Oracle® Functional Testing

Flow Builder User's Guide

Release 13.3.0.1

E48651-12

May 2018

ORACLE®

Oracle Functional Testing Flow Builder User's Guide Release 13.3.0.1

E48651-12

Copyright © 2013, 2018, Oracle and/or its affiliates. All rights reserved.

Primary Author: Rick Santos

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

Contributor: Drupad Panchal , Srinivas Potnuru

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

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

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

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

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

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

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

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

Contents

4 Defining Components Sets

5 Defining Flows

6 Using Notifications

7 Using History

8 Using Reports

9 Administering Oracle Flow Builder

A Keyword Reference

B Function Library Reference

Index

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 that have purchased support 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 Functional Testing Flow Builder Release Notes*
- *Oracle Application Testing Suite Installation Guide*
- *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 12.2*
- *Oracle Load Testing Load Testing User's 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.

Convention	Meaning
`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.

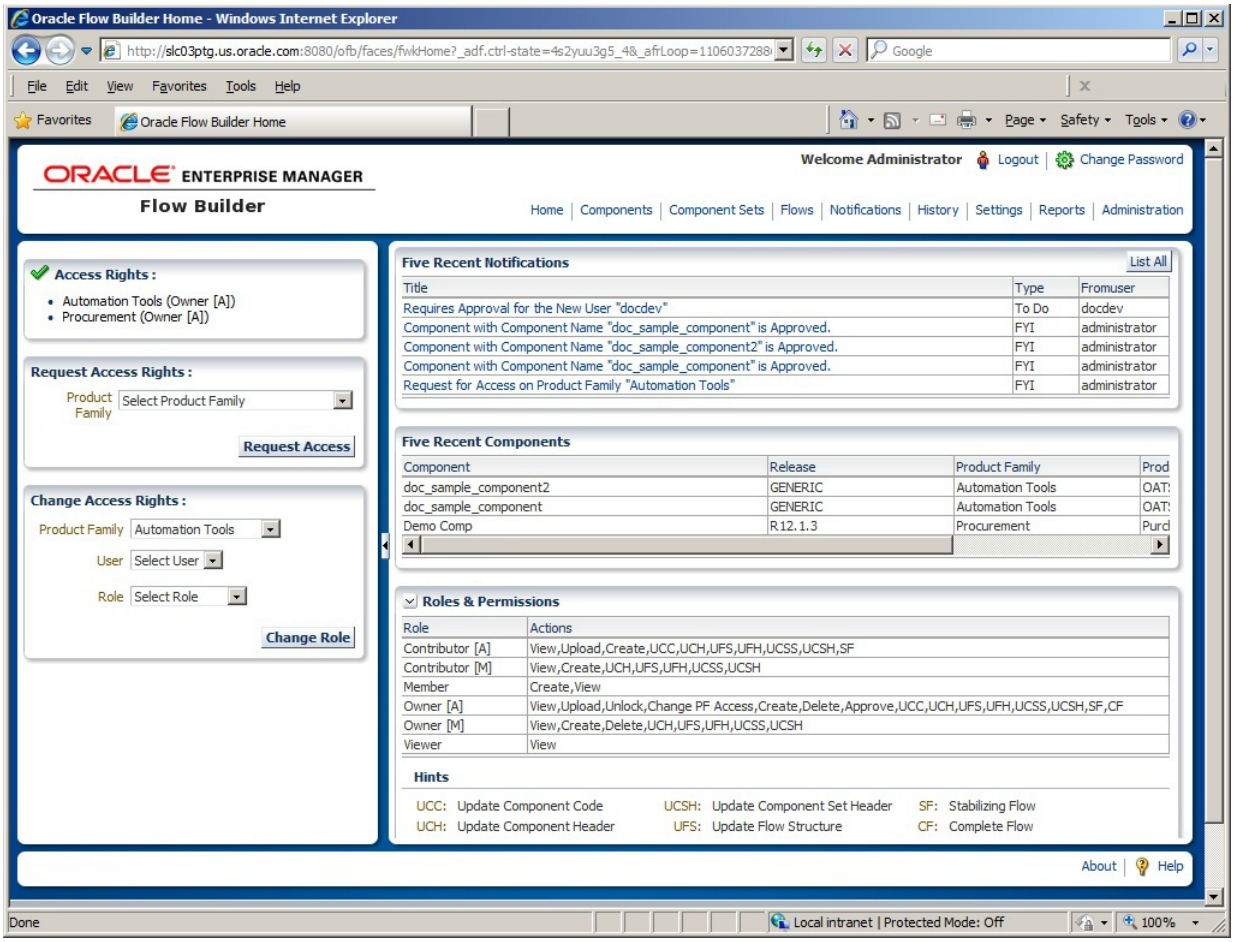
This chapter introduces the Oracle Flow Builder application and provides an overview of the features available. It contains the following sections:

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) or Oracle Database 12*c* Enterprise Edition (v. 12.1.0.2.0).
- Oracle Enterprise Linux version 5 or higher.
- Third party libraries dependence: `glibc.i686,unzip, lsof`. If you do not have these, please install with `yum`:

```
yum -y install glibc.i686  unzip lsof
```

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`

- Or, is running Oracle Database 12*c* Enterprise Edition (v. 12.1.0.2.0), which can be accessed from:

  `http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html`

- 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 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* documentation at the following location:

`http://docs.oracle.com/en/database/database.html`

2.4.1 Installing Oracle Database Enterprise Edition

The Oracle Flow Builder application requires an installed and functioning Oracle Database 11*g* or 12*c* Enterprise Edition. If necessary, download and install the database prior to installing the Oracle Flow Builder application.

To download and install Oracle Database:

1. Download the database zip files and extract the files:

2. Run `./database/runInstaller`.
3. Clear the **I wish to receive security updates** checkbox and click **Next**.
4. Click **Skip software updates**.
5. Select **Create and configure a database** and click **Next**.
6. Select **Desktop Class** and click **Next**.
7. Specify the Install Configuration as follows:
   - Oracle base: `/scratch/<username>/oracle/oee/app`
   - Software location: `/scratch/<username>/oracle/oee/app /product/<version>/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: (any password you wish to use as the default database administrator password). Passwords must contains at least 8 characters, cannot start with a number, and must contain at least one number. This password should be the same password used when installing Oracle Flow Builder.
8. Click **Yes** on the "Admin password entered does not conform to the Oracle standards" dialog box.
9. On Prerequisite Checks, if there are errors, click **Ignore All** and click **Next**.
10. Click **Yes** on the "You have chosen to ignore some of the prerequisites" dialog box.
11. On the Summary page, confirm your entries and click **Install**.
12. 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
    ```

13. 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/<version>/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: <database password>
$> 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: (any password you wish to use as the default OFB administrator account password). Passwords must contains at least 8 characters, cannot start with a number, and must contain at least one number. This password should be the same password used when installing Oracle Database.
- 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` (or the user name and password of a user with privileges to create the OFB user).
- Enter database admin Password: This password should be the same password used when installing Oracle Database and the password for db user OFB and OFB UI user "administrator".
- When prompted `OFB user already exists in database? [1:No;2:Yes]: (Enter for No)`, select 1 for a new database and the installation will create the OFB user in the database, then create the tables for OFB. If you select 2 for yes, you will need to create the OFB user manually.

  Log in to SqlPlus as a user with create user privileges and run the following SQL commands:

  ```
  CREATE USER OFB IDENTIFIED BY <password> DEFAULT TABLESPACE users;
  GRANT CREATE SESSION, UNLIMITED TABLESPACE, CREATE TABLE, CREATE CLUSTER,
  CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER, CREATE TYPE, CREATE
  OPERATOR, CREATE INDEXTYPE, CREATE ANY VIEW to OFB;
  ```

  `<password>` is the password for db user OFB.

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 2.4.5, "Oracle Flow Builder Server Maintenance" section for additional server options.

2.4.4 Upgrading from Earlier Versions

If you are upgrading from an earlier version of Oracle Flow Builder, there are additional steps required to update the database schema and reference the updated schema using the newer version. This section explains the steps required.

2.4.4.1 Checking Versions

The Oracle Flow Builder 12.5.0.2 (and higher) version includes features for checking versions:

- Use `./setup.sh version` to display the version of the download zip file. The `setup.sh` file is located in the location where the download zip file is unzipped.
- Use `./control.sh version` to display which version of Oracle Flow Builder is installed. The `control.sh` file is located in `<ofb-install-dir>/bin`.

2.4.4.2 Upgrading Oracle Flow Builder 12.5.0.2 to 12.5.0.3

The Oracle Flow Builder upgrade scripts support user upgrades only to the next higher version. For example, 12.5.0.1 to 12.5.0.2 and then 12.5.0.2 to 12.5.0.3. You cannot update directly from 12.5.0.1 to 12.5.0.3. This section explains the steps to upgrade a version 12.5.0.2 instance of Oracle Flow Builder to a version 12.5.0.3 instance of Oracle Flow Builder. You will need the Oracle Flow Builder 12.5.0.3 download zip file to obtain the upgrade scripts.

To upgrade Oracle Flow Builder 12.5.0.2 to 12.5.0.3:

1. Make a backup of the Oracle Flow Builder 12.5.0.2 database schema.
2. Open sqlplus and connect to the Oracle Flow Builder 12.5.0.2 database schema.
3. Locate the SQL update scripts for Oracle Flow Builder version 12.5.0.3 in the following location of the 12.5.0.3 download zip file:

   ```
   /common/install/static/data/scripts/en/12.5.0.3
   ```

4. Run the sql update files using the following commands:

   ```
   SQL> @ofb.sql;
   SQL> @ofb_template_data.sql;
   SQL> @ofb_template_data_ebs.sql;   (do this step if your product is ebs)
   ```

5. When the update script finishes, exit sqlplus.
6. Install Oracle Flow Builder 12.5.0.3 on a different host and database.
7. Change the Oracle Flow Builder data source to point to the updated 12.5.0.2 Oracle Flow Builder schema by performing the following steps:
   a. Login to the Weblogic console: `http://<server_name>:9190/console` (ofb/<install_password>) where Oracle Flow Builder 12.5.0.3 is running.
   b. Click the Data Sources link on the Home page.
   c. Click the KWDT Data Source link.
   d. Click the Connection Pool tab.
   e. Replace the URL field with the database connection pointing to the updated Oracle Flow Builder 12.5.0.2 database instance.
   f. Logout from the Weblogic console.
   g. Restart the Weblogic console running on the Oracle Flow Builder 12.5.0.3 instance using the following commands:

      ```
      <ofb-install-dir>/scripts/control.sh stop
      <ofb-install-dir>/scripts/control.sh start
      ```

8. Copy configuration files as needed from the Oracle Flow Builder 12.5.0.2 instance to the Oracle Flow Builder 12.5.0.3 instance, as follows:
   a. `<ofb-install-dir>/ofb/config/mail.properties`
   b. `<ofb-install-dir>/ofb/config/repository.properties` if using repositories other than the default "OATS" repository.

c. <*ofb-install-dir*>/ofb/data/function-libraries if using custom libraries.

9. Login to the Oracle Flow Builder 12.5.0.3 instance, which should point to the upgraded Oracle Flow Builder 12.5.0.2 database schema.

2.4.4.3 Upgrading Oracle Flow Builder 12.5.0.3 to 13.1.0.1

The Oracle Flow Builder upgrade scripts support user upgrades only to the next higher version. For example, 12.5.0.2 to 12.5.0.3 and then 12.5.0.3 to 13.1.0.1. You cannot update directly from 12.5.0.2 to 13.1.0.1. This section explains the steps to upgrade a version 12.5.0.3 instance of Oracle Flow Builder to a version 13.1.0.1 instance of Oracle Flow Builder. If you have an Oracle Flow Builder 12.5.0.2 instance, see Section 2.4.4.2, "Upgrading Oracle Flow Builder 12.5.0.2 to 12.5.0.3" before completing these steps. You will need the Oracle Flow Builder 13.1.0.1 download zip file to obtain the upgrade scripts.

To upgrade Oracle Flow Builder 12.5.0.3 to 13.1.0.1:

1. Make a backup of the Oracle Flow Builder 12.5.0.3 database schema.
2. Open sqlplus and connect to the Oracle Flow Builder 12.5.0.3 database schema.
3. Locate the SQL update scripts for Oracle Flow Builder version 13.1.0.1 in the following location of the 13.1.0.1 download zip file:

   ```
   /common/install/static/data/scripts/en/13.1.0.1
   ```

4. Run the sql update files using the following commands:

   ```
   SQL> @ofb.sql;
   SQL> @ofb_template_data.sql;
   SQL> @ofb_template_data_ebs.sql;   (do this step if your product is ebs)
   ```

5. When the update script finishes, exit sqlplus.
6. Install Oracle Flow Builder 13.1.0.1 on a different host and database.
7. Change the Oracle Flow Builder data source to point to the updated 12.5.0.3 Oracle Flow Builder schema by performing the following steps:
   a. Login to the Weblogic console: http://<*server_name*>:9190/console (ofb/<install_password>) where Oracle Flow Builder 13.1.0.1 is running.
   b. Click the Data Sources link on the Home page.
   c. Click the KWDT Data Source link.
   d. Click the Connection Pool tab.
   e. Replace the URL field with the database connection pointing to the updated Oracle Flow Builder 12.5.0.3 database instance.
   f. Logout from the Weblogic console.
   g. Restart the Weblogic console running on the Oracle Flow Builder 13.1.0.1 instance using the following commands:

      ```
      <ofb-install-dir>/scripts/control.sh stop
      <ofb-install-dir>/scripts/control.sh start
      ```

8. Copy configuration files as needed from the Oracle Flow Builder 12.5.0.3 instance to the Oracle Flow Builder 13.1.0.1 instance, as follows:
   a. <*ofb-install-dir*>/ofb/config/mail.properties

b. *<ofb-install-dir>*/ofb/config/repository.properties if using repositories other than the default "OATS" repository.

c. *<ofb-install-dir>*/ofb/data/function-libraries if using custom libraries.

9. Login to the Oracle Flow Builder 13.1.0.1 instance, which should point to the upgraded Oracle Flow Builder 12.5.0.3 database schema.

2.4.4.4 Upgrading Oracle Flow Builder 13.1.0.1 to 13.2.0.1

The Oracle Flow Builder upgrade scripts support user upgrades only to the next higher version. For example, 12.5.0.3 to 13.1.0.1 and then 13.1.0.1 to 13.2.0.1. You cannot update directly from 12.5.0.3 to 13.2.0.1. This section explains the steps to upgrade a version 13.1.0.1 instance of Oracle Flow Builder to a version 13.2.0.1 instance of Oracle Flow Builder. If you have an Oracle Flow Builder 12.5.0.3 instance, see Section 2.4.4.3, "Upgrading Oracle Flow Builder 12.5.0.3 to 13.1.0.1" before completing these steps. You will need the Oracle Flow Builder 13.2.0.1 download zip file to obtain the upgrade scripts.

To upgrade Oracle Flow Builder 13.1.0.1 to 13.2.0.1:

1. Make a backup of the Oracle Flow Builder 13.1.0.1 database schema.
2. Open sqlplus and connect to the Oracle Flow Builder 13.1.0.1 database schema.
3. Locate the SQL update scripts for Oracle Flow Builder version 13.2.0.1 in the following location of the 12.5.0.3 download zip file:

   ```
   /common/install/static/data/scripts/en/13.2.0.1
   ```

4. Run the sql update files using the following commands:

   ```
   SQL> @ofb.sql;
   SQL> @ofb_template_data.sql;
   SQL> @ofb_template_data_ebs.sql;   (do this step if your product is ebs)
   ```

5. When the update script finishes, exit sqlplus.
6. Install Oracle Flow Builder 13.2.0.1 on a different host and database.
7. Change the Oracle Flow Builder data source to point to the updated 13.1.0.1 Oracle Flow Builder schema by performing the following steps:

   a. Login to the Weblogic console: http://*<server_name>*:9190/console (ofb/<install_password>) where Oracle Flow Builder 13.2.0.1 is running.

   b. Click the Data Sources link on the Home page.

   c. Click the KWDT Data Source link.

   d. Click the Connection Pool tab.

   e. Replace the URL field with the database connection pointing to the updated Oracle Flow Builder 13.1.0.1 database instance.

   f. Logout from the Weblogic console.

   g. Restart the Weblogic console running on the Oracle Flow Builder 13.2.0.1 instance using the following commands:

      ```
      <ofb-install-dir>/scripts/control.sh stop
      <ofb-install-dir>/scripts/control.sh start
      ```

8. Copy configuration files as needed from the Oracle Flow Builder 13.1.0.1 instance to the Oracle Flow Builder 13.2.0.1 instance, as follows:
   a. <*ofb-install-dir*>/ofb/config/mail.properties
   b. <*ofb-install-dir*>/ofb/config/repository.properties if using repositories other than the default "OATS" repository.
   c. <*ofb-install-dir*>/ofb/data/function-libraries if using custom libraries.
9. Login to the Oracle Flow Builder 13.2.0.1 instance, which should point to the upgraded Oracle Flow Builder 13.1.0.1 database schema.

2.4.5 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*>/bin/control.sh start. For example:

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

- Stopping the Oracle Flow Builder Application Server

  Run <*ofb-install-dir*>/bin/control.sh stop. For example:

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

- Getting the Oracle Flow Builder Application Server running status

  Run <*ofb-install-dir*>/bin/control.sh status. For example:

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

- Getting the Oracle Flow Builder Application version number

  Run <*ofb-install-dir*>/bin/control.sh version. For example:

  ```
  > /scratch/username/oracle/ofb/bin/control.sh version
  ```

2.4.6 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*>/bin/deinstall.sh. For example:

