name, Value to verify.

B.6.45 verifyValueBasedOnLabel

Verifies a value in the specified type of component next to the label specified that matches with the expected value.

Test Data

The verifyValueBasedOnLabel function requires the following Test Data:

@caption, @param1, @param2 as Type of the Component, Value to Verify.

B.7 pROJLIB Function Library

The pROJLIB function library is used to develop component code and flows for Projects applications.

B.7.1 formsMinMaxViewOutput

Verifies a forms minimum/maximum output.

Test Data

The formsMinMaxViewOutput function requires the following Test Data:

@param1, @param2 as Item, Target Column Name.

B.7.2 leaseOpenAccountingPeriod

Opens the Lease accounting period from specific start to end periods.

Test Data

The leaseOpenAccountingPeriod function requires the following Test Data:

@param1, @param2 as start period, end period.

B.7.3 refreshPaymentProcessRequest

Refresh payment process request.

Test Data

The refreshPaymentProcessRequest function requires the following Test Data:

@param1, @param2, @param3, @param4, @param5, @param6 as refreshButtonName, requestSourceColName, requestSourceColValue, referenceSourceColName, referenceSourceColValue, StatusToBeVerified.

B.7.4 webImgVerfyCheckBox

Verifies the image label of a checkbox.

Test Data

The webImgVerfyCheckBox function requires the following Test Data:

@param1 as Image Label.

B.7.5 webSelectCheckBoxFromLOV

Select Checkbox in the table queried by the List of Values: Used specifically for selecting Planning Resources in 12.2.

Test Data

The webSelectCheckBoxFromLOV function requires the following Test Data:

@logical, @param1, @param2, @param3, @param4 as SearchByOption, SearchValue, ColName, ValueToSelect.

B.7.6 webVerfyMergTblValBasedOnLabel

Verifies a specific concurrent request output file.

Test Data

The webVerfyMergTblValBasedOnLabel function requires the following Test Data:

@param1 as Search Column Cell Value.

B.7.7 webVerfyTblValueBasedOnLabel

Verifies a specific concurrent request output file.

Test Data

The webVerfyTblValueBasedOnLabel function requires the following Test Data:

@param1 as Search Column Cell Value.

B.8 sCMLIB Function Library

The sCMLIB function library is used to develop component code and flows for Supply Chain Management applications.

B.8.1 disableInterCompanyRecord

Disable the intercompany records for the specified organizations and end dates.

Test Data

The disableInterCompanyRecord function requires the following Test Data:

@param1, @param2, @param3, @param4 as From org, end org, flow type, end date.

B.8.2 getDeliveryNumber

Gets the delivery number from the Ship Confirm Request string.

Test Data

The getDeliveryNumber function requires the following Test Data:

@param1 as Ship Confirm Request String.

B.8.3 getExpenditureGroup

Get the expenditure group.

Test Data

The getExpenditureGroup function requires the following Test Data:

@param1 as Enter expenditureGroupRowindex.

B.8.4 getLPNNameFromLog

Picks LPN number from Log file which is in 'Defined but Not used' status.

Test Data

The getLPNNameFromLog function does not require Test Data.

B.8.5 getLPNNumber

Select a specific serial number from the specified list.

Test Data

The getLPNNumber function requires the following Test Data:

@param1, @param2 as From LPN, LPN Index.

B.8.6 getShipConfirmReqIds

Capture request id's after Ship Confirm.

Test Data

The getShipConfirmReqIds function requires the following Test Data:

@param1, @param2, @param3 as Ship Confirm Request String, Before, After.

B.8.7 getTripStopReqId

Gets the Trip stop request number.

Test Data

The getTripStopReqId function requires the following Test Data:

@param1 as Ship Confirm Request String.

B.8.8 selectDayInMonth

Selects a day in a month.

Test Data

The selectDayInMonth function requires the following Test Data:

@param1 as Enter Day.

B.8.9 setTextInDualField

Sets the specified value to the Itemfield value in a Flex window.

Test Data

The setTextInDualField function requires the following Test Data:

@logical, @param1, @param2 as Enter label with commas, Enter testdata with commas.

B.8.10 unexpectedPopUp

Clicks the OK button on the popup window.

Test Data

The unexpectedPopUp function does not require Test Data.

B.8.11 verifyLabelContextXMLData

Verifies the XML output in the Label Content field in the Label request History Page.

Test Data

The verifyLabelContextXMLData function requires the following Test Data:

@logical, @param1, @param2 as Name attribute values of variable tag, Variable tag values.

B.9 tELNETLIB Function Library

The tELNETLIB function library is used to develop component code and flows for Telnet applications.

B.9.1 buttonPress

Presses Enter in a Telnet window.

Test Data

The buttonPress function does not require Test Data.

B.9.2 changeOrg

Changes the organization in a Telnet session.

Test Data

The changeOrg function requires the following Test Data:

@param1 as Org.

B.9.3 close

Closes a Telnet session.

Test Data

The close function does not require Test Data.

B.9.4 commit

Commits the transaction in a Telnet window.