   ```
   > /scratch/username/oracle/ofb/bin/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

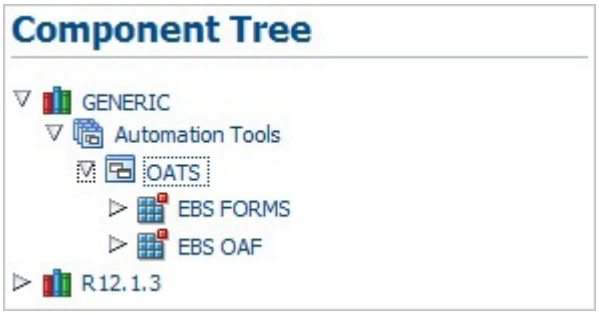
An administrator defines the Release, Product Family, Product, and Feature structure of the Component Tree. Users add Components to the tree and define the keywords and parameters that will be used to specify Component Sets and Flows that generate the OpenScript scripts used to test the application under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



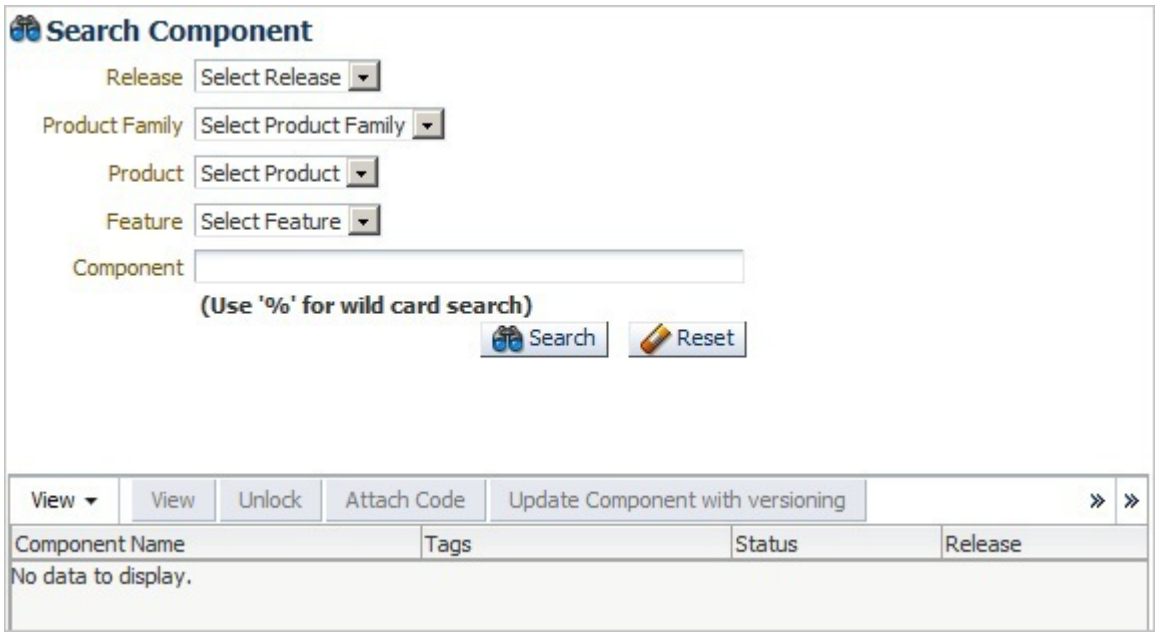
Release and application specific components are listed below the Release(s) listed in the Components Tree. Expanding the tree shows the list of products and features in the release tree:

Figure 3–4 Release Component Tree

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

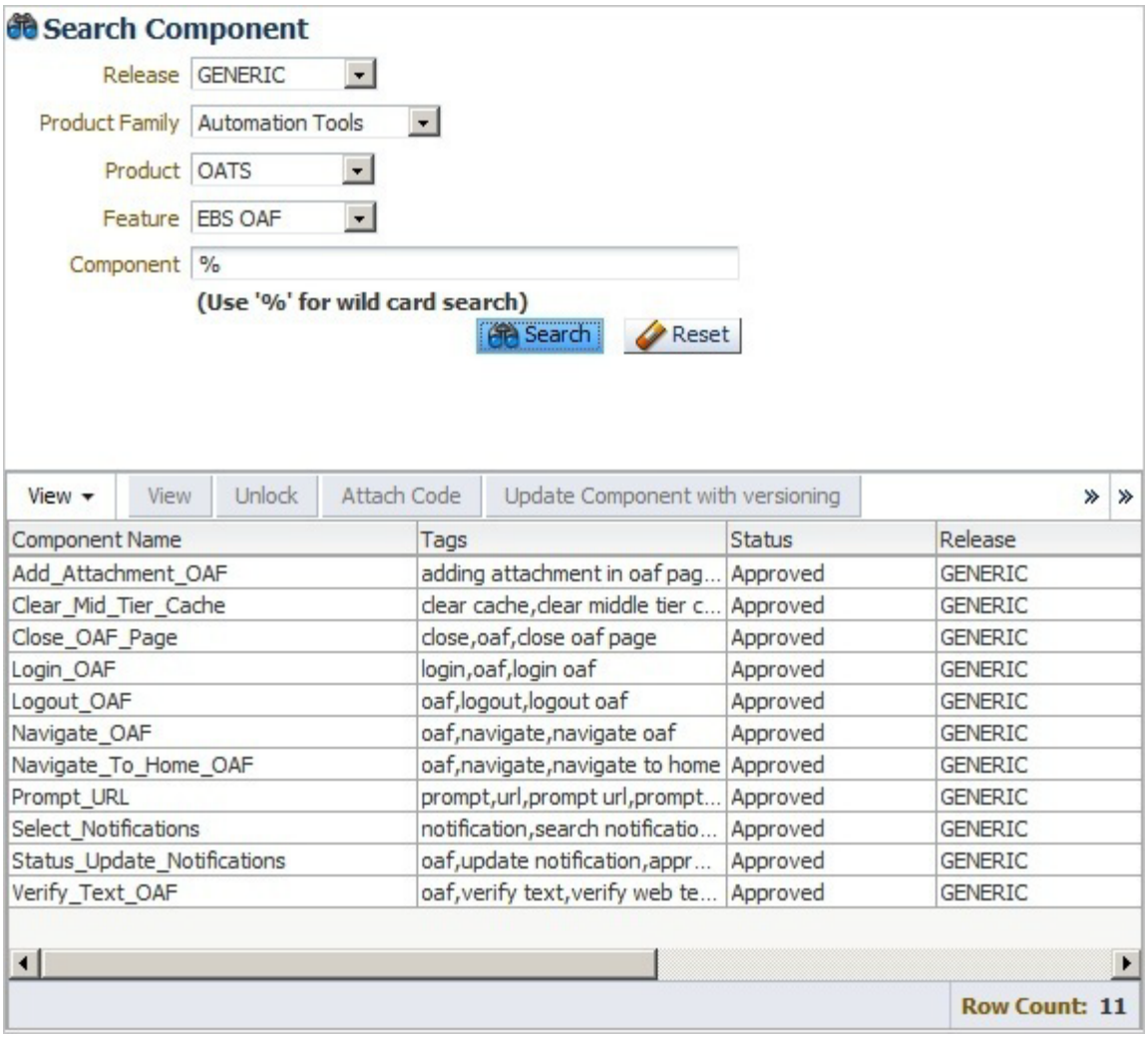
The Search pane of the Components page lets you search for specific components:

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

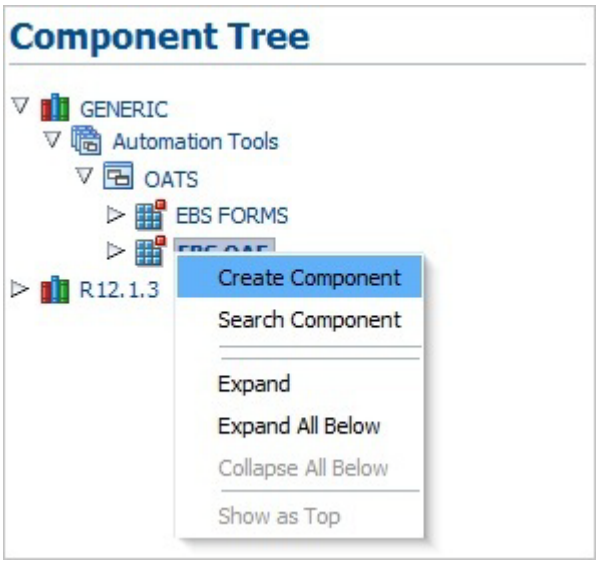
This section explains how to use the Component Tree to add an individual component to a product feature.

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.

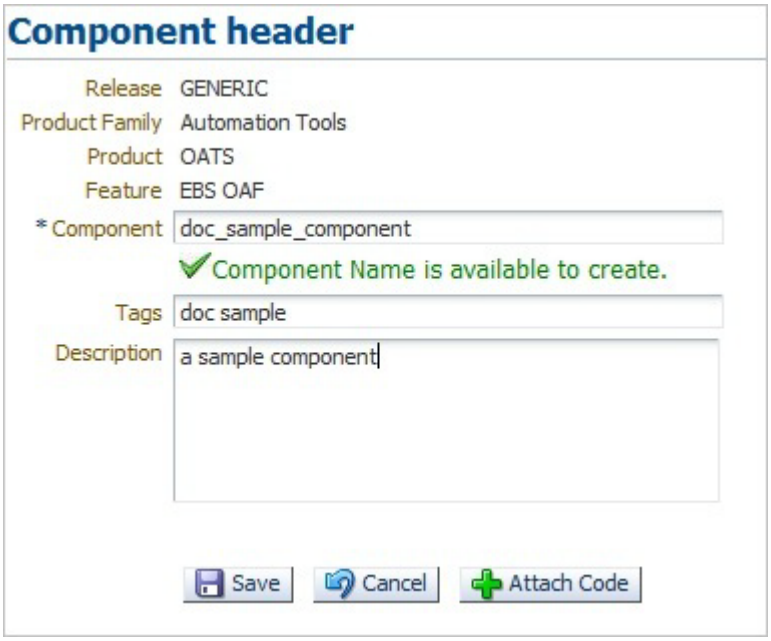
2. Right-click the Feature name and select **Create Component** from the shortcut menu.

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



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

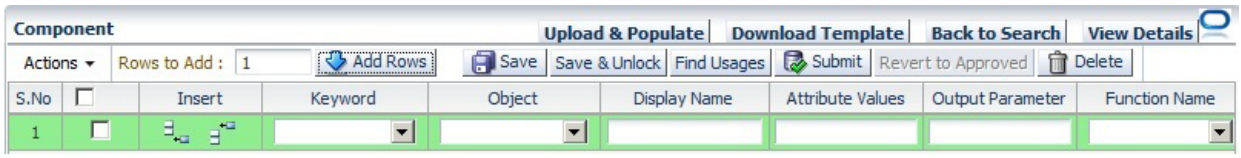
Figure 3–8 Component Header Pane



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

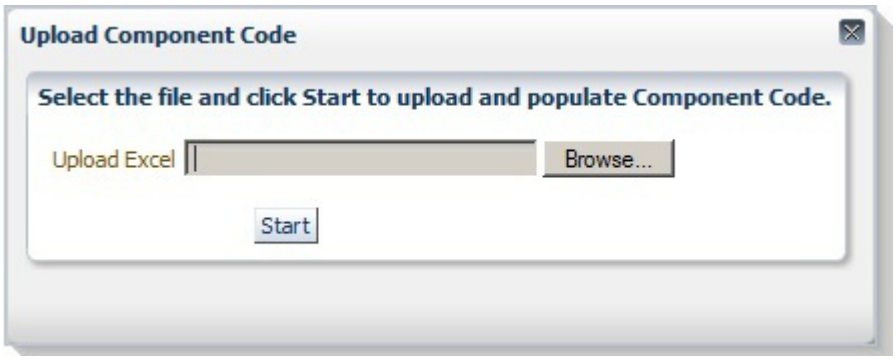
This section explains how to upload component code specified in an Excel spreadsheet. The Oracle Flow Builder includes a template Excel spreadsheet file that can be used to specify keywords, objects, and attributes to use to define a component. The Excel spreadsheet can be uploaded to the component code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

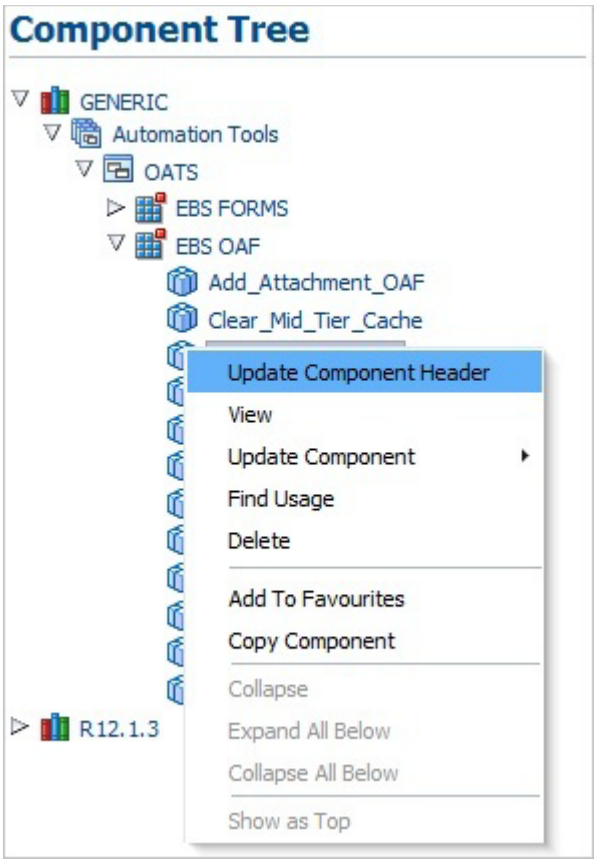
3.3 Updating Components

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

To update an existing component:

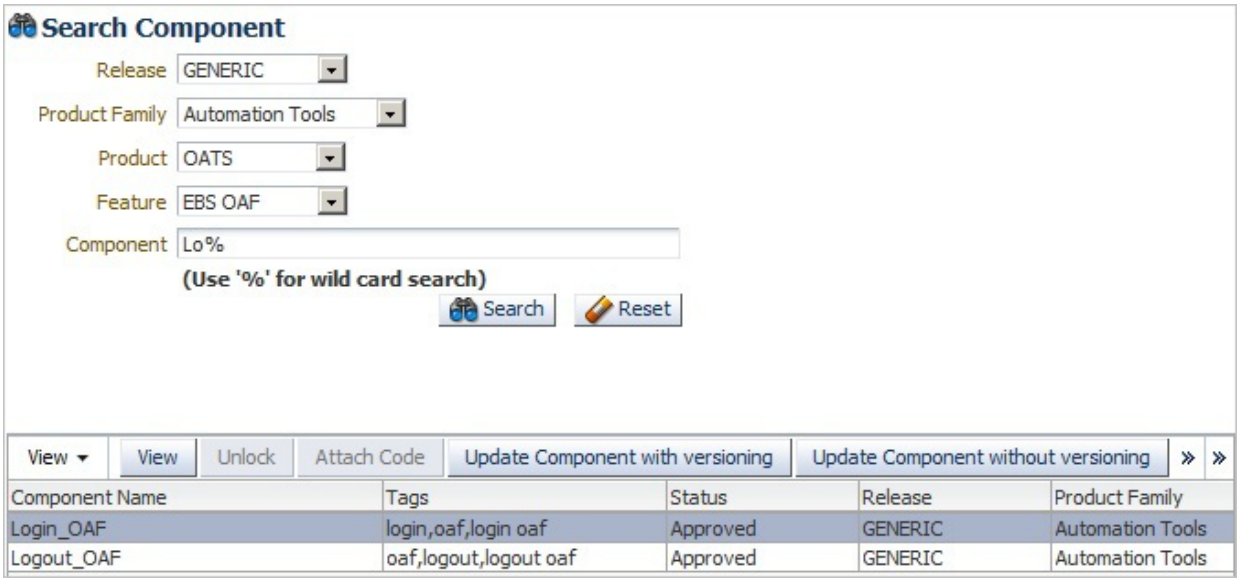
1. Select the component to update using the Component Tree or the Search Component options.
   - If using the Component Tree, right-click the component to open the shortcut menu:

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.

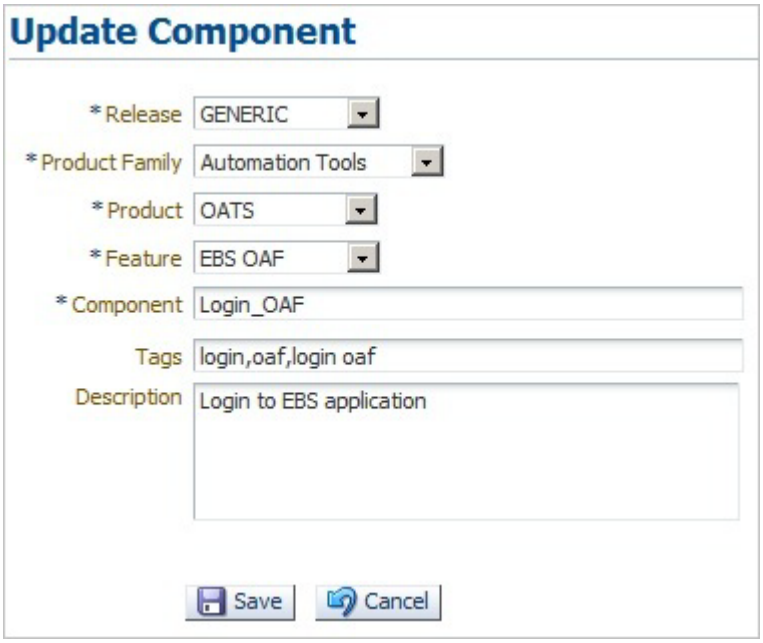
2. Select the update option.

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



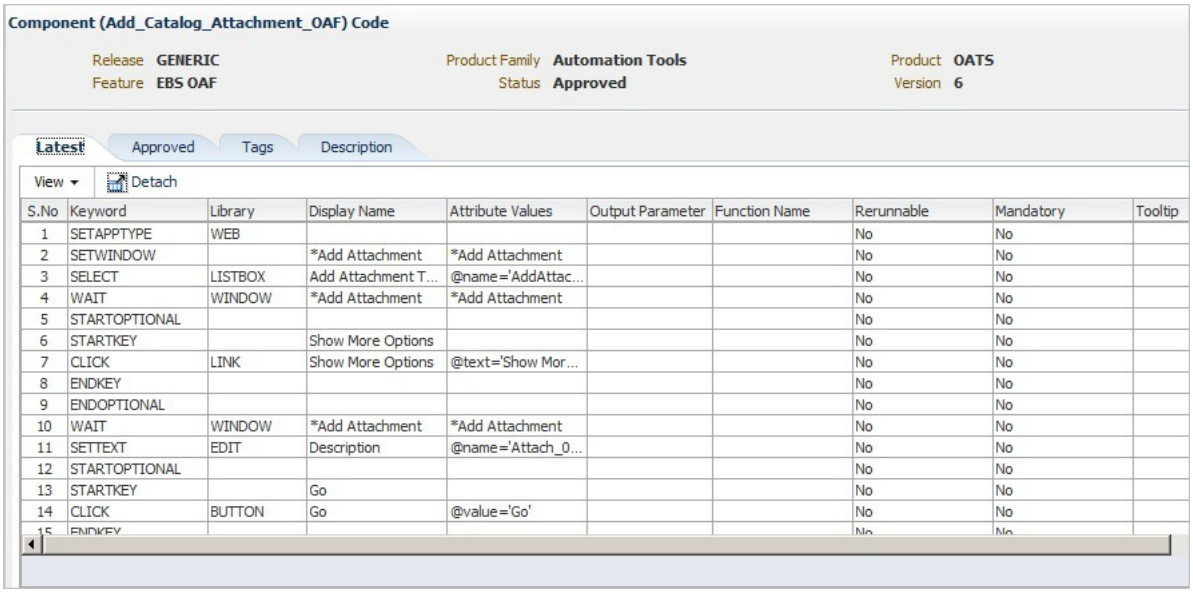
3. Update the component header information as needed and click **Save**.

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	

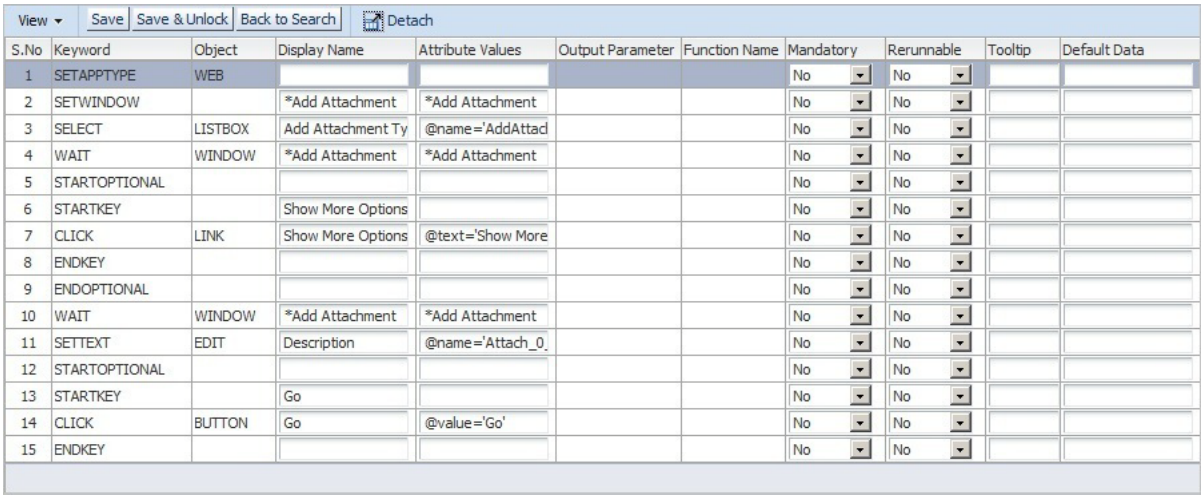
3. Use the **View** menu options to re-order columns and select specific columns to view.
4. Click **Detach** to open the component code in a larger view.
5. Click the **[X]** button to close the component code view when finished.

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



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

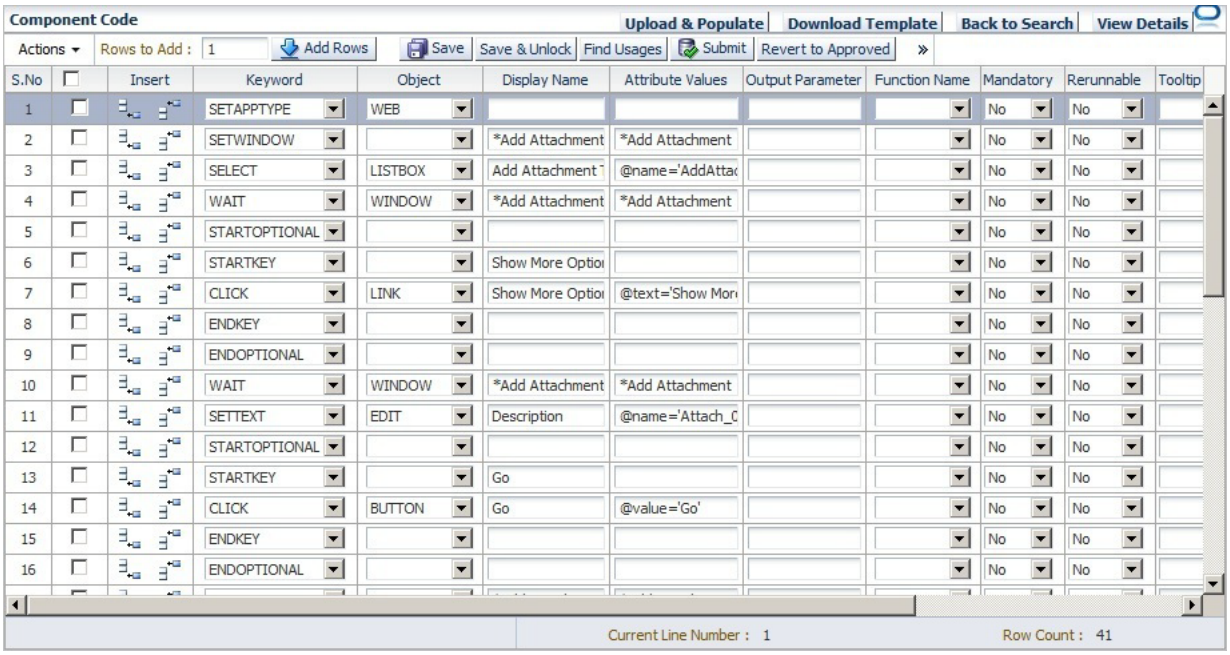
3. Click **Save** or **Save & Unlock** after necessary modifications are made. Click **Back to Search** to exit without saving changes.

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



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**.

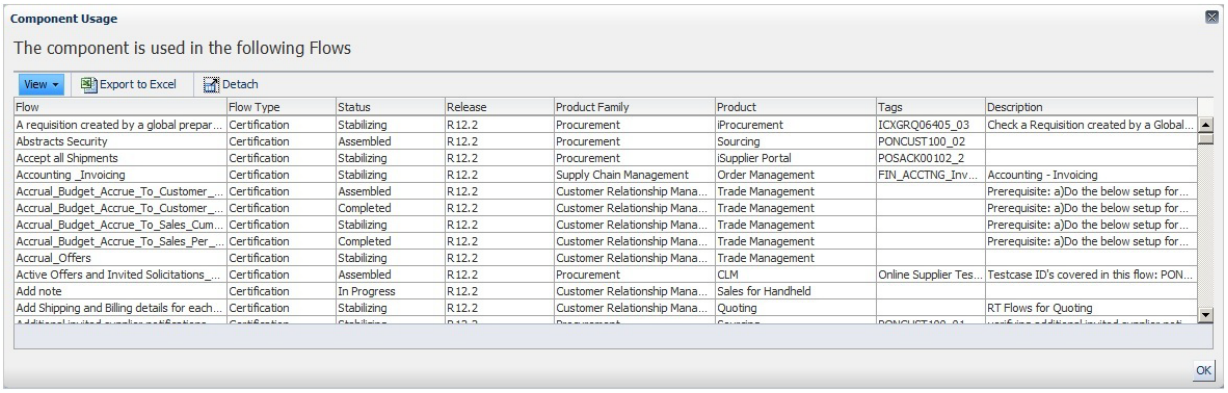
3. Click **Save** or **Save & Unlock** after necessary modifications are made. Click **Back to Search** to exit without saving changes.

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 For Web Components

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 for Web components.

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".

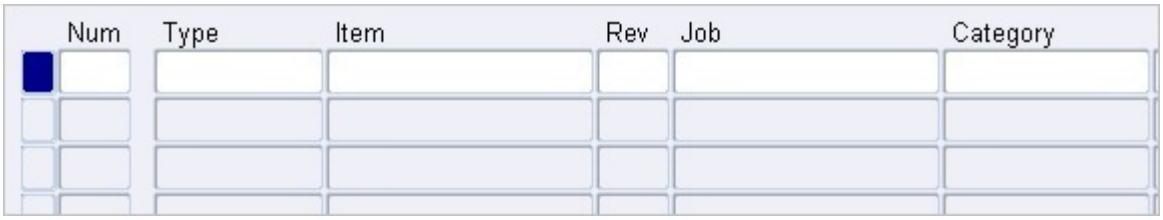
Step 5 through 10 define the STARTOPTIONAL and ENDOPTIONAL Keyword set for the Next button click.

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.

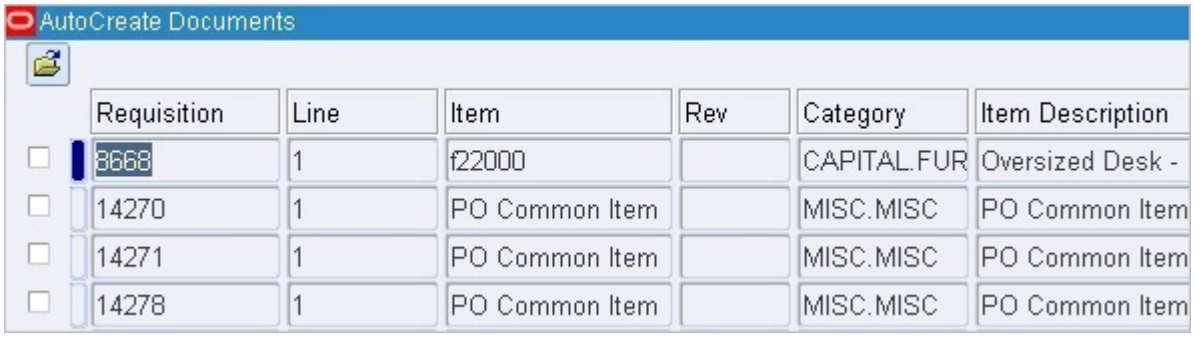
Step 5 and 6 set the text in the fields.

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

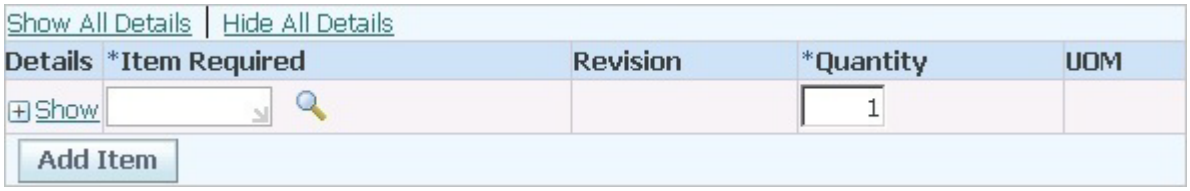
Step 1 clicks the Include Function tab.

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

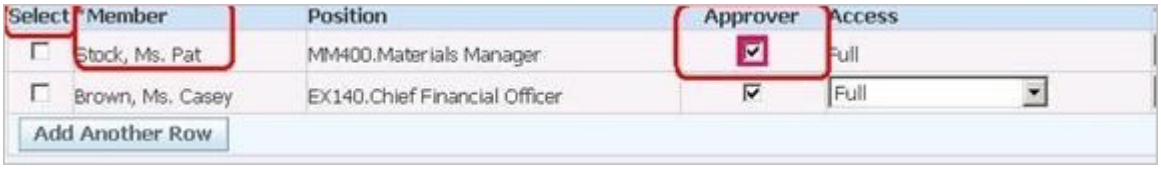
SEARCHCOLUMN 2nd column name as display name

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

3.4.2 Component Code for PLSQL or Open Interface Components

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

3.4.2.1 Keywords for PLSQL Components

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

Setting Application Type

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

Table 3–13 Keywords for Setting Application Type

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	PLSQL_OI		

Step 1 sets the application type as a PLSQL or Open interface.

Select Component

Select component is used to retrieve the data from the database using simple query. Statements in component code must be defined between STARTQUERY and ENDQUERY keywords as specified in the following scenario:

Table 3–14 Keywords for Select Statements

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	PLSQL_OI		
2	STARTQUERY			
3	SELECT	TABLE	Table Name	

Table 3–14 (Cont.) Keywords for Select Statements

Step #	Keyword	Object Type	Display Name	Attribute Values
4	STARTROWITERATOR		Enter Number of Columns	
5	SELECT	COLUMN	Column Details	
6	ENDROWITERATOR			
7	STARTROWITERATOR		Enter Number of Where Conditions	
8	SET	CONDITION	Where Condition	
9	ENDROWITERATOR			
10	ENDQUERY			

Step 1 sets the application type as a PLSQL or Open Interface.

Step 2 defines the start of the statement keywords STARTQUERY.

Step 3 specifies a SELECT action which requires to provide Database table name

Step 4 defines the start of the statement STARTROWITERATOR, This is used to specify number of column to retrieve from database.

Step 5 specifies the column name to retrieved from the database.

Step 6 defines the end of the statement ENDROWITERATOR.

Step 7 defines the start of the statement STARTROWITERATOR for specifying number of conditions for database query.

Step 8 specifies the conditions to be provided for simple database query.

Step 9 defines the end of the statement ENDROWITERATOR.

Step 10 defines the end of the statement ENDQUERY.

PLSQL Statements

Refers to **stored procedure** driven interfaces, where a call to stored procedure is made to perform an action within an Oracle Module, and the data from the external application is passed through the stored procedure's parameters.

Every PL/SQL constant, variable, parameter, and function return value has a **data type** that determines its storage format.

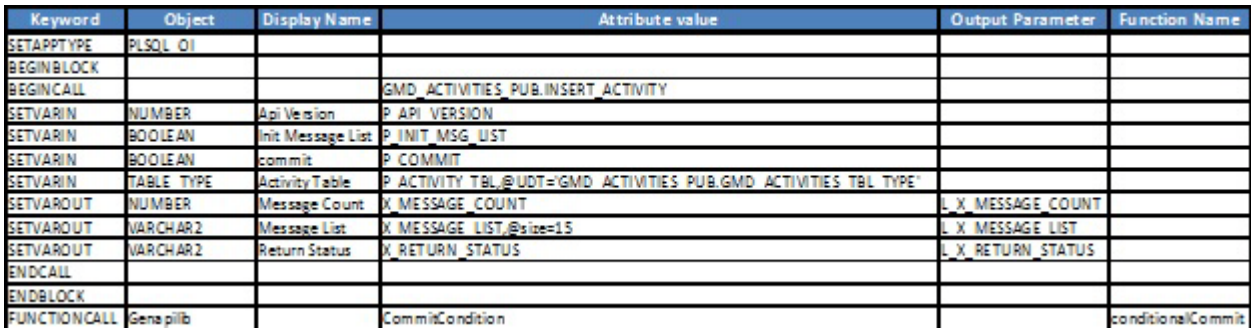
Procedure Component

PLSQL procedure component is a data structure that can hold all the parameters of a PLSQL procedure. These parameters can be of different data types such as Standard data types (Number, Boolean, Varchar etc.), record type, object type, varray type, and table type.

- These parameters can be of INPUT, OUTPUT and INOUT and those are defined with the keywords SETVARIN, SETVAROUT,SETVARINOUT.
- Variable should be defined for OUTPUT and INOUTparameters. These variables can be considered for further usage/verification.
- Procedure component should be defined between BEGINBLOCK and ENDBLOCK, BEGINCALL and ENDCALL keywords.

- If a procedure is only defined between BEGINCALL and ENDCALL, then that procedure is called as part of another main procedure execution.

Figure 3–24 Example PLSQL Procedure Statements



Keyword	Object	Display Name	Attribute value	Output Parameter	Function Name
SETAPPTYPE	PLSQL_OI				
BEGINBLOCK					
BEGINCALL			GMD_ACTIVITIES_PUB.INSERT_ACTIVITY		
SETVARIN	NUMBER	Api Version	P_API_VERSION		
SETVARIN	BOOLEAN	Init Message List	P_INIT_MSG_LIST		
SETVARIN	BOOLEAN	commit	P_COMMIT		
SETVARIN	TABLE_TYPE	Activity Table	P_ACTIVITY_TBL,@UDT='GMD_ACTIVITIES_PUB.GMD_ACTIVITIES_TBL_TYPE'		
SETVAROUT	NUMBER	Message Count	X_MESSAGE_COUNT	L_X_MESSAGE_COUNT	
SETVAROUT	VARCHAR2	Message List	X_MESSAGE_LIST,@size=15	L_X_MESSAGE_LIST	
SETVAROUT	VARCHAR2	Return Status	X_RETURN_STATUS	L_X_RETURN_STATUS	
ENDCALL					
ENDBLOCK					
FUNCTIONCALL	Genapilib		CommitCondition		conditionalCommit

Table 3–15 Keywords for PLSQL Procedure Statements

Step	Keyword	Object	Display Name	Attribute Values	Output Parameter	Function Name
1	SETAPPTYPE	PLSQL_OI				
2	BEGINBLOCK			@udt=GMD_ ACTIVITIES _PUB.GMD_ ACTIVITIES _REC_TYPE	L_ACTIVITY_ REC	
3	BEGINCALL			GMD_ ACTIVITIES_ PUB.INSERT_ ACTIVITY		
4	SETVARIN	NUMBER	Api Version	P_API_VERSION		
5	SETVARIN	BOOLEAN	Init Message List	P_INIT_MSG_LIST		
6	SETVARIN	VARCHAR	Commit	P_COMMIT		
7	SETVAROUT	VARCHAR2	Return Status	X_RETURN_ STATUS	L_X_RETURN_ STATUS	
8	SETVAROUT	NUMBER	Message Count	X_MESSAGE_ COUNT	L_X_MSG_ COUNT	
9	SETVARINOUT	VARCHAR2	Resp Appl Name	M_RESP_APPL_ NAME	L_M_RESP_ APPL_NAME	
10	SETVARINOUT	NUMBER	Resp Appl Id	M_RESP_APPL_ID	L_M_ RESP_ APPL_ID	
11	SETVARIN	RECORD_ TYPE	Service Request rec	P_SERVICE_ REQUEST_ REC,@udt='CS_ SERVICEREQUEST _PUB.SERVICE_ REQUEST_REC_ TYPE'		

Table 3–15 (Cont.) Keywords for PLSQL Procedure Statements

Step	Keyword	Object	Display Name	Attribute Values	Output Parameter	Function Name
12	SETVAROUT	RECORDTYPE	Sr Create Out Rec	X_SR_CREATE_OUT_REC,@udt='CS_SERVICEREQUEST_PUB.SR_CREATE_OUT_REC_TYPE'	L_X_SR_CREATE_OUT_REC	
13	SETVARINOUT	RECORDTYPE	Service Request rec	M_SR_CREATE_OUT_REC,@udt='CS_SERVICEREQUEST_PUB.SR_CREATE_OUT_REC_TYPE'	L_M_SR_CREATE_OUT_REC	
14	SETVARIN	TABLE_TYPE	Contacts	P_CONTACTS,@udt='CS_SERVICEREQUEST_PUB.CONTACTS_TABLE'		
15	SETVAROUT	TABLE_TYPE	Notes	P_NOTES,@udt='CS_SERVICEREQUEST_PUB.NOTES_TABLE'	L_X_SR_NOTE_OUT_TABLE	
16	SETVARINOUT	TABLE_TYPE	Contacts	M_CONTACTS,@udt='CS_SERVICEREQUEST_PUB.CONTACTS_TABLE'	L_M_SR_NOTE_OUT_TABLE	
17	SETVARIN	OBJECT_TYPE	Assign	P_ASSIGN,@udt='CS_SERVICEREQUEST_PUB.ASSIGN_OBJECT'		
18	SETVAROUT	OBJECT_TYPE	Assign	X_ASSIGN,@udt='CS_SERVICEREQUEST_PUB.ASSIGN_OBJECT'	L_X_SR_ASSIGN_OBJECT	
19	SETVARINOUT	OBJECT_TYPE	Notes	M_NOTE,@udt='CS_SERVICEREQUEST_PUB.NOTES_OBJECT'	L_M_SR_NOTE_OBJECT	
20	SETVARIN	VARRAY_TYPE	User Id	P_USER,@udt=WF_PARAMETER_LIST_T		

Table 3–15 (Cont.) Keywords for PLSQL Procedure Statements

Step	Keyword	Object	Display Name	Attribute Values	Output Parameter	Function Name
21	ENDCALL					
22	FUNCTIONCALL	gENAPILIB	Error Handling			GetErrorHandlingBlock
23	ENDBLOCK					

Step 1 sets the application type as a PLSQL application.

Step 2 defines the start of the statement BEGINBLOCK.

Step 3 defines the start of the statement BEGINCALL.

Step 4 thru step 6 defines INPUT standard data type parameters.

Step 7 thru step 8 defines OUTPUT standard data type parameters.

Step 9 thru step 10 defines INOUT standard data type parameters.

Step 11 defines INPUT record type parameter.

Step 12 defines OUTPUT record type parameter.

Step 13 defines INOUT record type parameter.

Step 14 defines INPUT table type parameter.

Step 15 defines OUTPUT table type parameter.

Step 16 defines INOUT table type parameter.

Step 17 defines INPUT object type parameter.

Step 18 defines OUTPUT object type parameter.

Step 19 defines INOUT object type parameter.

Step 20 defines INPUT varray type parameter.

Step 21 defines the end of the statement BEGINCALL using keyword ENDCALL.

Step 22 defines the GENAPILIB function to handle the message in block.

Step 23 defines the end of the statement BEGINBLOCK using keyword ENDBLOCK.

Record Type Component

A record component is to represent a PLSQL Record Data type to hold single database record. The fields of a component can be of any PLSQL data type.

- Record Component is defined between STARTRECORDTYPE and ENDRECORDTYPE.
- All the fields of the record are to be defined with SETVAR keyword and corresponding data type as OFB component object.
- A variable should be defined for the Record Component.

Figure 3–25 Example PLSQL Record Statements

S.No	Keyword	Object	Display Name	Attribute Values	Output Parameter
1	SETAPPTYPE	PLSQL_OI			
2	STARTRECORDTYPE		Activities Rec Type	@udt=GMD_ACTIVITIES_PUB.ACTIVITIES_REC_TYPE	L_ACTIVITIES_REC
3	SETVAR	VARCHAR2	Activity	ACTIVITY	
4	SETVAR	VARCHAR2	Cost Analysis Code	COST_ANALYSIS_CODE	
5	SETVAR	NUMBER	Delete Mark	DELETE_MARK	
6	SETVAR	NUMBER	Text Code	TEXT_CODE	
7	SETVAR	NUMBER	Trans Cnt	TRANS_CNT	
8	SETVAR	VARCHAR2	Activity Desc	ACTIVITY_DESC	
9	ENDRECORDTYPE				

Table 3–16 Keywords for PLSQL Record Statements

Step #	Keyword	Object	Display Name	Attribute Values	Output Parameter
1	SETAPPTYPE	PLSQL_OI			
2	STARTRECORDTYPE	PLSQL_OI		@udt=GMD_ ACTIVITIES _PUB.GMD_ ACTIVITIES _REC_TYPE	L_ACTIVITY_REC
3	SETVAR	NUMBER	Transaction	TRANS_CNT	
4	SETVAR	VARCHAR2	Delete mark	DELETE_MARK	
5	SETVAR	VARCHAR2	Text Code	TEXT_CODE	
6	SETVAR	VARCHAR2	Activity	ACTIVITY	
7	SETVAR	VARCHAR2	Cost Analysis Code	COST_ANALYSIS_ CODE	
8	SETVAR	VARCHAR2	Activity Description	ACTIVITY_DESC	
9	ENDRECORDTYPE				

Step 1 sets the application type as a PLSQL application.

Step 2 defines the start of the statement STARTRECORDTYPE.

- This keyword will hold the record name along with the package in the attribute field.
- Variable should be provided in the output parameter field.

Step 3 thru step 8 defines variable of standard data type.

Step 9 defines the end of the statement STARTRECORDTYPE using keyword ENDRECORDTYPE.

Object Type Component

An Object component is to represent PLSQL Object Data type. The fields of an Object component can be of any PLSQL data type.

- Object Component is defined between STARTOBJECTTYPE and ENDOBJECTTYPE.
- All the fields of the object are to be defined with SETVAR keyword and corresponding data type as OFB component object.
- A variable should be defined for the Object Component.

Table 3–17 Keywords for PLSQL Object Statements

Step #	Keyword	Object	Display Name	Attribute Values	Output Parameter
1	SETAPPTYPE	PLSQL_OI			
2	STARTOBJECTTYPE		Wf Event	@udt=WF_EVENT_T	L_WF_EVENT_T
3	SETVAR	VARCHAR2	Error Stock	ERROR_STOCK	
4	SETVAR	DATE	Received Date	RECEIVED_DATE	
5	SETVAR	NUMBER	Priority	PRIORITY	
6	ENDOBJECTTYPE				

Step 1 sets the application type as a PLSQL application.

Step 2 defines the start of the statement STARTOBJECTTYPE.

- This keyword will hold the object name along with the package, if exists in the attribute field.
- Variable should be provided in the output parameter field.

Step 3 thru 5 defines variable of standard data type.

Step 6 defines the end of the statement STARTOBJECTTYPE using keyword ENDOBJECTTYPE.

Note:PLSQL OBJECT Component supports only Standard type parameters.

Table of Standard Types Component

A table component is to represent a PLSQL Table Data type to hold a set of database standard type rows. The fields of a table component is of PLSQL Standard data type.

- Table Component is defined between STARTTABLETYPE and ENDTABLETYPE.
- Number of rows to be passed to the table is to be defined between STARTROWITERATOR and ENDROWITERATOR.
- All the fields of the table are to be defined with SETVAR keyword and corresponding Standard data type as OFB component object.
- A variable should be defined for the Table of Standard Type Component.

Table 3–18 Keywords for PLSQL Table of Standard Types Statements

Step #	Keyword	Object	Display Name	Attribute Values	Output Parameter
1	SETAPPTYPE	PLSQL_OI			
2	STARTTABLETYPE		Activity Table	@udt=GMD_ACTIVITIES _PUB.GMD_ACTIVITIES _TBL_TYPE	L_ACTIVITY_TABLE
3	STARTROWITERATOR		No of Rows		
4	SETVAR	VARCHAR2	Activity Id	ACTIVITY_ID	
5	ENDROWITERATOR				
6	ENDTABLETYPE				

Step 1 sets the application type as a PLSQL application.

Step 2 defines the start of the statement STARTTABLETTYPE.

- This keyword will hold the table name along with the package in the attribute field.
- Variable should be provided in the output parameter field.

Step 3 defines the start of the statement STARTROWITERATOR. This keyword is used to specify number of Rows to passed to table.

Step 4 defines standard data type field.

Step 5 defines the end of the statement STARTROWITERATOR using keyword ENDROWITERATOR.

Step 6 defines the end of the statement STARTTABLETYPE using keyword ENDTABLETYPE.

Table of Record Type Component

A table of Records component is to represent a PLSQL Table Data type to hold a set of database record rows. The fields of a table component are of PLSQL Record data type.

- Table Component is defined between STARTTABLETTYPE and ENDTABLETYPE.
- Number of rows to be passed to the table is to be defined between STARTROWITERATOR and ENDROWITERATOR.
- All the fields of the table are to be defined with SETVAR keyword and RECORD Type as OFB component object.
- A variable should be defined for the Table of Records Component

Figure 3–26 Example PLSQL Table of Records Statements

Keyword	Object	Display Name	Attribute Values	Output Parameter
SETAPPTYPE	PLSQL_OI			
STARTTABLETYPE		Gmd Activities Tbl Type	@udt=GMD_ACTIVITIES_PUB.GMD_ACTIVITIES_TBL_TYPE	L_GMD_ACTIVITIES_TBL
STARTROWITERATOR		Number of Rows		
SETVAR	RECORD_TYPE	Activities Rec Type	@udt=GMD_ACTIVITIES_PUB.ACTIVITIES_REC_TYPE	
ENDROWITERATOR				
ENDTABLETYPE				

Table 3–19 Keywords for PLSQL Table of Records Statements