Test Data

The commit function does not require Test Data.

B.9.5 commitAndVerify

Commits the transaction and verifies the status bar message in a Telnet window.

Test Data

The commitAndVerify function requires the following Test Data:

@param1 as Expected StatusBar Text.

B.9.6 commitExpandAndVerify

Commits the transaction and verifies the complete status bar message in a Telnet window.

Test Data

The commitExpandAndVerify function requires the following Test Data:

@param1 as Expanded Verify Message.

B.9.7 connect

Connects to a Telnet window.

Test Data

The connect function requires the following Test Data:

@param1, @param2 as Host, Port.

B.9.8 ctrl

Presses the specified key combined with the Ctrl key in a Telnet window.

Test Data

The ctrl function requires the following Test Data:

@param1 as Key.

B.9.9 ctrlAndEnter

Presses Ctrl and Enter the specified message on the Telnet window.

Test Data

The ctrlAndEnter function requires the following Test Data:

@param1 as Message.

B.9.10 esc

Presses the Escape key on a Telnet window.

Test Data

The esc function does not require Test Data.

B.9.11 expandVerifyStatusBarValue

Expands and verifies if the status bar value the specified message.

Test Data

The expandVerifyStatusBarValue function requires the following Test Data:

@param1 as Expanded Verify Message.

B.9.12 getCurrField

Gets the title of the current field.

Test Data

The getCurrField function does not require Test Data.

Gets the value from the current field in a Telnet window.

Test Data

The getCurrField function does not require Test Data.

B.9.13 getFieldValue

Gets the field value from a Telnet window.

Test Data

The getFieldValue function does not require Test Data.

B.9.14 getFullStatus

Gets the complete status bar message from a Telnet window.

Test Data

The getFullStatus function does not require Test Data.

B.9.15 getPreviousField

Gets the title of the previous field in a Telnet screen.

Test Data

The getPreviousField function does not require Test Data.

Gets the value from the previous field from a Telnet window.

Test Data

The getPreviousField function does not require Test Data.

B.9.16 getScreen

Gets a Telnet screen's content.

Test Data

The getScreen function does not require Test Data.

B.9.17 getStatusBar

Gets the status bar message of a Telnet window.

Test Data

The getStatusBar function does not require Test Data.

B.9.18 gotoTopField

Goes to the top field in a Telnet window.

Test Data

The gotoTopField function does not require Test Data.

B.9.19 initializeTelnetService

Initializes the Telnet session before login.

Test Data

The initializeTelnetService function does not require Test Data.

B.9.20 login

Logs in to Telnet session with the specified username and password.

Test Data

The login function requires the following Test Data:

@param1, @param2 as Username, Password.

B.9.21 logout

Logs out from a Telnet session.

Test Data

The logout function does not require Test Data.

B.9.22 navigateByName

Navigates to the specified path.

Test Data

The navigateByName function requires the following Test Data:

@param1 as Resp.

B.9.23 selectOption

Selects an option from the list of options present in a Telnet window.

Test Data

The selectOption function requires the following Test Data:

@param1 as Option.

B.9.24 set

Enters the specified value in the current field in a Telnet window.

Test Data

The set function requires the following Test Data:

@param1 as Value.

B.9.25 setScreenBufferTime

Sets the screen buffer time in milliseconds.

Test Data

The setScreenBufferTime function requires the following Test Data:

@param1 as Milliseconds.

B.9.26 skipDown

skips down one field in a Telnet window.

Test Data

The skipDown function does not require Test Data.

B.9.27 skipUp

Skips up one field in a Telnet window.

Test Data

The skipUp function does not require Test Data.

B.9.28 sleep

Waits for the default wait time.

Test Data

The sleep function does not require Test Data.

B.9.29 switchResp

Switches to the specified responsibility in a Telnet window.

Test Data

The switchResp function requires the following Test Data:

@param1 as Resp.

B.9.30 verifyFieldValue

Verifies the specified value for the field in a Telnet window.

Test Data

The verifyFieldValue function requires the following Test Data:

@param1, @param2 as Verify field label name, Expected value to verify.

B.9.31 verifyStatusBarValue

Verifies the status bar message in a Telnet window.

Test Data

The verifyStatusBarValue function requires the following Test Data:

@param1 as Expected status bar message.

B.9.32 waitForActionComplete

Waits for the current action to be completed in a Telnet window.

Test Data

The waitForActionComplete function does not require Test Data.

B.9.33 waitForScreen

Waits for the screen until the text appears on the Telnet window.

Test Data

The waitForScreen function requires the following Test Data:

@param1, @param2 as Text, Time.

B.9.34 waitForTitle

Waits for the Telnet window until the title appears.

Test Data

The waitForTitle function requires the following Test Data:

@param1, @param2 as Title, Time.

B.10 wEBTABLELIB Function Library

The wEBTABLELIB function library is used to develop component code and flows for Web Table applications.

B.10.1 CLICKIMAGE

Clicks the Web table image object.

Test Data

The CLICKIMAGE function does not require Test Data.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W