Step #	Keyword	Object	Display Name	Attribute Values	Output Parameter
1	SETAPPTYPE	PLSQL_OI			
2	STARTTABLETYPE		Activity Table	@udt=GMD_ ACTIVITIES _PUB.GMD_ ACTIVITIES _TBL_TYPE	L_ACTIVITY_ TABLE
3	STARTROWITERATOR		Number of Records		

Table 3–19 (Cont.) Keywords for PLSQL Table of Records Statements

Step #	Keyword	Object	Display Name	Attribute Values	Output Parameter
4	SETVAR	RECORD_ TYPE	Activity Record	@udt=GMD_ ACTIVITIES _PUB.GMD_ ACTIVITIES _REC_TYPE	
5	ENDROWITERATOR				
5	ENDTABLETYPE				

Step 1 sets the application type as a PLSQL application.

Step 2 defines the start of the statement STARTTABLETTYPE.

- This keyword will hold the table name along with the package in the attribute field.
- Variable should be provided in the output parameter field.

Step 3 defines the start of the statement STARTROWITERATOR. This keyword is used to specify number of Rows to be passed to the table.

Step 4 defines the record data type field.

Step 5 defines the end of the statement STARTROWITERATOR using keyword ENDROWITERATOR.

Step 6 defines the end of the statement STARTTABLETYPE using keyword ENDTABLETYPE.

Table of Object Component

A table of Objects component is to represent a PLSQL Table Data type to hold a set of database object rows. The fields of a table component are of PLSQL Object data type.

- Table Component is defined between STARTTABLETTYPE and ENDTABLETYPE.
- Number of rows to be passed to the table is to be defined between STARTROWITERATOR and ENDROWITERATOR.
- All the fields of the table are to be defined with SETVAR keyword and OJECT_ TYPE as OFB component object.
- A variable should be defined for the Table of Object Component.

Table 3–20 Keywords for PLSQL Table of Object Statements

Step #	Keyword	Object	Display Name	Attribute Values	Output Parameter
1	SETAPPTYPE	PLSQL_OI			
2	STARTTABLETYPE		Activity Table	@udt=GMD_ ACTIVITIES _PUB.GMD_ ACTIVITIES _TBL_TYPE	L_ACTIVITY_ TABLE
3	STARTROWITERATOR		Number of Objects		

Table 3–20 (Cont.) Keywords for PLSQL Table of Object Statements

Step #	Keyword	Object	Display Name	Attribute Values	Output Parameter
4	SETVAR	OBJECT_ TYPE	Wf Event	@udt=WF_EVENT_ T	
5	ENDROWITERATOR				
6	ENDTABLETYPE				

Step 1 sets the application type as a PLSQL application.

Step 2 defines the start of the statement STARTTABLETTYPE.

- This keyword will hold the table name along with the package in the attribute field.
- Variable should be provided in the output parameter field.

Step 3 defines the start of the statement STARTROWITERATOR. This keyword is used to specify number of Rows to be passed to the table.

Step 4 defines var object data type field.

Step 5 defines the end of the statement STARTROWITERATOR using keyword ENDROWITERATOR.

Step 6 defines the end of the statement STARTTABLETYPE using keyword ENDTABLETYPE.

VARRAY Type Component

VArray component is to represent PLSQL VArray Data type. The fields of a VArray component can be of any PLSQL standard data type.

- VArray Component is defined between STARTVARRAYTYPE and ENDVARRAYTYPE.
- All the fields of the object are to be defined SETVAR keyword and corresponding data type as OFB component object.
- A variable should be defined for the VArray Component.

Table 3–21 Keywords for PLSQL VArray Statements

Step #	Keyword	Object	Display Name	Attribute Values	Output Parameter
1	SETAPPTYPE	PLSQL_OI			
2	STARTVARRAYTYPE		Wms Epc TagId Type	@udt=WMS_EPC_ TAGID_ TYPE,@index=false, @size=100	L_WMS_EPC_ TAGID
3	STARTROWITERATOR				
4	SETVAR	VARCHAR2	Varray of VARCHAR2	@size=260	
5	ENDROWITERATOR				
6	ENDVARRAYTYPE				

Step 1 sets the application type as a PLSQL application.

Step 2 defines the start of the statement STARTVARRAYTYPE.

- This keyword will hold the VArray name in the attribute field.
- Variable should be provided in the output parameter field.

Step 3 defines the start of the statement STARTROWITERATOR. This keyword is used to specify number of Rows to be passed to the table.

Step 4 defines Standard data type field.

Step 5 defines the end of the statement STARTROWITERATOR using keyword ENDROWITERATOR.

Step 6 defines the end of the statement STARTVARRAYTYPE using keyword ENDVARRAYTYPE.

Note: Oracle Flow Builder PLSQL does not support Nested Records (Record within a Record) and Table within a Record.

3.4.2.2 Keywords for Open Interface Components

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

Setting Application Type

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

Table 3–22 Keywords for Setting Application Type

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	PLSQL_OI		

Step 1 sets the application type as a PLSQL or Open interface.

Open Interface Statements

This interface is used as an integration point between external application and the Oracle module. These components are used for loading data and data validations.

Insert Component

Insert Component is used to insert data into database table.Statements in component code must be defined between STARTQUERY and ENDQUERY in the following scenario:

Table 3–23 Keywords for Insert Statements

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	PLSQL_OI		
2	STARTQUERY			
3	INSERT	TABLE	PO Lines Interface Table	PO_LINES_ INTERFACE
4	SET	COLUMN	Interface Line ID	INTERFACE_LINE_ID
5	SET	COLUMN	Line Number	LINE_NUM
6	ENDQUERY			

Step 1 sets the application type as a PLSQL or Open Interface application.

Step 2 defines the start of the statement STARTQUERY.

Step 3 defines the start of the statement INSERT. This keyword is used to specify database table name.

Step 4 specifies a SET action which specifies database column name.

Step 5 specifies a SET action which specifies database column name.

Step 6 defines the end of the statement STARTQUERY using keyword ENDQUERY.

Open Interface Concurrent Program Component

Concurrent program component is to represent the concurrent program which needs to be executed after inserting data into temporary tables. The concurrent program is a PLSQL procedure call. Statements in component code must be defined between STARTBLOCK and ENDBLOCK in the following scenario:

Table 3–24 Keywords for Concurrent Program Component Statements

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	PLSQL_OI		
2	STARTBLOCK			
3	BEGINCALL		WIP Mass Load	FND_ REQUEST.SUBMIT_ REQUEST,@returntype =NUMBER
4	SETVARIN	VARCHAR2	Application Short Name	application
5	SETVARIN	DATE	Start Date	Start_date
6	SETVARIN	BOOLEAN	Sub Request	Sub_request
7	SETVARIN	VARCHAR2	Print Report	Print_report
8	ENDCALL			
9	ENDBLOCK			

Step 1 sets the application type as a PLSQL or Open Interface.

Step 2 defines the start of the statement STARTBLOCK.

Step 3 defines the start of the statement BEGINCALL.

Steps 4 thru 7 specify a SETVARIN action which are all the parameters required to execute concurrent program.

Step 8 defines the end of the statement BEGINCALL using keyword ENDCALL.

Step 9 defines the end of the statement STARTBLOCK using keyword ENDBLOCK.

3.4.2.3 Keywords for Webservice Components

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

Setting Application Type

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

Table 3–25 Keywords for Setting Application Type

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	WS		

Step 1 sets the application type as Webservice.

Webservice Statements

XML based protocols and standards used for exchanging data between applicants.

Webservice Structure Component

Webservice payload structure statements in component code must be defined between WS-STARTSTRUCTURE and WS-ENDSTRUCTURE keywords as specified in the following scenario:

Table 3–26 Keywords for Webservice Structure Component

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	WS		
2	WS-STARTSTRUCTURE			
3	WS-SETWEBSERVICENAME		WSDL URL Static Part	https://{{HOST}}:{{PORT}}/webservices/AppsWSProvider/oracle/apps/rrs/site/service/SiteService?wsdl
4	WS-NODENAMEWITHATTRIBUTE		Envelop	soapenv:Envelope,@xmlns:ser="http://xmlns.oracle.com/apps/fnd/ServiceBean",@xmlns:ser1="http://xmlns.oracle.com/apps/rrs/site/service",@xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
5	WS-OPERATIONNAME		Operation Name	SiteService_CreateSite
6	WS-PROCESSWSREQUEST			
7	WS-ENDSTRUCTURE			

Step 1 sets the application type as a Webservice.

Step 2 defines the start of the Webservice structure statement WS-STARTSTRUCTURE.

Step 3 specifies the Webservice WSDL URL Static Part.

Step 4 specifies the attributes of the payload root node.

Step 5 specifies the operation name for the xml payload.

Step 6 specifies to process the xml payload request

Step 7 defines the end of the Webservice structure statement WS-STARTSTRUCTURE using keyword WS-ENDSTRUCTURE.

Payload Request Node Definition Component

All the nodes in Webservice payload, having elements which require input data, is defined as a component.

Table 3–27 Keywords for Payload Request Node Definition Component

Step #	Keyword	Object Type	Display Name	Attribute Values
1	SETAPPTYPE	WS		
2	WS-PARENT	NODE		soapenv:Body
3	WS-NODENAME		WIP Mass Load	ser1:SiteService_ CreateSite/site
4	WS-SETXMLELEMENT		SiteNumber	SiteNumber
5	WS-SETXMLELEMENT		SiteName	SiteName
5	WS-SETXMLELEMENT		SiteType	SiteType
5	WS-SETXMLELEMENT		SiteStatus	SiteStatus

Step 1 sets the application type as a Webservice.

Step 2 specifies the Parent node.

Step 3 specifies the node name of the payload.

Steps 4 thru 7 defines the xml elements of a particular node.

Webservice Response Definition Component

All the nodes in Webservice response, having elements with data is defined as a component.

Table 3–28 Keywords for Webservice Response Definition Component

Step #	Keyword	Object	Display Name	Attribute Values	Output Parameter
1	SETAPPTYPE	WS			
2	WS-PARENT	NODE		@outXpath=soapenv:Envelope/soapenv:Body/ser:SiteService_CreateSite_Response/Message	SS_CST_Message
3	WS-SETXMLELEMENT		Code	Code	
4	WS-SETXMLELEMENT		Type	Type	
5	WS-SETXMLELEMENT		Text	Text	
6	WS-SETXMLELEMENT		DataSourceName	DataSourceName	

Step 1 sets the application type as a Webservice.

Step 2 defines the node path from root to current node.

Steps 3 thru 6 specifies the elements of a particular node in Webservice response.

4

Defining Components Sets

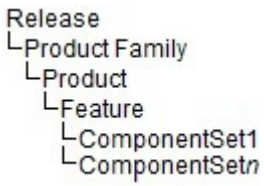
This chapter explains how to add and update components sets in the Component Set Tree. This chapter contains the following sections:

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



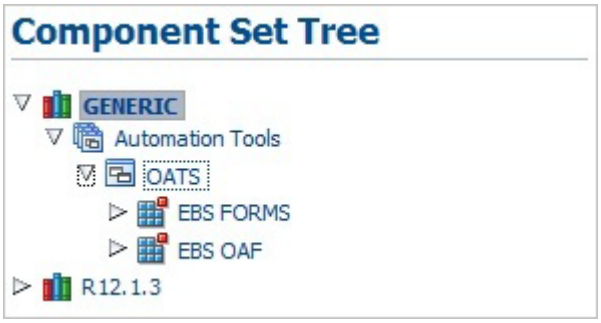
A Oracle Flow Builder administrator defines the Release, Product Family, Product, and Feature structure of the Component Set Tree. Oracle Flow Builder users add Components to the tree and define the keywords and parameters that will be used to specify Component Sets and Flows that generate the OpenScript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

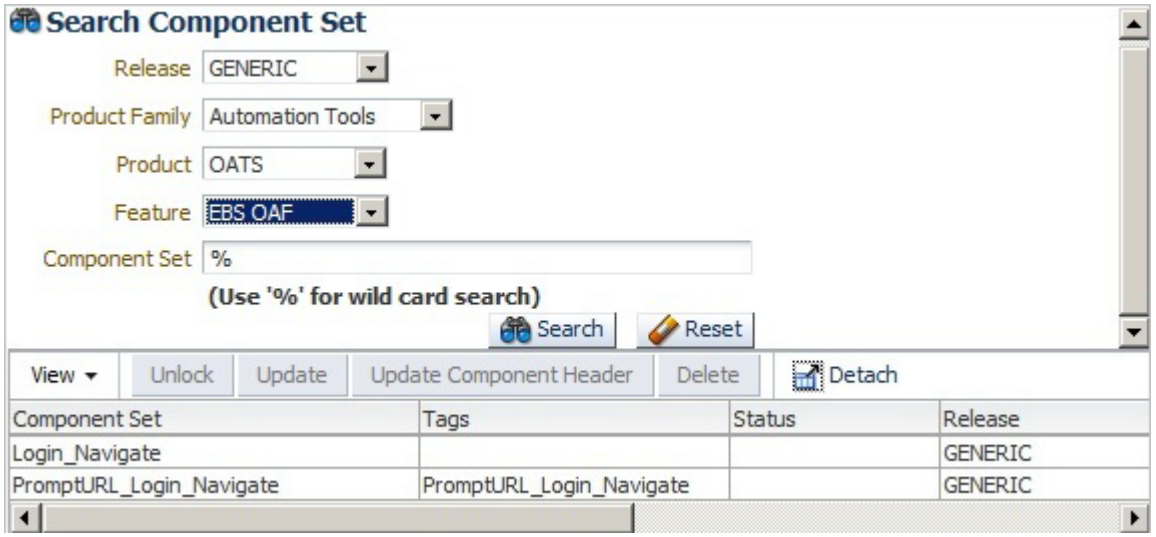
Note that the list of available products and features may vary based upon your specific installation.

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



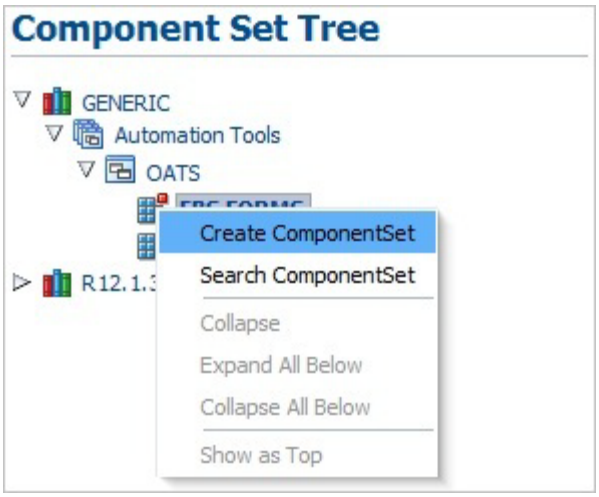
4.2 Adding Component Sets

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:

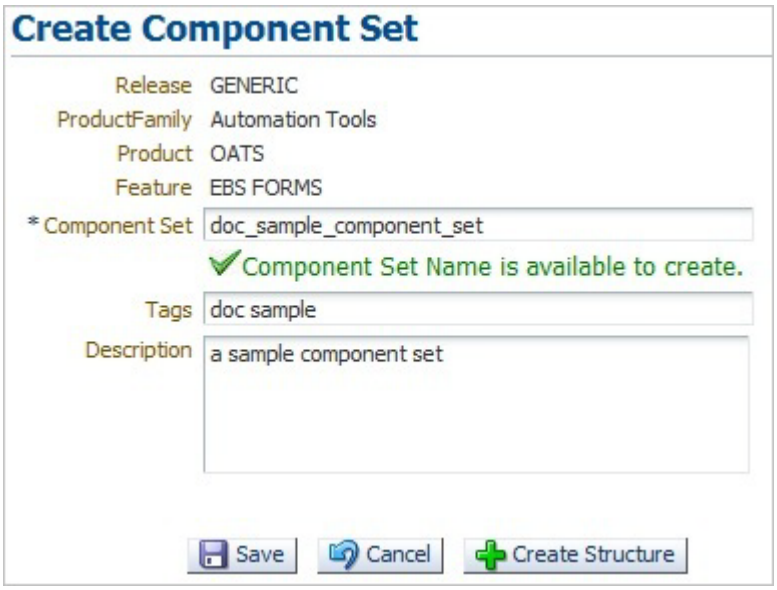
1. Expand the Component Set Tree to the product feature where you want to add the component set.
2. Right-click the Feature name and select **Create Component Set** from the shortcut menu.

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.

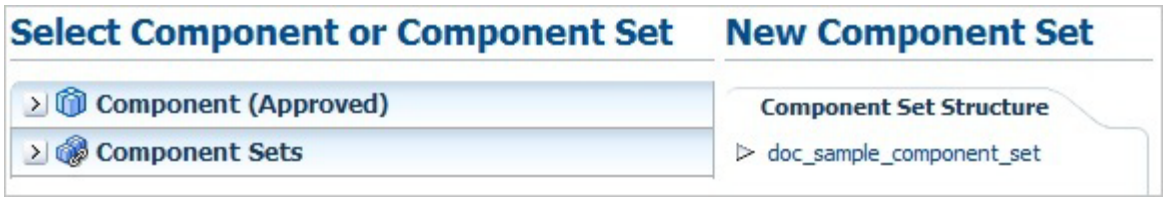
Figure 4–8 Create Component Set Pane



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.

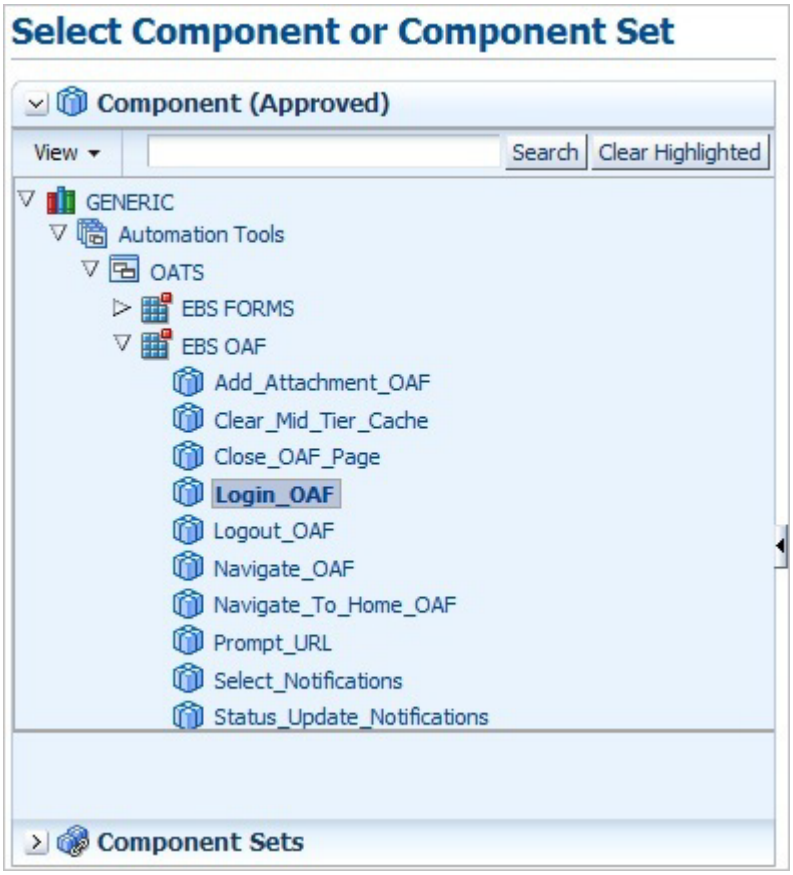
4. Click **Create Structure**. The Select Component or Components Set and the New Component Set pane opens for specifying the components or component sets to add to the new component set.

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

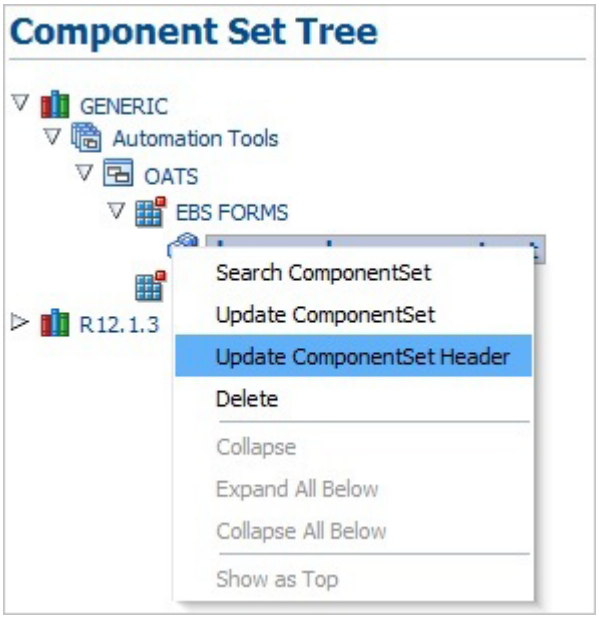
Existing component sets can be updated to add or remove components or components sets from a component set.

4.3.1 Updating Component Set Headers

To update component set header:

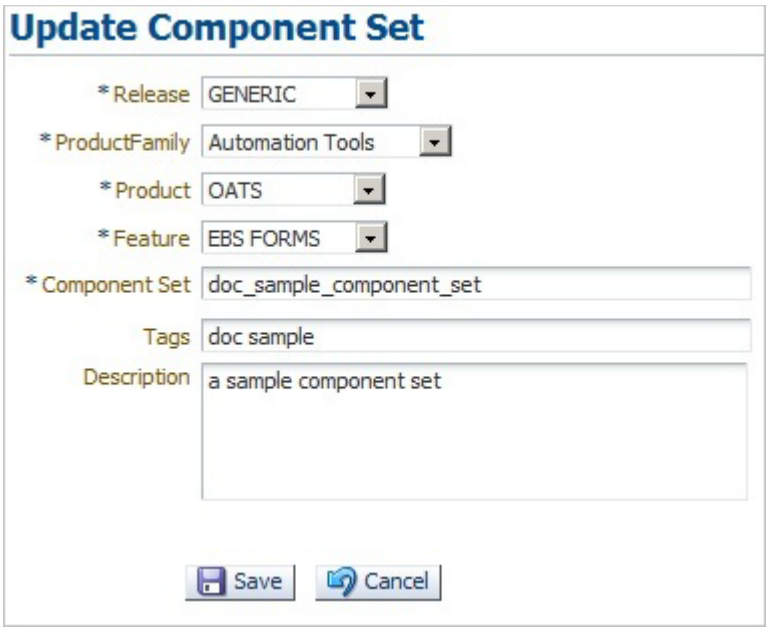
1. Expand the Component Set Tree to the component set you want to update.
2. Right-click the Component Set name and select **Update Component Set Header** from the shortcut menu.

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



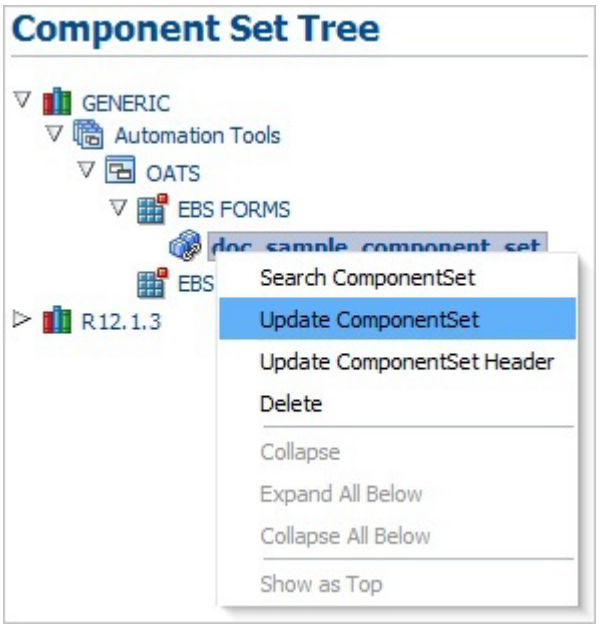
4. Click **Save** when finished to save and exit the Update Component Set pane. The Search pane indicates *Updated Component Set <set name> Successfully.*

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:

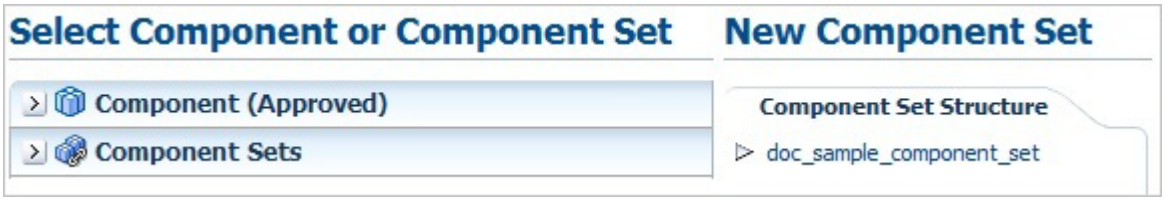
1. Expand the Component Set Tree to the component set you want to update.
2. Right-click the Component Set name and select **Update Component Set** from the shortcut menu.

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



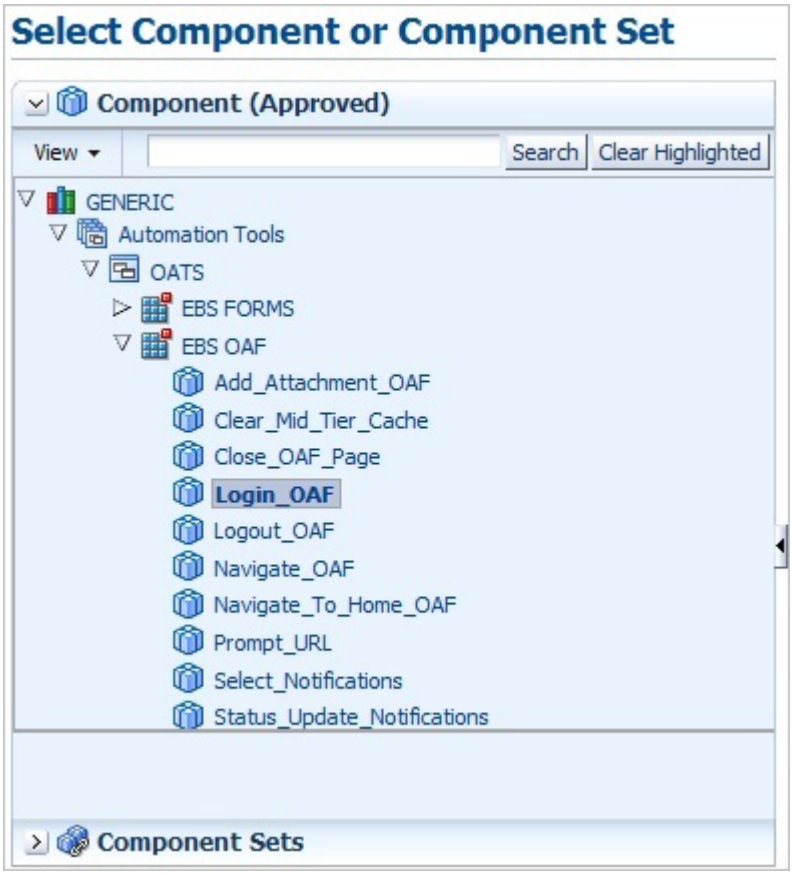
3. The Select Component or Components Set and the New Component Set pane opens for specifying the components or component sets to add to the new component set.

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

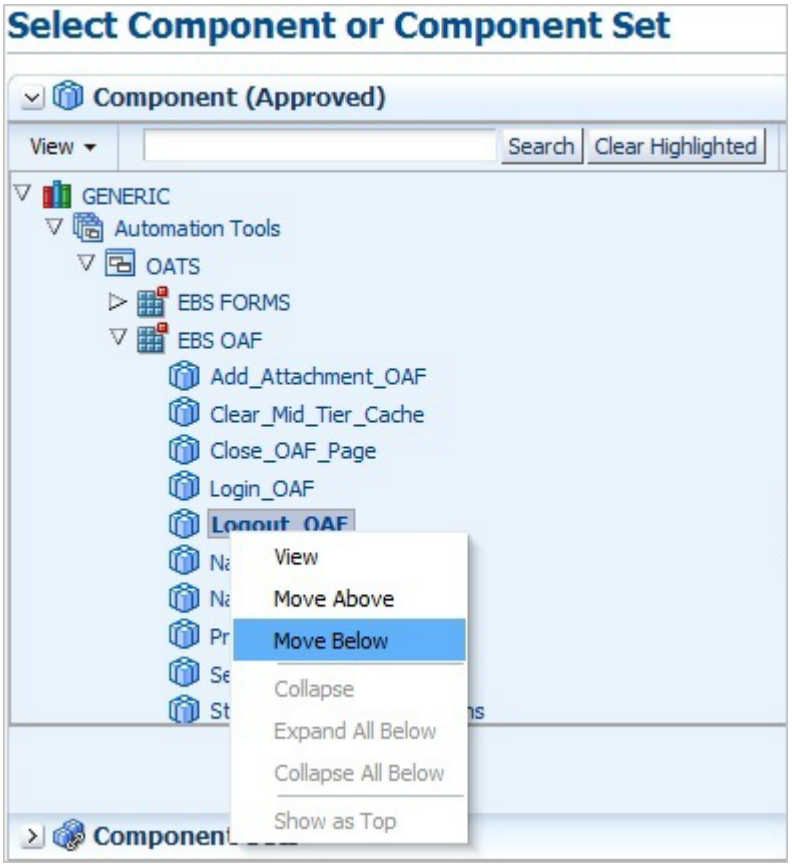


5. Expand the New Component Set tree to view the existing components or component sets.

Figure 4–17 New Component Set Pane with Components Added

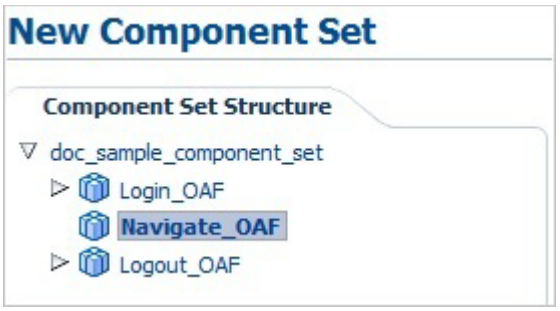
6. Select a component or component set in the New Component Set tree where you want to add the new component or component set.
7. Right-click the component or component set in the Select Component or Component Set tree that you want to add to the New Component Set and click **Move Above** or **Move Below**.

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



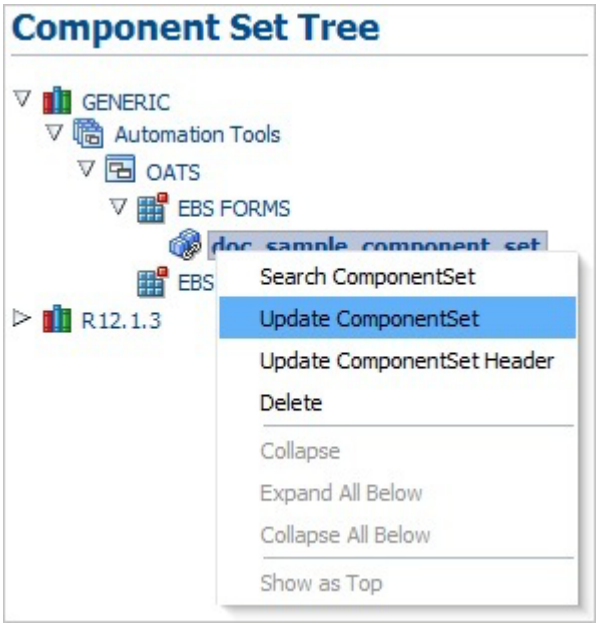
9. Click **Unlock** when finished to save and exit the New Component Set pane. The Search pane indicates *Publish Component Set Successfully.*

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

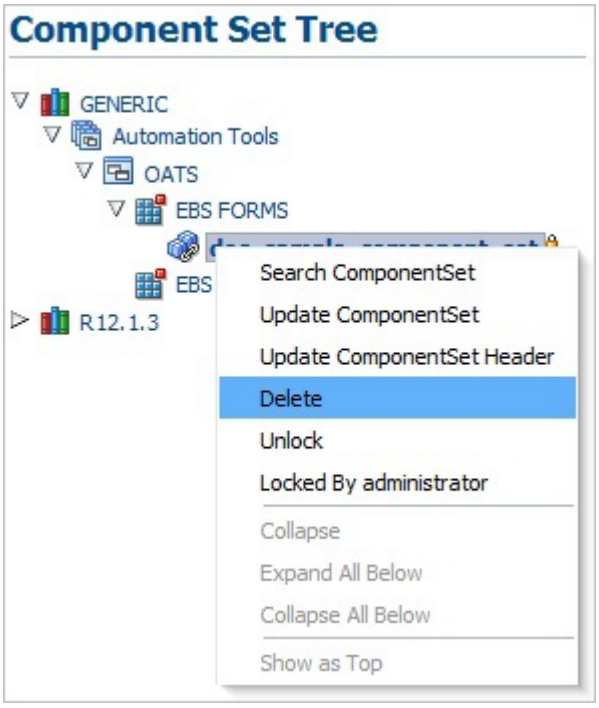
4. Right-click the component or component set in the New Component Set tree that you want to remove from the New Component Set and click **Delete**.
5. Click **OK** to confirm.
6. Repeat steps 4and 5 to remove additional components or component sets from the New Component Set as needed.
7. Click **Unlock** 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**.

4.5 Component Set Development Guidelines for PLSQL

Component Sets for PLSQL or Open Interface have specific component sequences that should be used when creating Component Sets. This section provides guidelines for creating these Component Sets.

4.5.1 PLSQL Component Sets

A PLSQL procedure is a combination of multiple components such as:

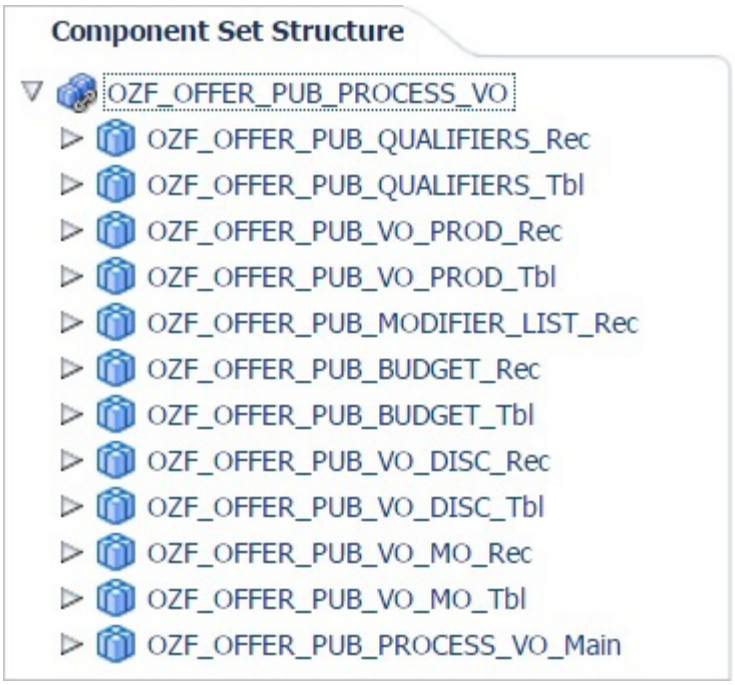
- PLSQL Procedure component
- PLSQL Record Component
- PLSQL Table Component
- PLSQL Object Component

A PLSQL Component Set is a set of components which are required to execute a specific procedure.

- PLSQL procedure component should come as a the last component in the Component Set.
- All the other components, such as record, table, etc., should be above the procedure component in the sequence.
- All the components should be added into the Component Set based on the parameter sequence.
- If there is a table component and this table is defined for records or objects, then components should be added as follows:
  - Record or Object Component
  - Table Component
- Sequence of Components in the component set should be a follows:
  - All the Record, Table or Object Components which are to be passed as input to the procedure.
  - All the Record, Table or Object Components which are returned as output from the procedure.
  - Procedure Component.

The following figure shows an example PLSQL Component Set with a sequence of components:

Figure 4–23 Example PLSQL Component Set



4.5.2 Open Interface Component Set

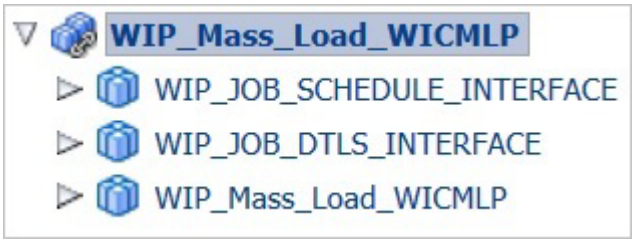
An Open Interface Component Set is a set of components which are required to execute a specific concurrent request after insertion of data into temporary tables.

An Open interface Component Set is a created in the following sequence:

- Inset Table components
- PLSQL Procedure Component

The following figure shows an example Open Interface Component Set with a sequence of components:

Figure 4–24 Example Open Interface Component Set



4.5.3 Webservice Component Set

A WebService Component Set is a set of components required to execute a Webservice payload. Webservice payload has multiple components such as Webservice Structure and Webservice node components.

A WebService Component Set is a created in the following sequence:

- Webservice Structure Component

- Webservice Node Component

The following figure shows an example Open Interface Component Set with a sequence of components:

Figure 4–25 Example Webservice Component Set

5

Defining Flows

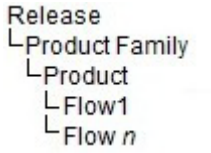
This chapter explains how to add and update flows in the Flow Tree. This chapter contains the following sections:

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, 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

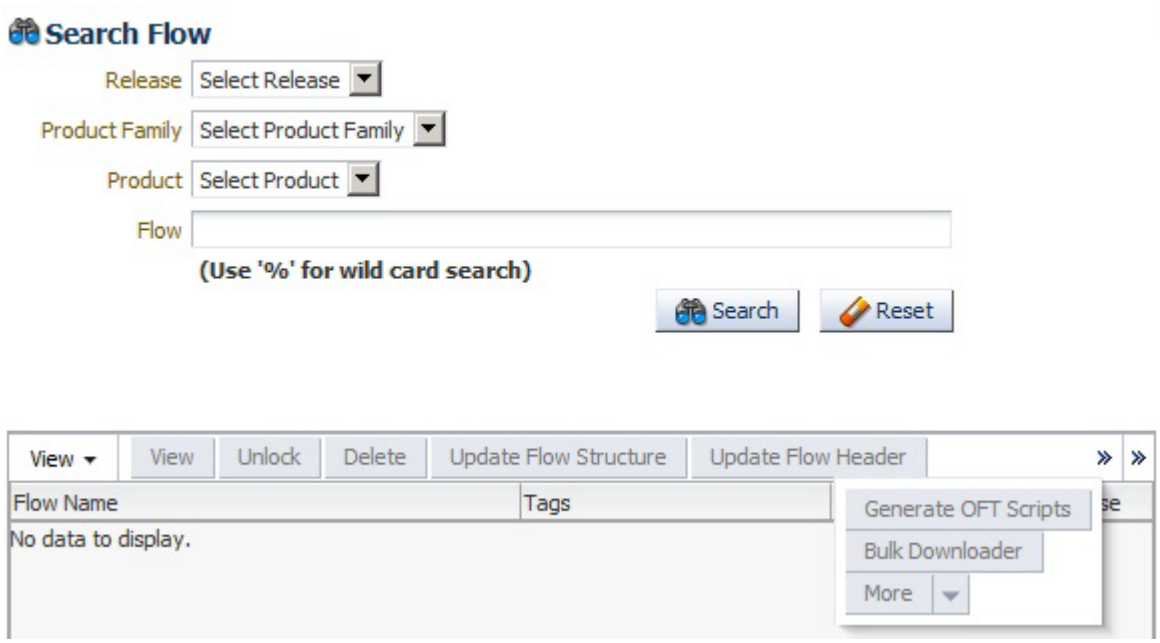
Release and application specific flows are listed below the Release(s) listed in the Flow Tree. Expanding the tree shows the list of products and features in the release tree:

Figure 5–4 Release Flow Tree

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

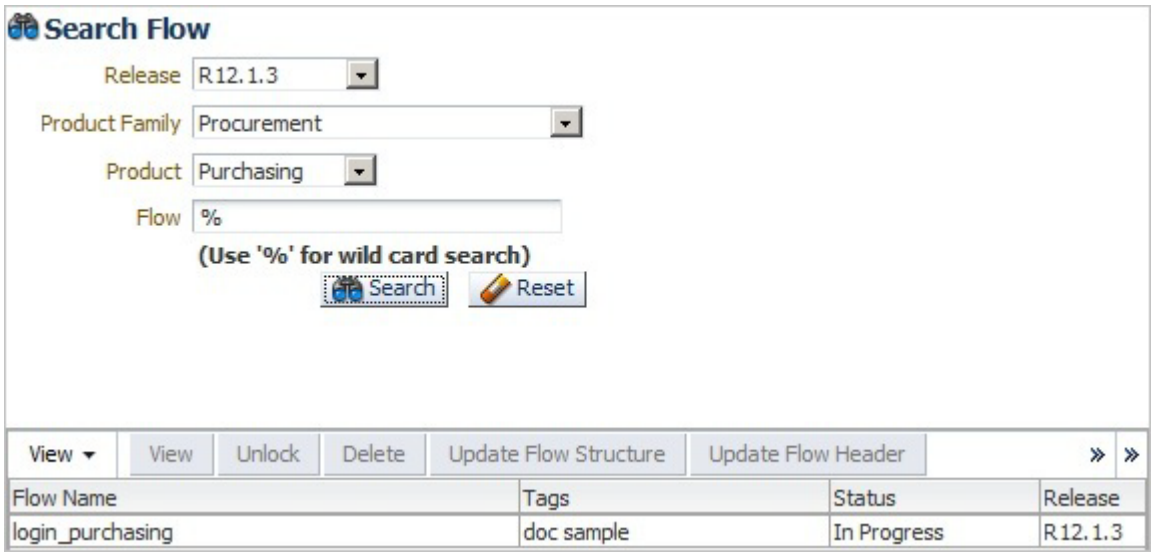
The Search pane of the Flow page lets you search for specific flows:

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.

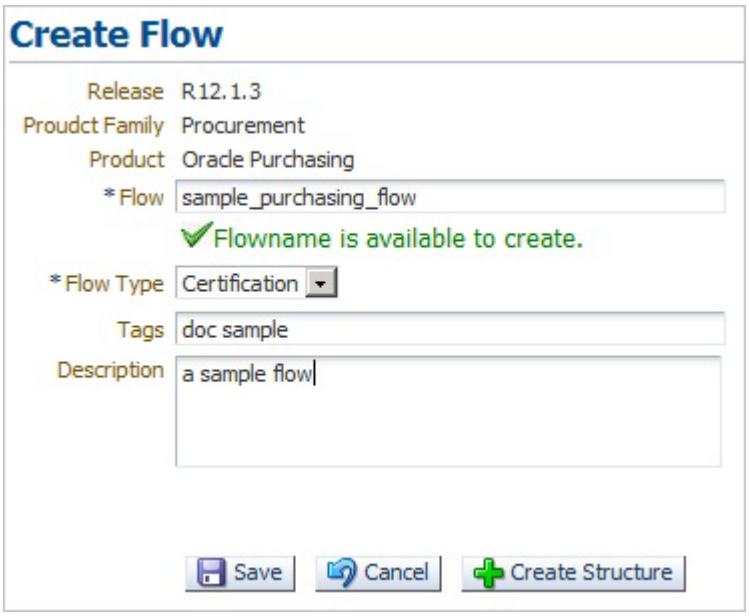
To add a flow to the Flows tree:

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.

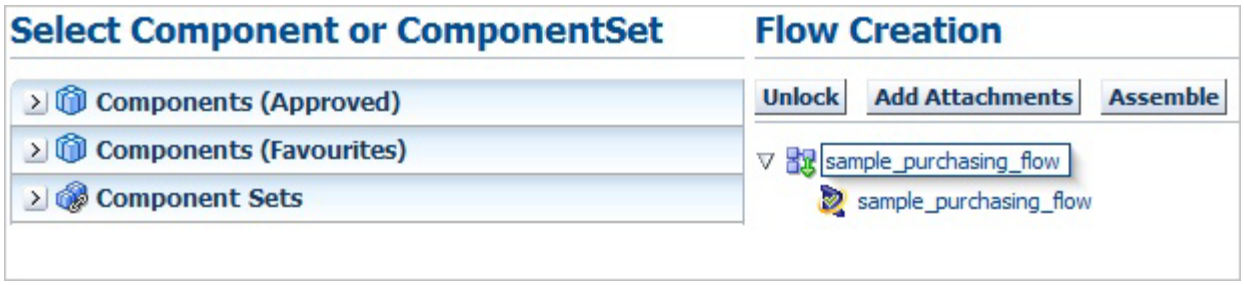
Figure 5–8 Create Flow Pane



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.

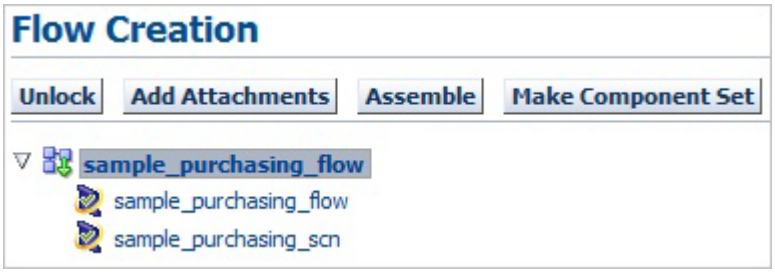
4. Click **Create Structure**. The Select Component or Components Set and the Flow Creation pane opens for specifying the components or component sets to add to the new flow and a new scenario is added to the flow tree. A scenario defines a specific OpenScript child script name within the overall flow. 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



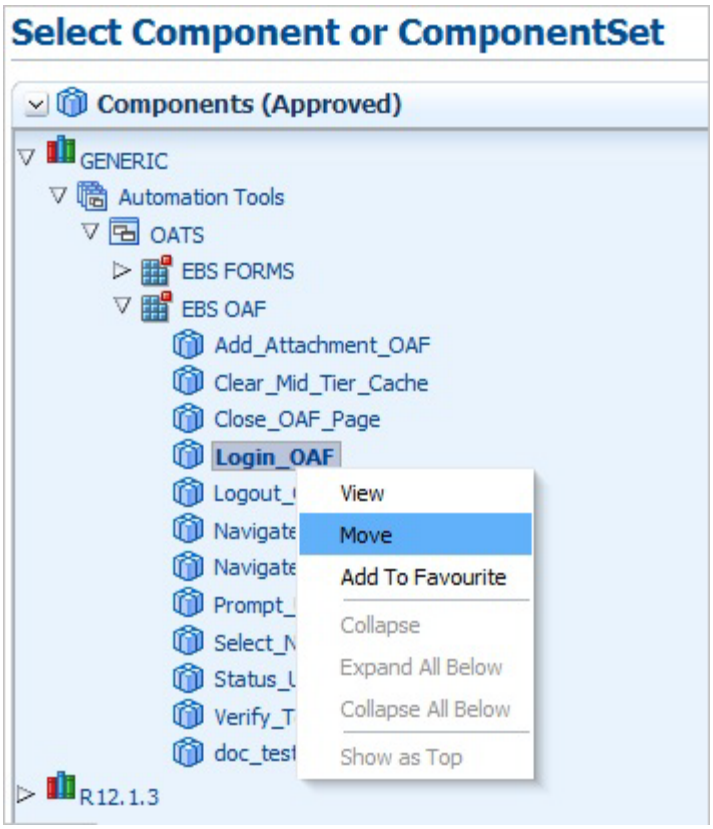
5. Expand the Flow tree and select the scenario where you want to add components or component sets.

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

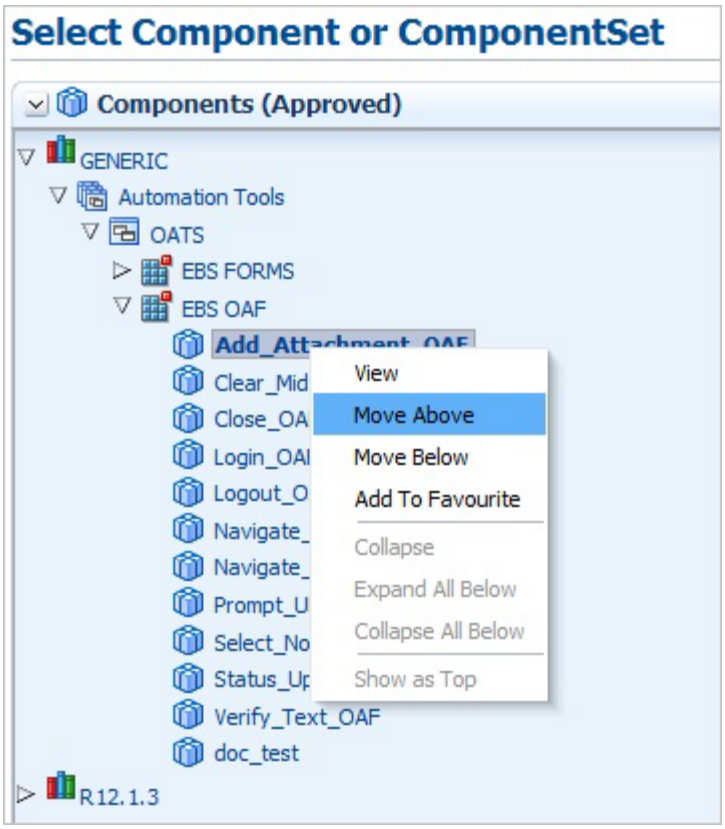


7. Right-click the component or component set to add to the Flow Creation tree and click **Move** (or click and drag the component or component set to the flow tree).

Figure 5–12 Flow Creation Pane with Components Added

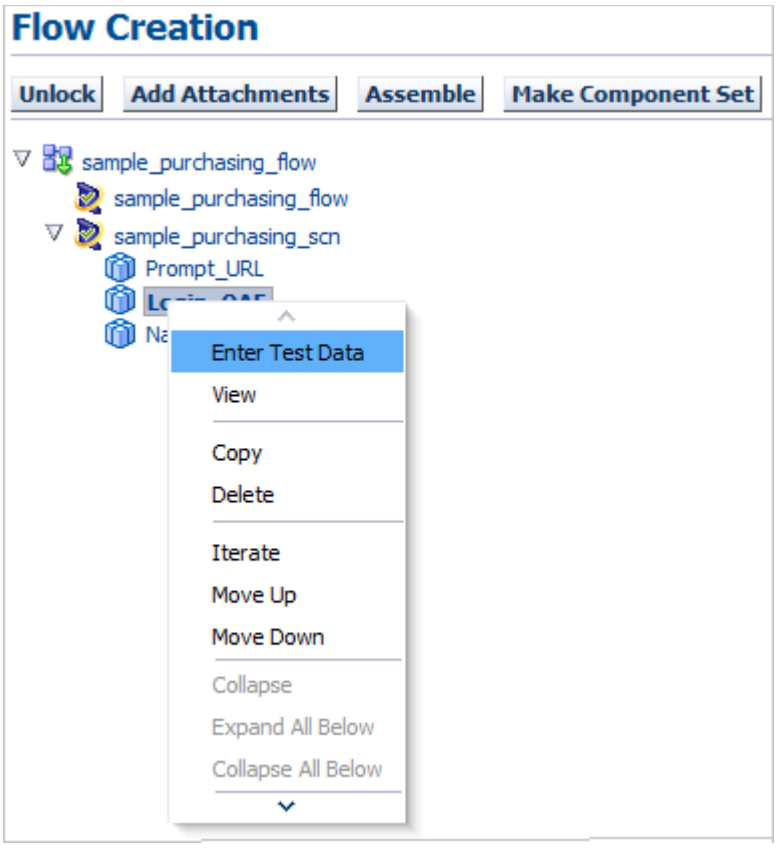
8. Repeat steps 5-7 to add additional components or component sets to the Flow Creation tree as needed. Select **Move Above** or **Move Below** to add the component or component set above or below the currently selected component or component set in the Flow Creation tree.

Figure 5–13 Select Component Pane Shortcut Menu



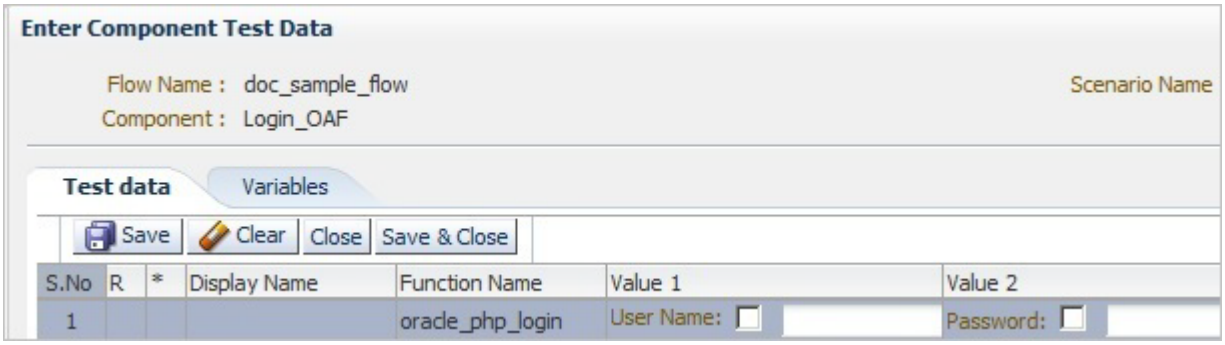
9. Right-click a component in the Flow and select **Enter Test Data** from the shortcut menu. You can also download a Test Data Excel file for the flow, enter the Test Data for the flow into the Excel file, then upload the Excel file. See Section 5.2.2, "Entering Test Data Using an Excel File" for additional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

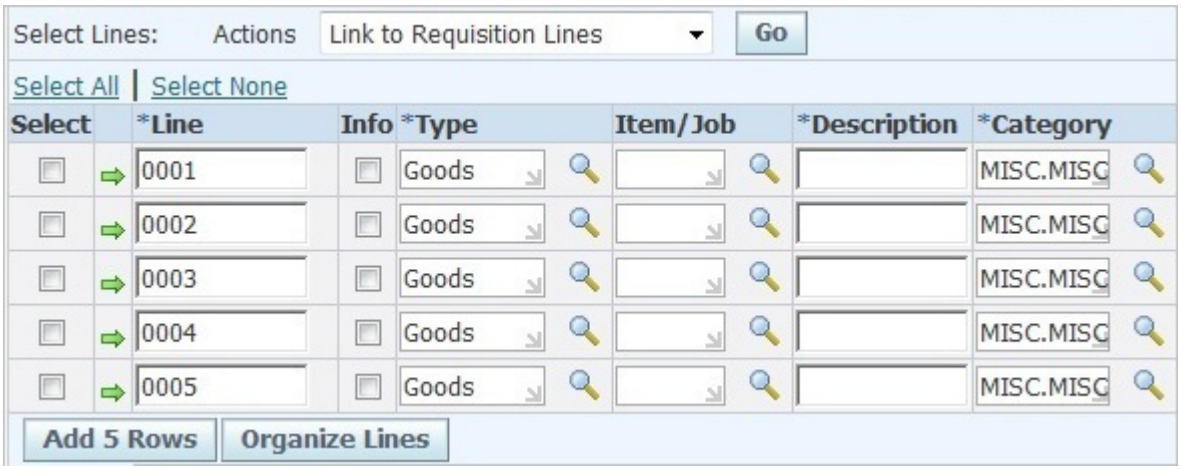
  Resp: Purchasing, Vision Operations (USA)

  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

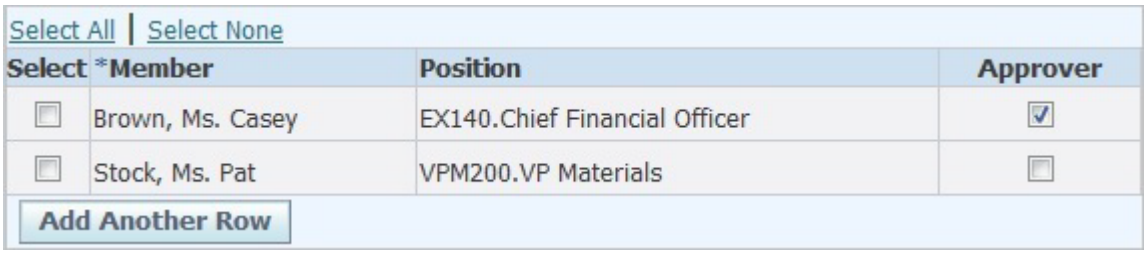
line number where values are to be entered. For example, to enter data for the 1st line in the form or table, enter 1 as the Test Data.

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



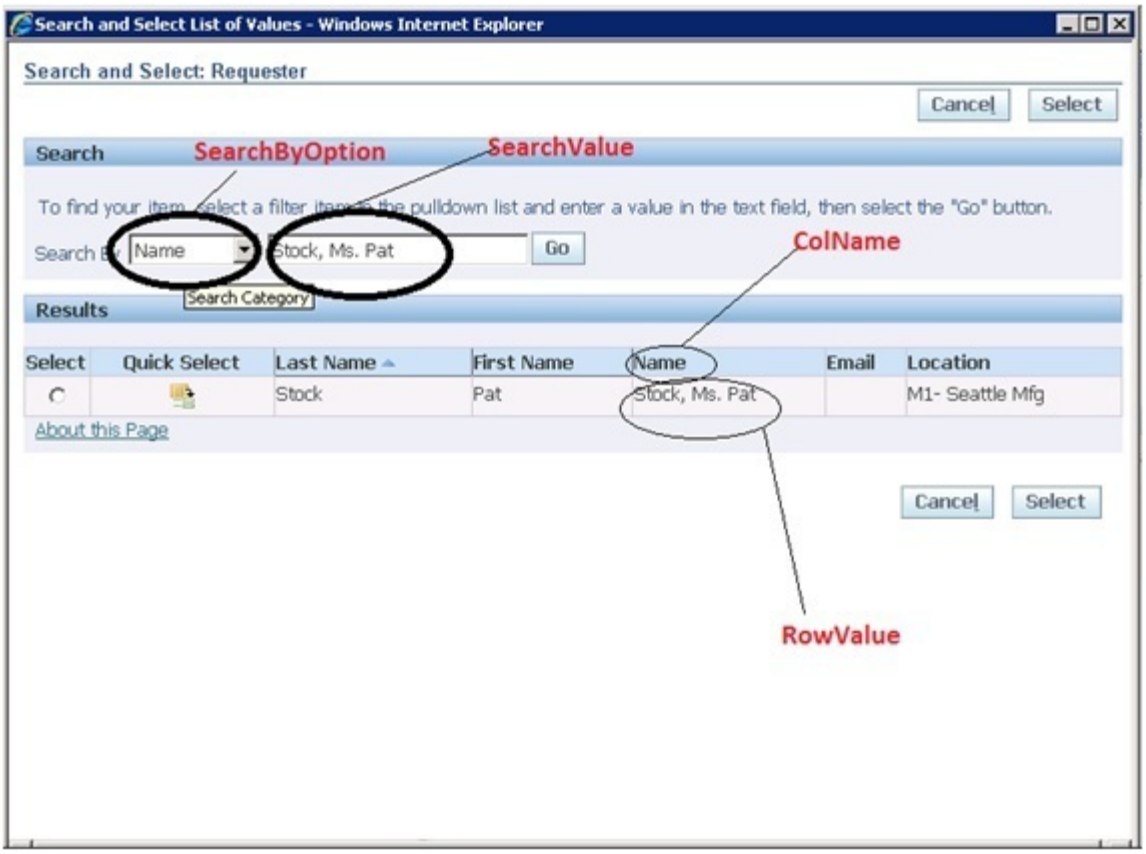
- **Webtable/Forms table-Verify/Update:** For components that require updating a line in a table or performing an action on any element in a table, the Test Data entered is the unique column name.For example, if the action to perform is check the checkbox in the "Select" column for *Member = Stock, Ms.Pat,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

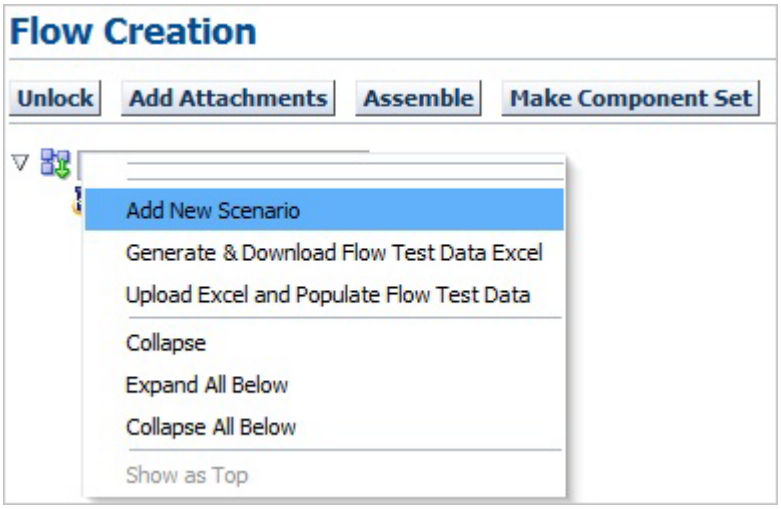
When you create a flow structure, a scenario name is automatically created to define the OpenScript child script name. Each scenario in a flow corresponds to an OpenScript child script. A flow may have only one scenario or it can have multiple scenarios depending upon the flow and the automation process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

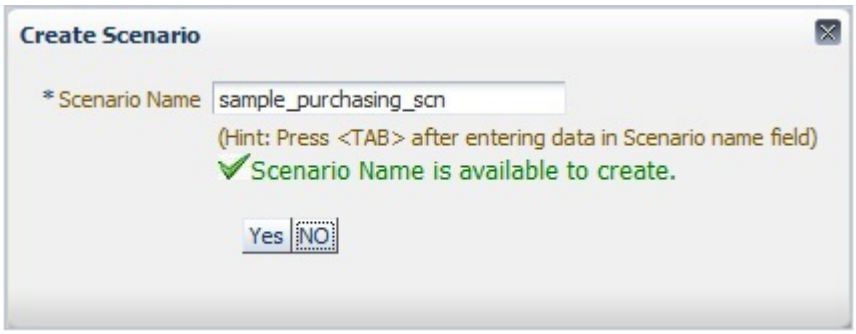
1. Open the flow tree and select the flow name.
2. Right-click the flow name and select **Add New Scenario** from the shortcut menu.

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

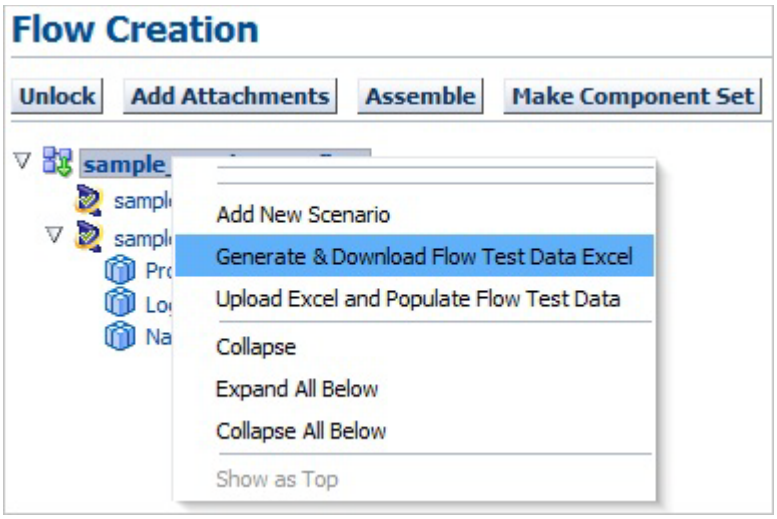
You can enter Test Data for an entire flow into an Excel file and then upload the Excel file to the flow. The basic steps are as follows:

- 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:

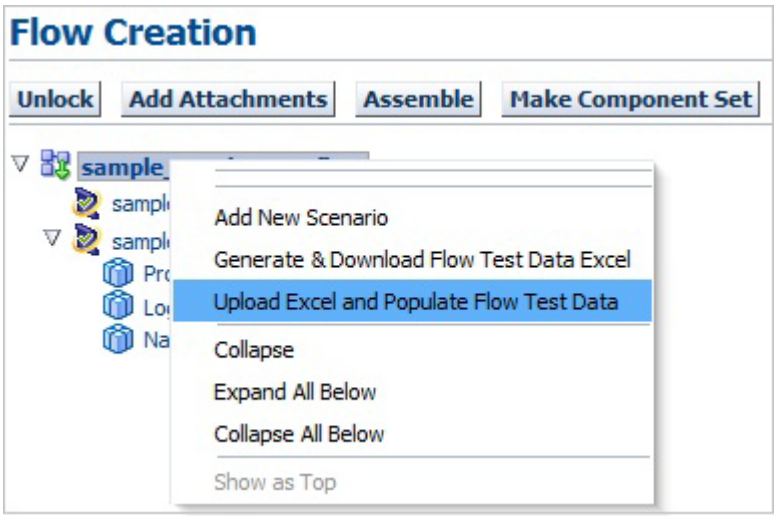
1. Create a new flow or search for an existing flow.
2. Select **Update Flow Structure**.
3. Right-click the flow name in the Flow Creation pane and select **Generate & Download Flow Test Data Excel** from the shortcut menu.

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

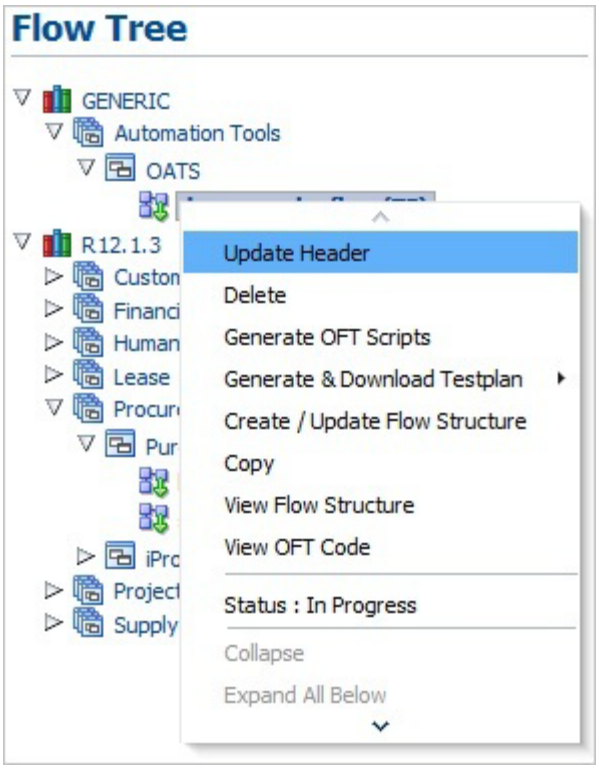
Existing flows can be updated to add or remove components or components sets from a flow or change Test Data.

5.3.1 Updating Flow Headers

To update a flow header:

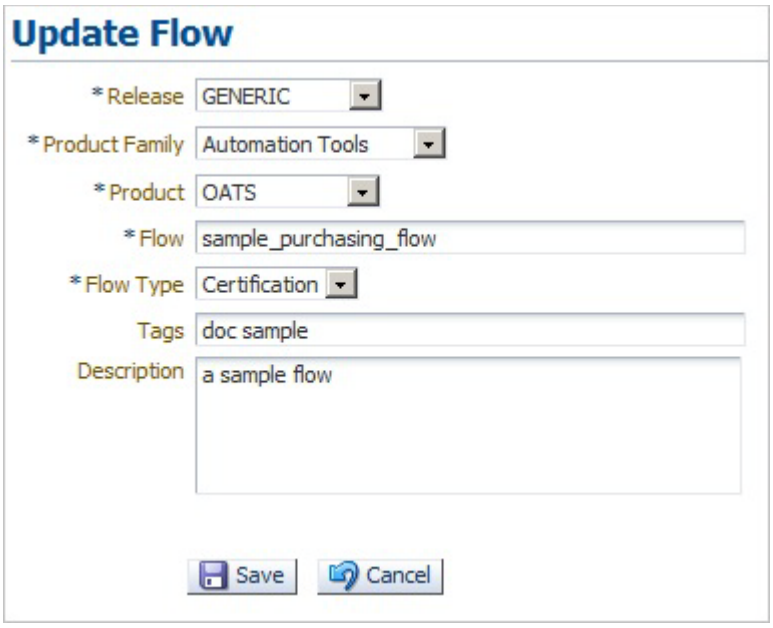
1. Expand the Flow Tree to the flow you want to update.
2. Right-click the flow name and select **Update Header** from the shortcut menu.

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



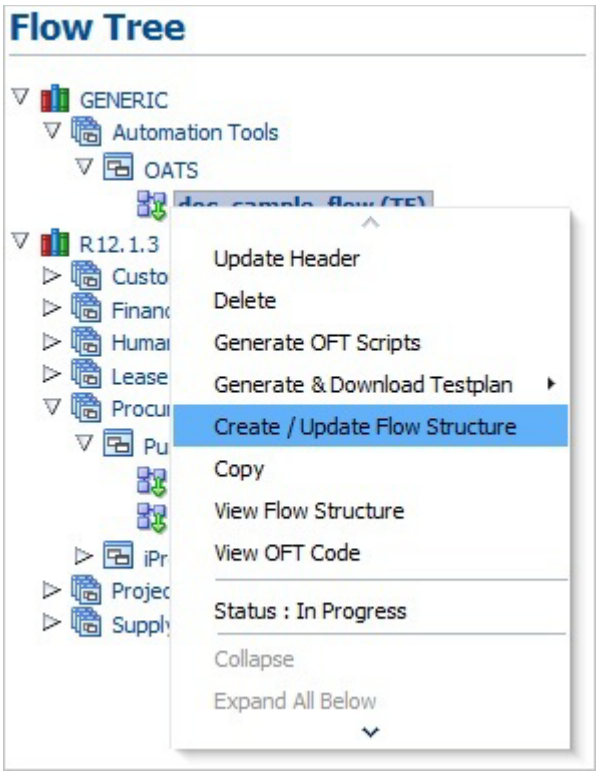
4. Click **Save** when finished to save and exit the Update Component Set pane. The Search pane indicates *Updated Flow <name> Successfully.*

5.3.2 Adding Components or Component Sets to an Existing Flow

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

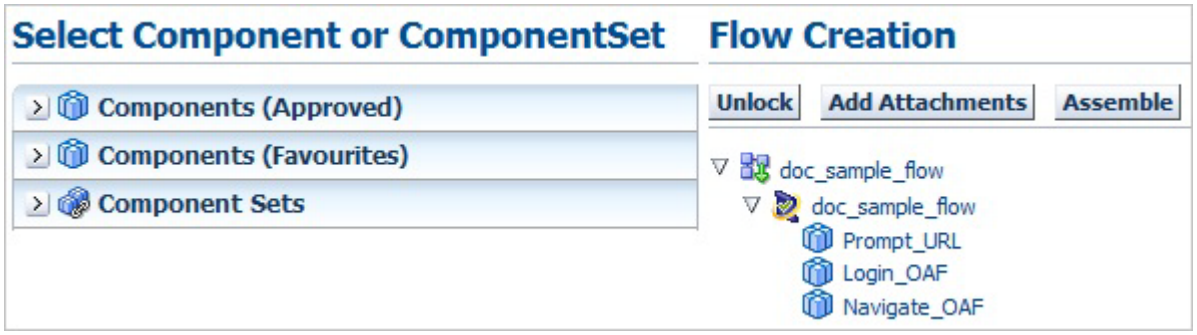
1. Expand the Flow Tree to the flow you want to update.
2. Right-click the Flow name and select **Create/ Update Flow Structure** from the shortcut menu.

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

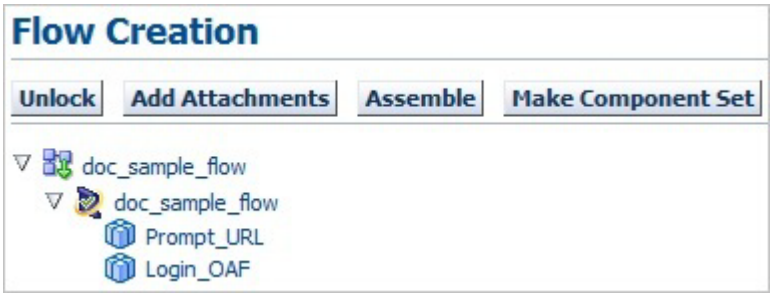


4. Expand the Components (Approved), Components (Favourites), or Components Sets tree to view the available components or component sets.

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

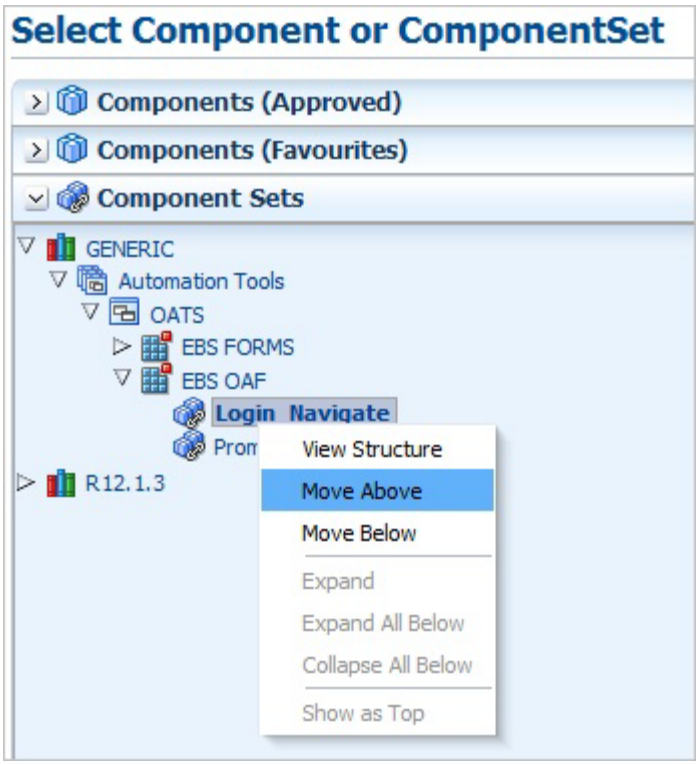
5. Expand the Flow Creation tree to view the existing components or component sets.

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

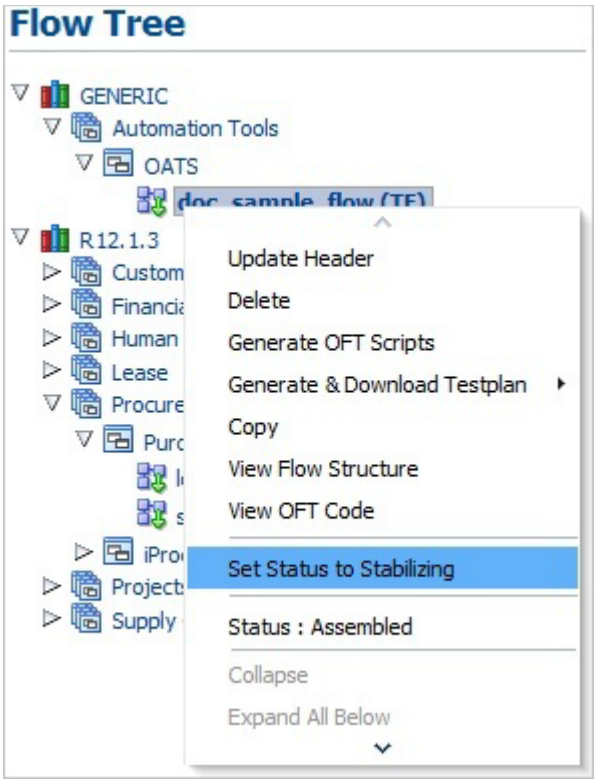


8. Repeat step 7 to add additional components or component sets to the Flow Creation tree Component Set as needed.

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

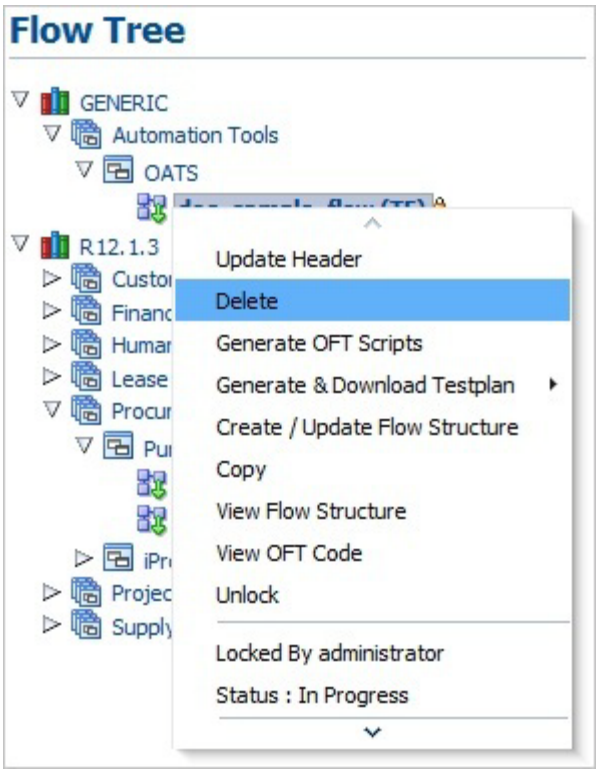


5.4 Deleting Flows

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



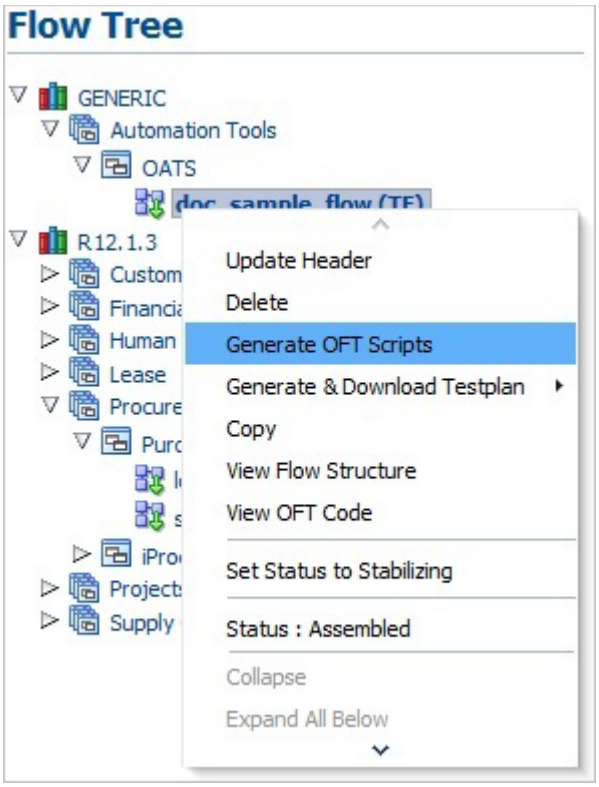
3. Click **Yes** to confirm.
4. Click **OK**.

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



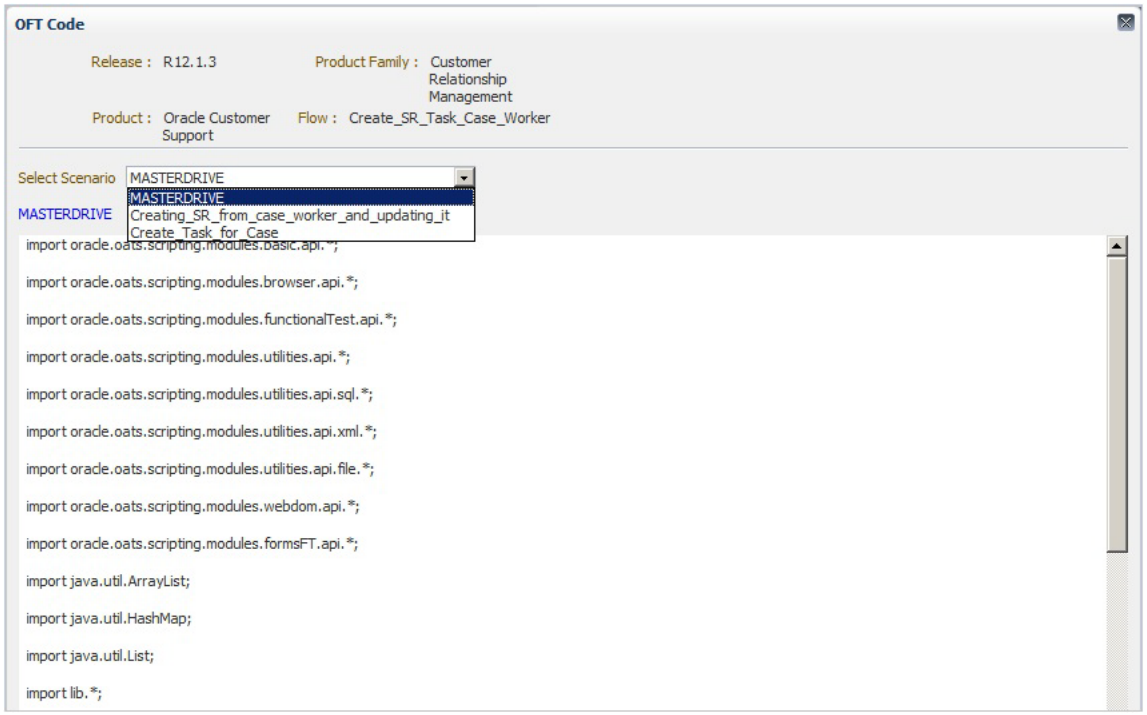
3. Enter the Execution location folder.
4. Enter the Normal wait time.
5. Enter the Page wait time.
6. Select if the script is rerunable or not.
7. If Rerunable, select the time zone, specify the Rerun type.
8. Select the time zone.
9. Click **Download Suite**.
10. Extract the zip file into the OpenScript 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.

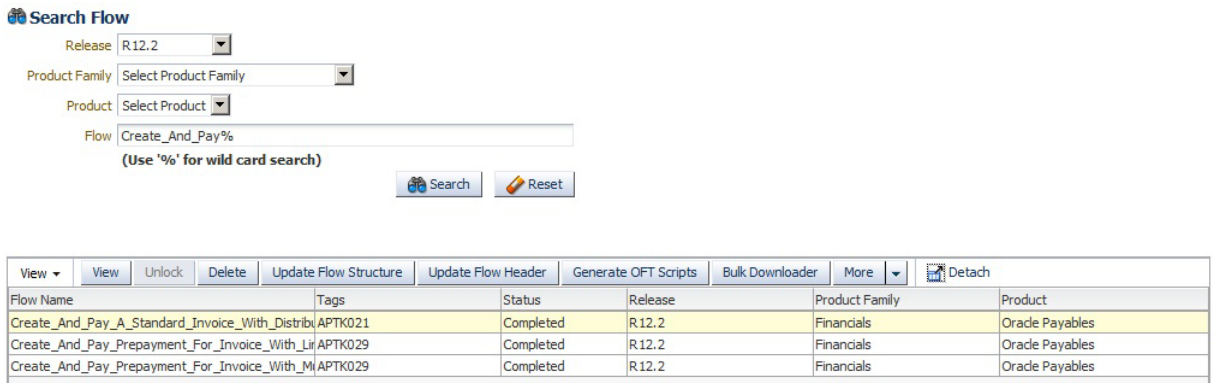
5.7 Using the Bulk Downloader

The Bulk Downloader lets you select and download multiple flows as generated OFT scripts.

To download multiple flows as generated OFT scripts:

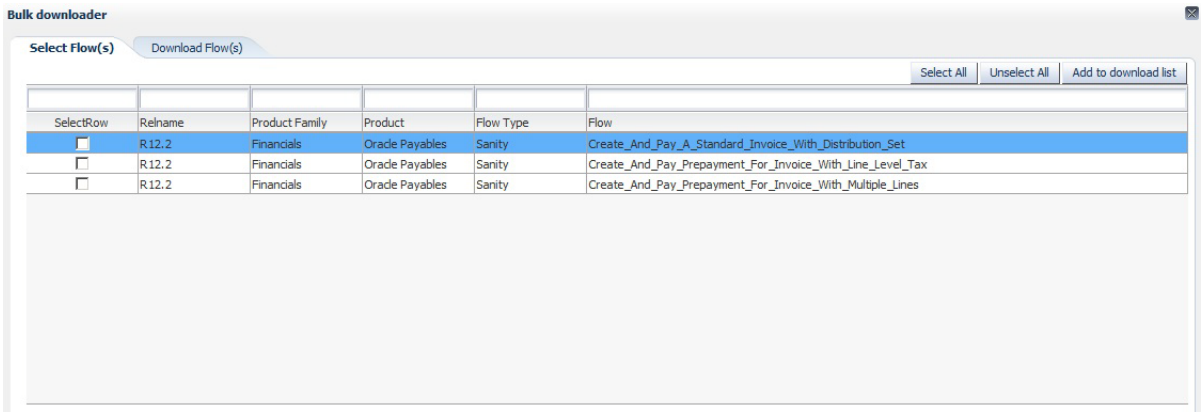
1. Search for the flows you want to download using the Search criteria.

Figure 5–35 Search Pane with Search Criteria and Results



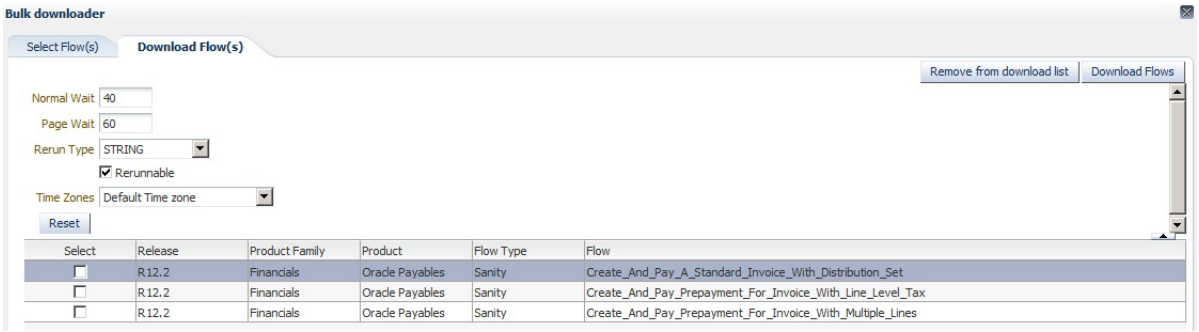
2. Select a flow in the Search Results.
3. Click the **Bulk Downloader** button.

Figure 5–36 Bulk Downloader with Flows Available for Download



4. Select the flows to include in the download by selecting the Select Row check box for the flow(s) or the **Select All**/**Unselect** All buttons.
5. Click the **Add to download list** button.
6. Click the **Download Flow(s)** tab.

Figure 5–37 Bulk Downloader with Flows Added for Download



7. Set the Wait times, Rerun Type, Rerunable option, and Time Zone.
8. Click the **Download Flows** button. Depending upon the number of flows selected, it may take a few moments for Oracle Flow Builder to generate the OFT scripts and create the BulkDownload_Administrator_#.zip file.
9. When the File dialog box appears, select **Save File** and click **OK**.
10. Save the BulkDownload_Administrator_#.zip file.
11. Extract the zip file into the OpenScript repository.
12. Start OpenScript.
13. 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.

5.8 Generating Test Plans

You can generate a Test Plan document for flows using the **Generate & Download Test Plan** option on the **More** menu of the Search pane.

To generate sand download a Test Plan for a flow:

1. Search for the flow you want to generate and download a Test plan for using the Search criteria.
2. Select the Flow name in the Search Results.
3. Select the **Generate & Download Test Plan** option on the **More** menu.
4. When the File dialog box appears, select **Save File** and click **OK**.
5. Select the file location and click **Save**.
6. Extract the *flowname*.zip file to view the .doc, .html, or .log file in the appropriate editor/viewer.

5.9 Flow Development Guidelines for PLSQL Flows

This section provides guidelines for developing flows for PLSQL flows.

- An OFB flow can have multiple scenarios.
- Each scenario in a flow can have only one of the following as scenario module:
  - PLSQL Procedure
  - Open Interface
  - WebService
  - UI Scenario

  A Scenario can not have a combination of PLSQL / Open Interface / Webservice / UI components.
- Oracle Flow Builder provides the flexibility to create hybrid flows. For example, a flow can have combination of different interfaces and GUI.
- Oracle Flow Builder supports Multiple Test Data. For example, a single scenario with multiple sets of data. Multiple Test Data is supported for the following interfaces:
  - PLSQL
  - Open Interface
  - Webservice
- The better practice is to have a hybrid flow with a single set of Test Data.
- Multiple Test Data is supported at the scenario level.
- Excel is used for flow level Test Data only for GUI.
  - Use Excel for flow level Test Data for GUI flows.
  - Do not use Excel for flow level Test Data for interfaces like PLSQL, Open interface and Webservice with single or Multiple Test Data.
  - Do not use Excel for flow level Test Data for hybrid flows.

5.9.1 Adding PLSQL Flows

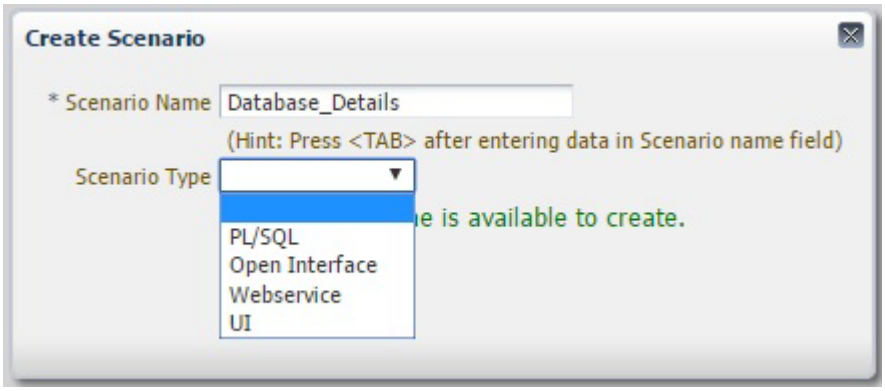
Oracle Flow Builder supports creation of flow for execution of PLSQL procedures with single and Multiple Test Data. OFB PLSQL flow can be created as follows:

- Use an Oracle Flow Builder Flow scenario to provide database details.

Create Scenario by right clicking on the flow and selecting **Add New Scenario**. Provide the following details in the **Create Scenario** dialog box.

- Scenario Name
- Scenario Type as Empty

Figure 5–38 Create Scenario Details: Database



Drag the component "Database_WS_Details" into the scenario.

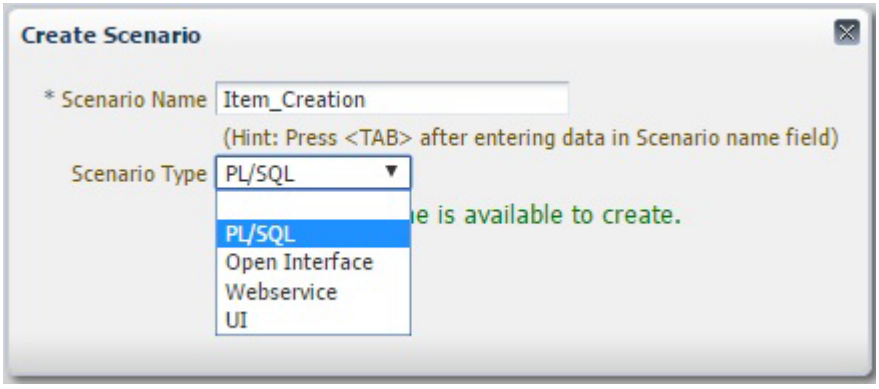
- Each procedure call should have one Flow scenario.
- A separate scenario is to be created after PLSQL procedure execution scenario for verification of data and to retrieve any data from the database using queries. The following components are required for verification and select query:
  - Verification Components - for verification of data.
  - Select Components - to retrieve data from the database.

5.9.1.1 PLSQL Flow Scenario Structure

Create Scenario by right clicking on the flow and selecting Add New Scenario. Provide the following details in the Create Scenario dialog box as follows:

- Scenario Name
- Scenario Type as "PL/SQL"

Figure 5–39 Create Scenario Details: PL/SQL



The Scenario Type will be displayed next to the scenario names as "(PLSQL)".

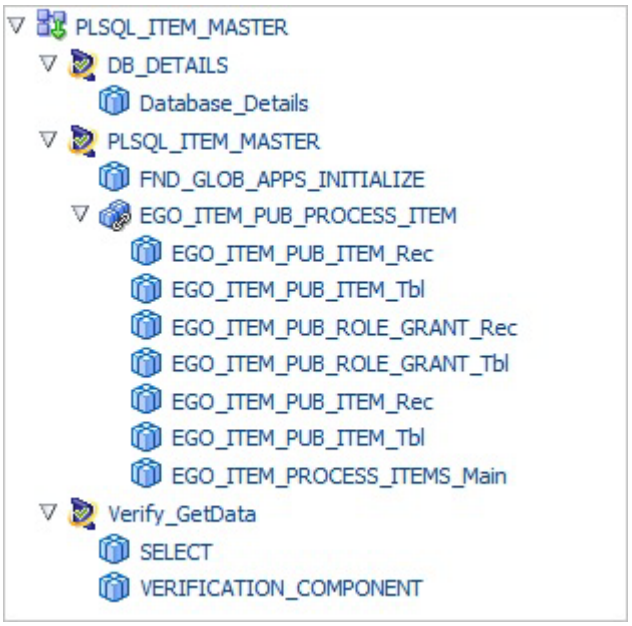
A PLSQL scenario should have following structure:

- Add components related to PLSQL initialization like FND_GLOBAL.APPS_INITIALIZE, which is required to execute a specified procedure. Some of the initialization/Org Context components are as follows:
  - FND_REQUEST_SET_ORG_ID
  - FND_GLOBAL_SET_NLS_CONTEXT
  - MO_GLOBAL_SET_POLICY_CONTEXT
  - MO_GLOBAL_INIT
- Add Component Set(s) related to the procedure to be executed.

> **Note:** If a procedure has only "Standard Types" as parameters, then there will be only one component available for that procedure and there will not be any Component Set created for that. You need to drag the Component instead of the Component Set while creating the PLSQL scenario for that procedure execution.

The following shows an example PLSQL Flow Structure:

Figure 5–40 Example PLSQL Flow Structure



5.9.1.2 Test Data for PLSQL Flows

There are two types of Test Data for PLSQL flows:

- PLSQL flow with a single set of Test Data.
- PLSQL flow with multiple sets of Test Data.

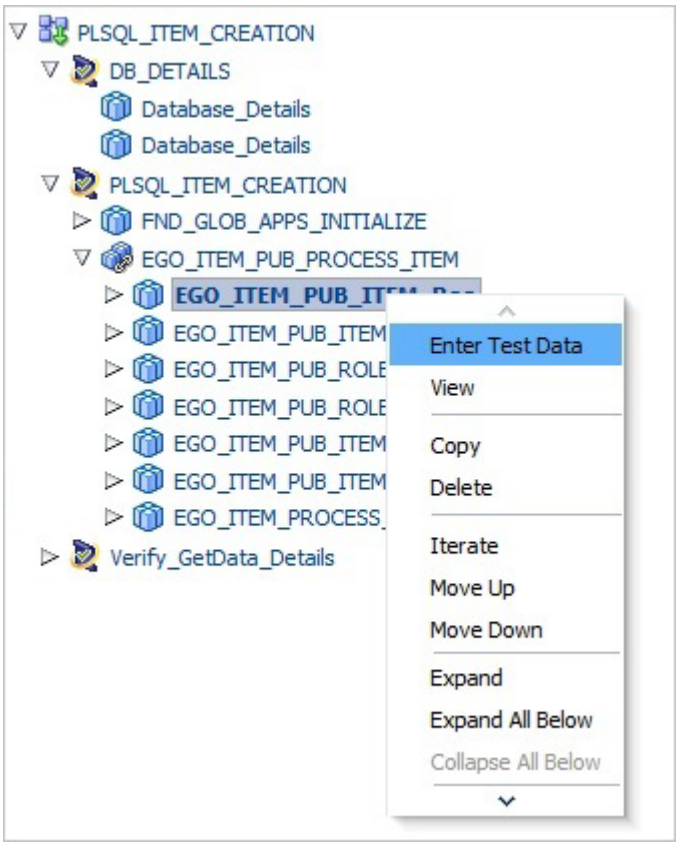
The following sections provide details for entering Test Data for the two types of flows.

5.9.1.2.1 PLSQL Flows with a Single Set of Test Data

For PLSQL Flows with a single set of Test Data, do the following:

1. Right-click on a component in the PLSQL scenario and select **Enter Test Data** from the menu.

Figure 5–41 Example Enter Test Data Menu Option for PLSQL Flows



2. Specify the Test Data for the PLSQL procedure and its associated components, such as Records/Table of Records, etc.

   a. Specify the Test Data for the all the Standard type parameters that are available in the procedure component.

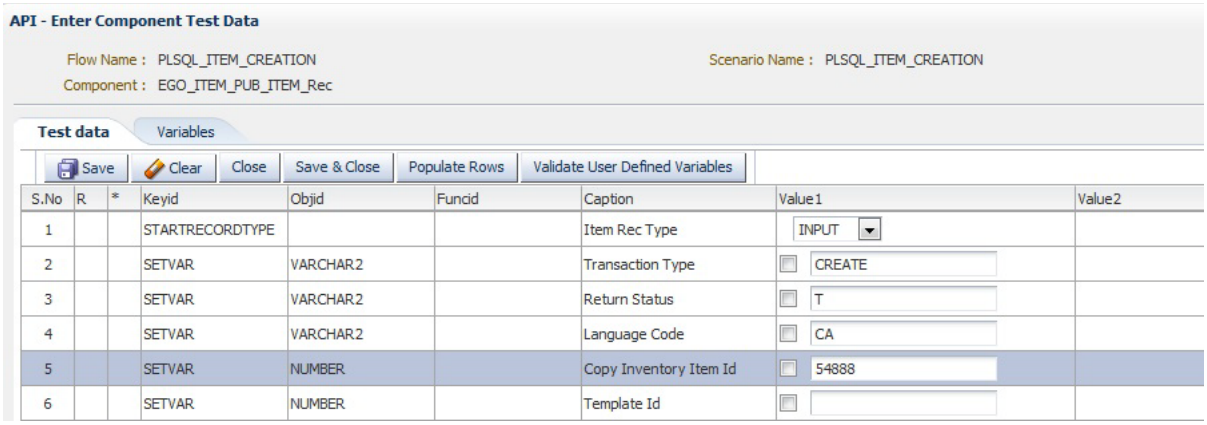
   b. If any user defined data type like **Record/Object is taken as input** to the procedure, then go the corresponding component in the component set and specify the data for that component.

      Specify data as “INPUT” for the first field “START RECORD TYPE”/”START OBJECT TYPE”.

      Specify data for the Record/Object fields.

      The following figure shows an example of the Record/Object Component Test Data.

Figure 5–42 Example Record/Table Component Test Data



c. If any user defined data type like **Table of Records/Table of Objects is taken as input** to the procedure, then go the corresponding component in the component set and specify the data.

Table of Records/Objects corresponds to two components:

- Record/Object

- Table

Need to drag the Record/Object component above the table component based on the number of records to be passed to the table.

Specify the data for all the Record/Objects.

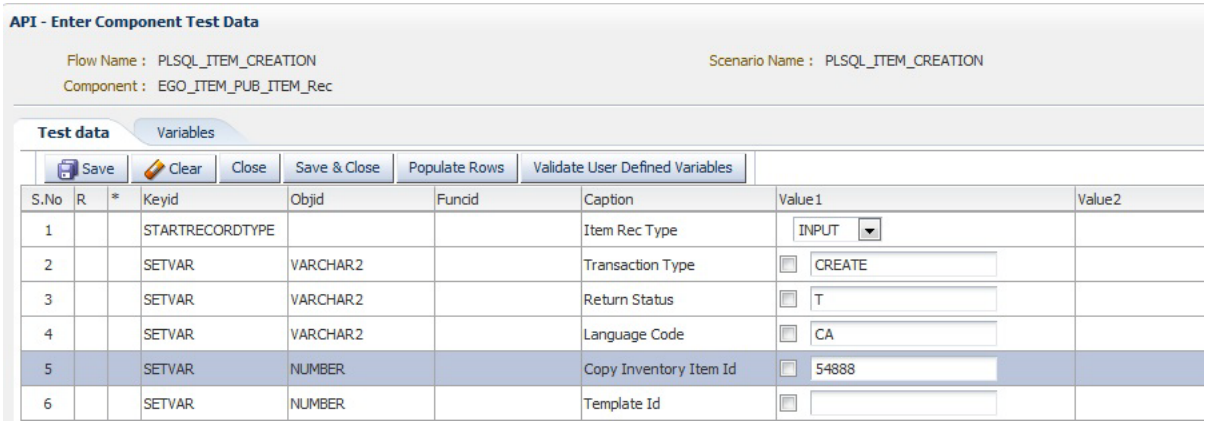
Specify the data for the table component, as follows:

- Specify the number of rows that are to be passed to the table at "STARTROWITERATOR" and click "Populate Rows".

- Associate all the dragged record/object component variables in the table component for each row.

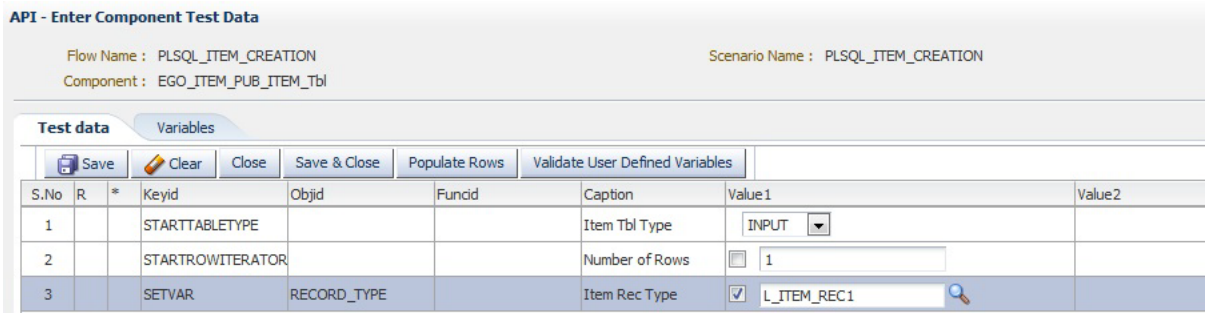
The following figure shows an example of the Record/Object Component Test Data.

Figure 5–43 Example Record/Object Component Test Data



The following figure shows an example of the Table Component Test Data.

Figure 5–44 Example Table Component



d. If any user defined data type like **Record/Object is returned as output** from the procedure, then go the corresponding component in the component set and provide the data as mentioned.

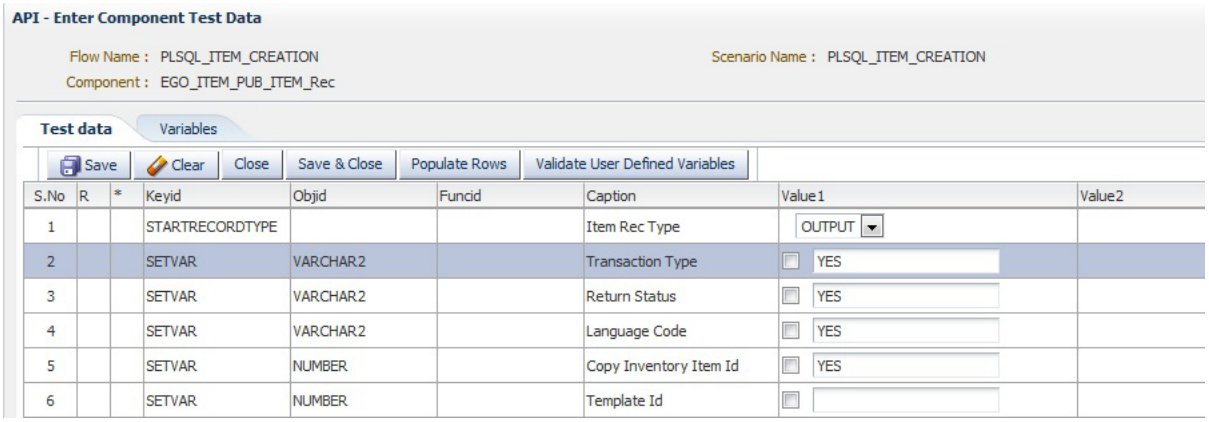
- Specify data as "OUTPUT" for the first field "START RECORD TYPE"/"START OBJECT TYPE".

- Specify data as "YES" for all the fields which are required as output from specified Record/Object.

- These fields can be used in the subsequent scenarios as {{RecordVariable_FieldName}}.

The following figure shows an example of the Record/Object Component Test Data.

Figure 5–45 Example Record/Object Component Test Data



e. If any user defined data type like **Table of Records/Table of Objects is returned as output** from the procedure, then go the corresponding components in the component set and specify the data as follows.

- Specify the data for the Record/Object component of the table as follows:

1. Specify data as "OUTPUT" for the first field "START RECORD TYPE"/"START OBJECT TYPE".

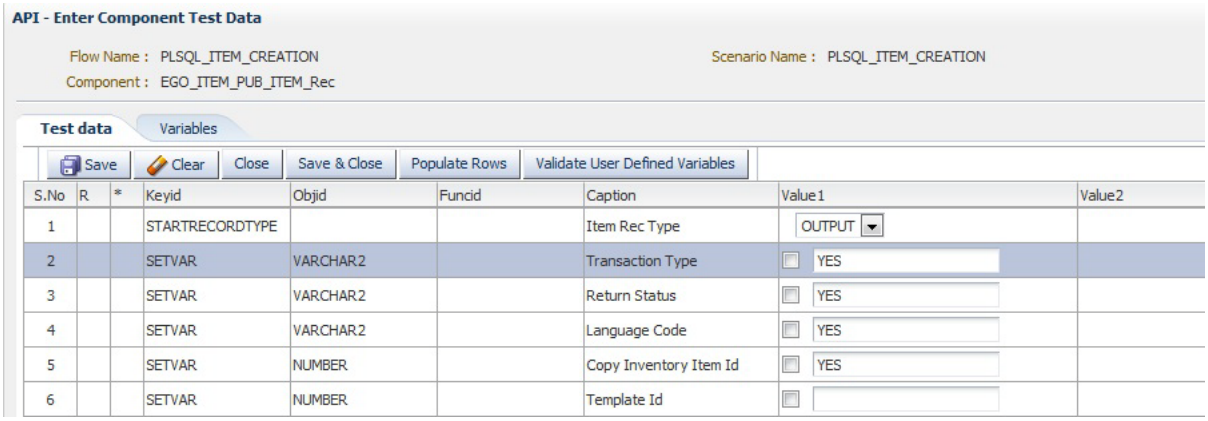
2. Specify data as "YES" for all the fields which are required as output from specified Record/Object.

- Specify data as "OUTPUT" for the first field "START TABLE TYPE" of the table component.

- These fields can be used in the subsequent scenarios as {{TableVariable_RowNumber_FieldName}}.

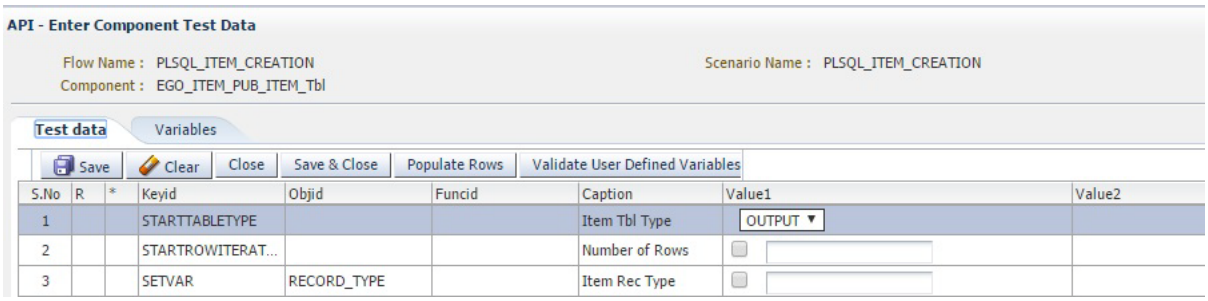
The following figure shows an example of the Record/Object Component Test Data.

Figure 5–46 Example Record /Object Component Test Data



The following figure shows an example of the Table Component Test Data.

Figure 5–47 Example Table Component Test Data

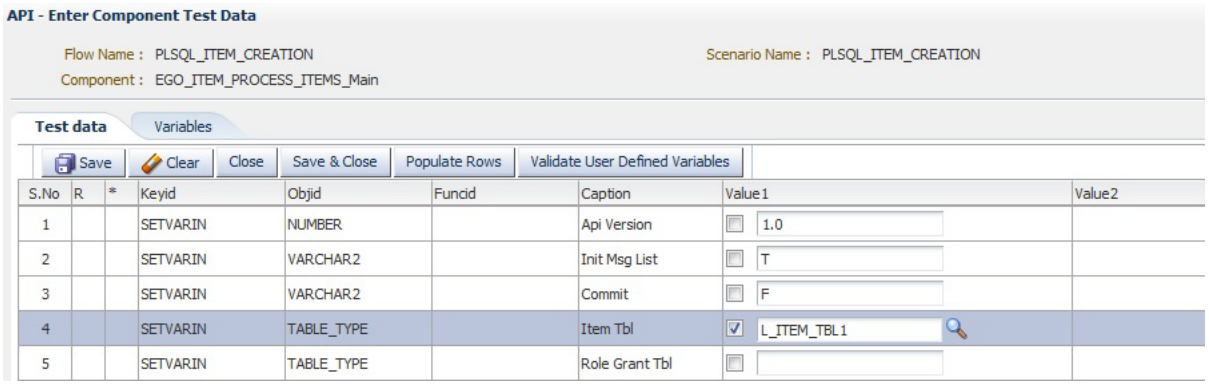


f. Specify the Test Data for the main procedure as follows:

- Specify Test Data for all the "Standard data types"(Varchar, Boolean, Number etc.) with the required data.

- Test data for all the "User defined data types"(Record/Object/Table of Objects/Table of Records) can be provided by selecting the variable name of the corresponding components.

Figure 5–48 Example Procedure Test Data



g. Specify Test Data for DATE field as follows:

Value1: Specify date as #SYSDATE(+n), for example: #SYSDATE(+1)

#SYSDATE(0) - Today's date

#SYSDATE(+1) - Tomorrow's date

#SYSDATE(-1) - Yesterday's date

Value2: Specify date format as "DD-MON-YYYY"

The following figure shows an example of the Date Test Data.

Figure 5–49 Example Date Test Data

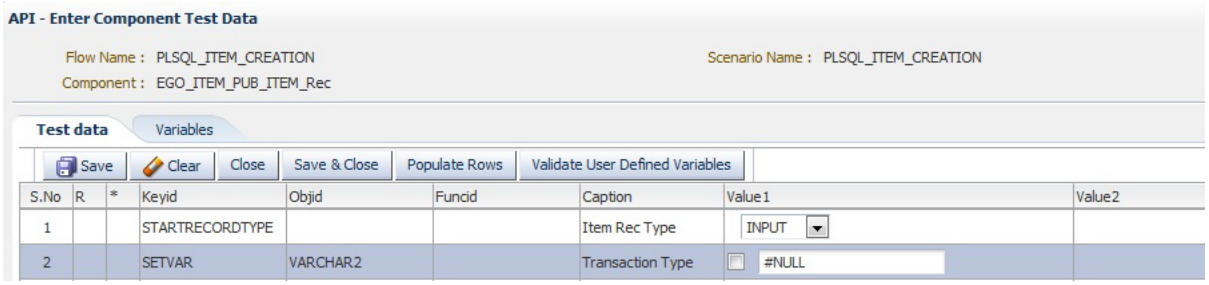
h. Specify Empty in the Test Data as #EMPTY.

i. Specify a function call in the Test Data as #FUNCTIONCALL(functionName).

j. Specify Null in the Test Data as #NULL.

The following figure shows an example of the Null Test Data.

Figure 5–50 Example Null Test Data



3. Providing Test Data for some of the procedure requirements

a. Passing user defined types like record/table of records to the procedure without providing any Test Data, for example:

- Declare Record/Table of Records variable

- Pass the Record/Table of Records variable to the procedure.

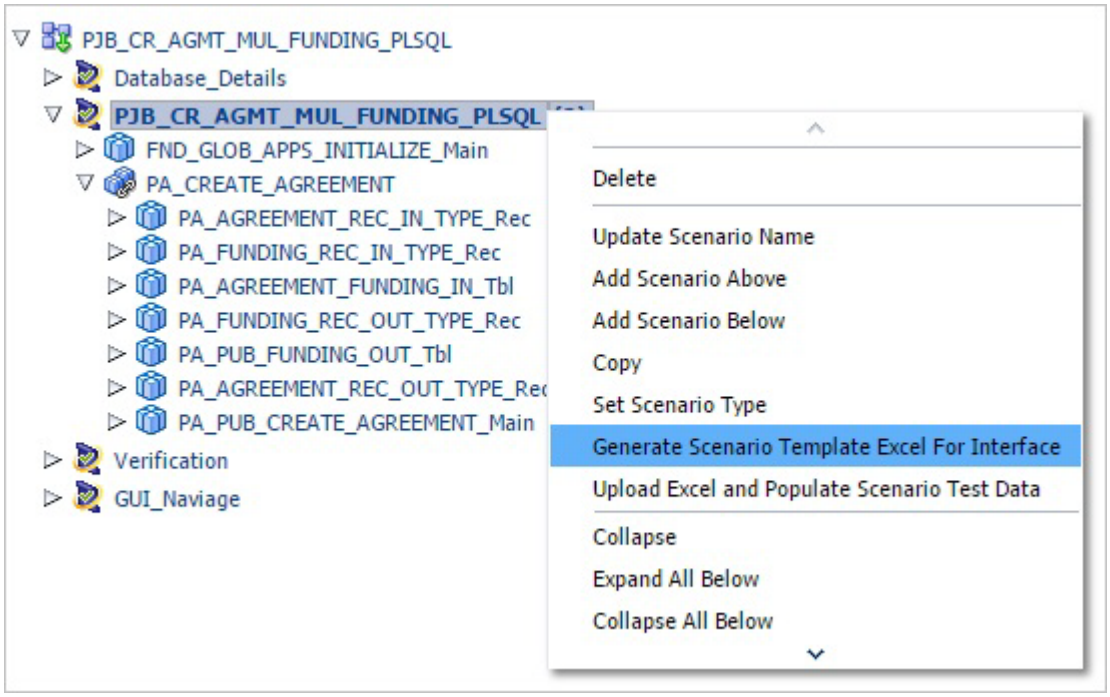
This can be achieved in an Oracle Flow Builder Flow by just providing data as "INPUT" to "START RECORD TYPE" or "START TABLE TYPE" field and associate the record or table component variable with the corresponding field in the procedure component.

5.9.1.2.2 PLSQL Flows with Multiple Sets of Test Data

Multiple Test Data for PLSQL flows should be specified in the Excel spreadsheet and a separate Excel spreadsheet should be generated for each procedure Scenario.

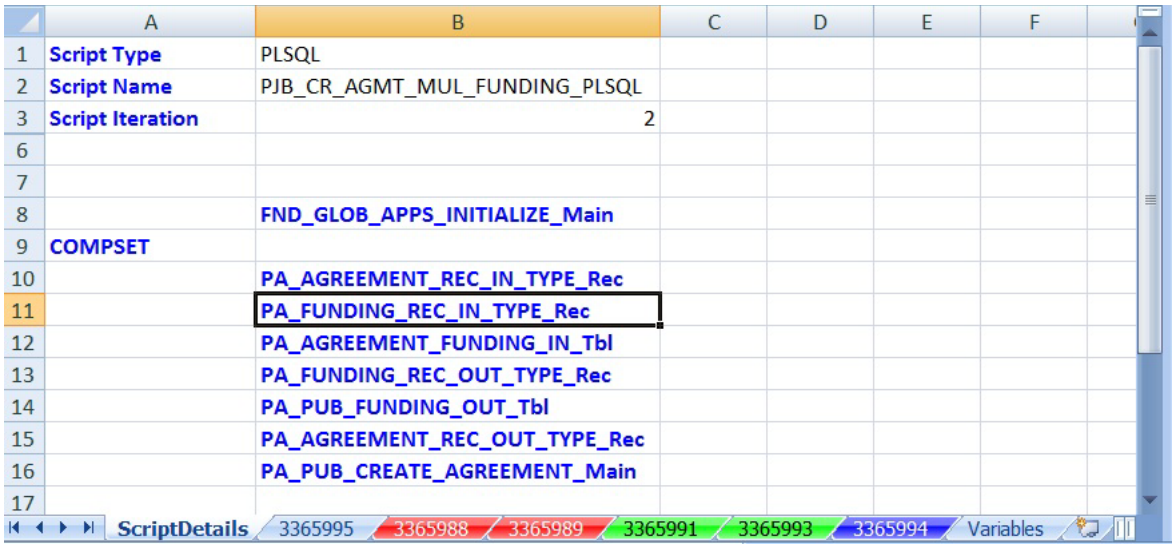
1. Right-Click on the Scenario name in the Flow Structure pane and select **Generate Scenario Template Excel For Interface**. Oracle Flow Builder will generate the Multiple Test Data Excel spreadsheet file with a .xlsm extension for the specified procedure. The Excel spreadsheet file will be downloaded as *scenarioname*.xlsm.

Figure 5–51 Generate Scenario Template Excel for Interface Menu Option



The multiple PLSQL Excel spreadsheet file will have the following details.

Figure 5–52 Example Multiple PLSQL Excel Spreadsheet



	A	B	C	D	E	F
1	Script Type	PLSQL				
2	Script Name	PJB_CR_AGMT_MUL_FUNDING_PLSQL				
3	Script Iteration	2				
6						
7						
8		FND_GLOB_APPS_INITIALIZE_Main				
9	COMPSET					
10		PA_AGREEMENT_REC_IN_TYPE_Rec				
11		PA_FUNDING_REC_IN_TYPE_Rec				
12		PA_AGREEMENT_FUNDING_IN_Tbl				
13		PA_FUNDING_REC_OUT_TYPE_Rec				
14		PA_PUB_FUNDING_OUT_Tbl				
15		PA_AGREEMENT_REC_OUT_TYPE_Rec				
16		PA_PUB_CREATE_AGREEMENT_Main				
17						

ScriptDetails | 3365995 | 3365988 | 3365989 | 3365991 | 3365993 | 3365994 | Variables

- **Script Details** sheet - this sheet will have details about the Scenario Name, Scenario Type, Scenario Iterations, and Scenario Structure.
  - Scenario Iteration indicates the number of times scenario needs to be executed.
  - Scenario structure will have the hyperlinks to navigate to the corresponding sheet in the Excel spreadsheet.
  - Components in the component structure will be mapped to the Excel sheets as follows:

    Each Record/Object will have one sheet.

    Table of Records/Table of Objects will be comprised of two components and those two components together will have one sheet. These two components are available in the corresponding sheet.

    Table of Standard types will have one sheet.

    One sheet for the Procedure Component and will be highlighted with blue color in the Excel spreadsheet.
- **Variables** sheet - this sheet includes all the variables from the first scenario to the current scenario.

Open the Excel file and enable macros.

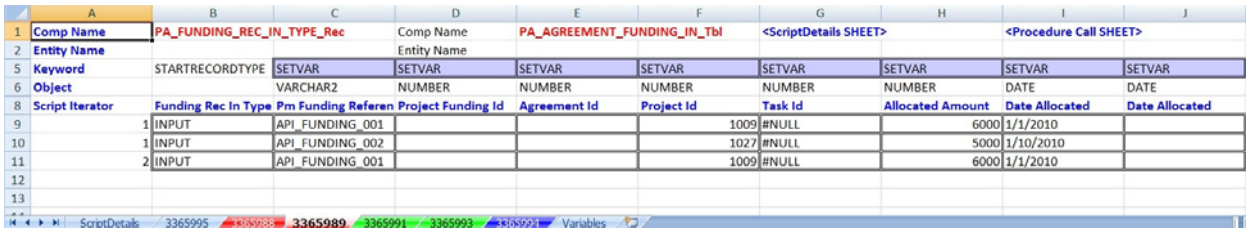
Specify the following in the Script Details sheet:

- Enter number of iterations in "Script Iteration" field to specify number of times the scenario need to be executed.

Test Data for Table of Records/Table of Objects As Input

Specifying Test Data for Table of Records/Table of Objects which are passed as input to the procedure requires the following details:

Figure 5–53 Example Test Data for Table of Records/Table of Objects as Input



	A	B	C	D	E	F	G	H	I	J
1	Comp Name	PA_FUNDING_REC_IN_TYPE_Rec		Comp Name	PA_AGREEMENT_FUNDING_IN_Tbl		<ScriptDetails SHEET>		<Procedure Call SHEET>	
2	Entity Name			Entity Name						
5	Keyword	STARTRECORDTYPE	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR
6	Object		VARCHAR2	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	DATE	DATE
8	Script Iterator	Funding Rec In Type	Pm Funding Referen	Project Funding Id	Agreement Id	Project Id	Task Id	Allocated Amount	Date Allocated	Date Allocated
9	1	INPUT	API_FUNDING_001			1009	#NULL	6000	1/1/2010	
10	1	INPUT	API_FUNDING_002			1027	#NULL	5000	1/10/2010	
11	2	INPUT	API_FUNDING_001			1009	#NULL	6000	1/1/2010	
12										
13										

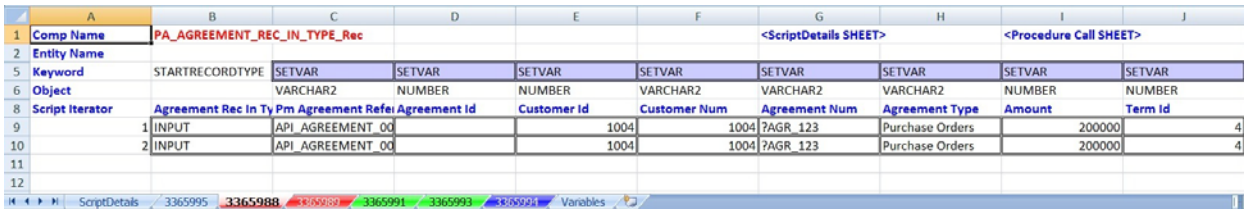
ScriptDetails 3365995 3365989 3365991 3365993 Variables

- Table of Records/Objects corresponds to two components in Oracle Flow Builder, as follows:
  - Record/Object component
  - Table component
- These two components become one sheet in the Excel spreadsheet. The component names will be displayed in the top header of the sheet.
- Script Iterator indicates data for the specific iteration.
- If you wish to pass multiple rows to the table, then provide all the rows with same script Iterator. For example:
  - If you want to pass two rows to the table during first iteration, then provide "1" for those two rows of data.
  - If you want to pass a single row to the table during second iteration, then provide "2" for the single row of data.

Test Data for Records/Objects as Input

Specifying Test Data for Records/Objects which are passed as input to the procedure requires the following detail:

Figure 5–54 Example Test Data for Records/Objects as Input



	A	B	C	D	E	F	G	H	I	J
1	Comp Name	PA_AGREEMENT_REC_IN_TYPE_Rec					<ScriptDetails SHEET>		<Procedure Call SHEET>	
2	Entity Name									
5	Keyword	STARTRECORDTYPE	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR
6	Object		VARCHAR2	NUMBER	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	NUMBER	NUMBER
8	Script Iterator	Agreement Rec In Ty	Pm Agreement Refe	Agreement Id	Customer Id	Customer Num	Agreement Num	Agreement Type	Amount	Term Id
9	1	INPUT	API_AGREEMENT_00		1004	1004	?AGR_123	Purchase Orders	200000	4
10	2	INPUT	API_AGREEMENT_00		1004	1004	?AGR_123	Purchase Orders	200000	4
11										
12										

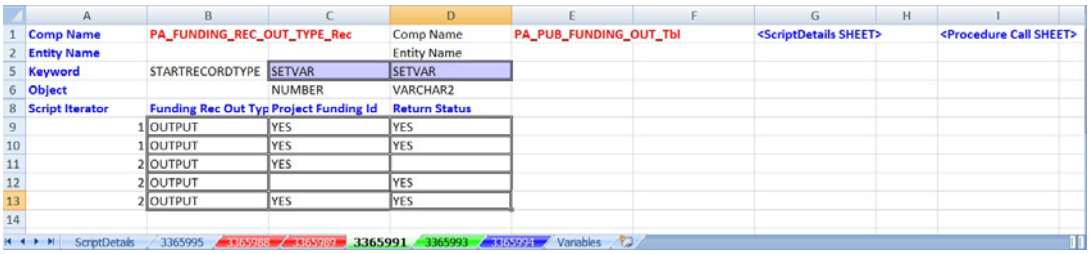
ScriptDetails 3365995 3365988 3365991 3365993 Variables

- Record/Table Component will have one sheet in the Excel spreadsheet.
- Script Iterator indicates data for the specific iteration.
- Record/Object can have only one row of data for each iteration.
- Two rows of data can not have the same iterator.
- Specify the script iterator for each row of data.

Test Data for Table of Records/Table of Objects as Output

Specifying Test Data for Table of Records/Table of Objects which are returned as output from the procedure requires the following details:

Figure 5–55 Example Test Data for Table of Records/Table of Objects as Output



	A	B	C	D	E	F	G	H	I
1	Comp Name	PA_FUNDING_REC_OUT_TYPE_Rec		Comp Name	PA_PUB_FUNDING_OUT_Tbl		<ScriptDetails SHEET>		<Procedure Call SHEET>
2	Entity Name			Entity Name					
5	Keyword	STARTRECORDTYPE	SETVAR	SETVAR					
6	Object		NUMBER	VARCHAR2					
8	Script Iterator	Funding Rec Out Typ	Project Funding Id	Return Status					
9	1	OUTPUT	YES	YES					
10	1	OUTPUT	YES	YES					
11	2	OUTPUT	YES						
12	2	OUTPUT		YES					
13	2	OUTPUT	YES	YES					
14									

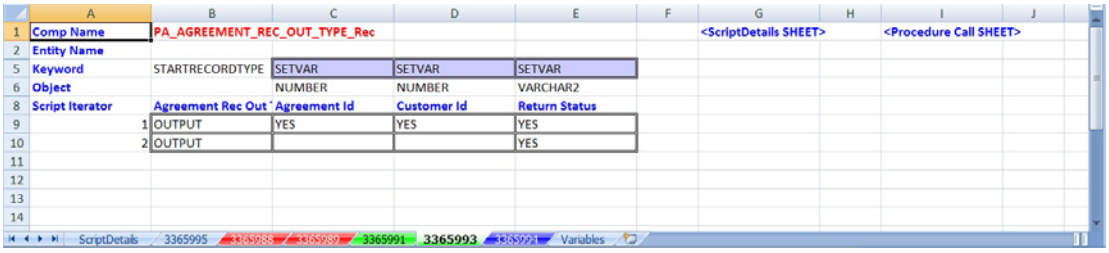
ScriptDetails | 3365995 | 3365991 | 3365993 | Variables

- Table of Records/Objects corresponds to two components in Oracle Flow Builder:
  - Record/Object component
  - Table component
- These two components becomes one sheet in the Excel spreadsheet. The component names will be displayed in the top header of the sheet.
- Script Iterator indicates data for the specific iteration.
- If the table is returning multiple rows as output, then provide all the rows with same script Iterator and "YES" for the fields which needs to be returned as output.

Test Data for Records/Objects as Output

Specifying Test Data for Records/Objects which are returned as output from the procedure requires the following detail:

Figure 5–56 Example Test Data for Records/Objects as Output



	A	B	C	D	E	F	G	H	I	J
1	Comp Name	PA_AGREEMENT_REC_OUT_TYPE_Rec					<ScriptDetails SHEET>		<Procedure Call SHEET>	
2	Entity Name									
5	Keyword	STARTRECORDTYPE	SETVAR	SETVAR	SETVAR					
6	Object		NUMBER	NUMBER	VARCHAR2					
8	Script Iterator	Agreement Rec Out	Agreement Id	Customer Id	Return Status					
9	1	OUTPUT	YES	YES	YES					
10	2	OUTPUT			YES					
11										
12										
13										
14										

ScriptDetails | 3365995 | 3365991 | 3365993 | Variables

- Script Iterator indicates data for the specific iteration.
- If the record/object is returning as output, then provide specific row with Iterator and "YES" for the fields which needs to be returned as output.
- Two rows should not have same iterator.

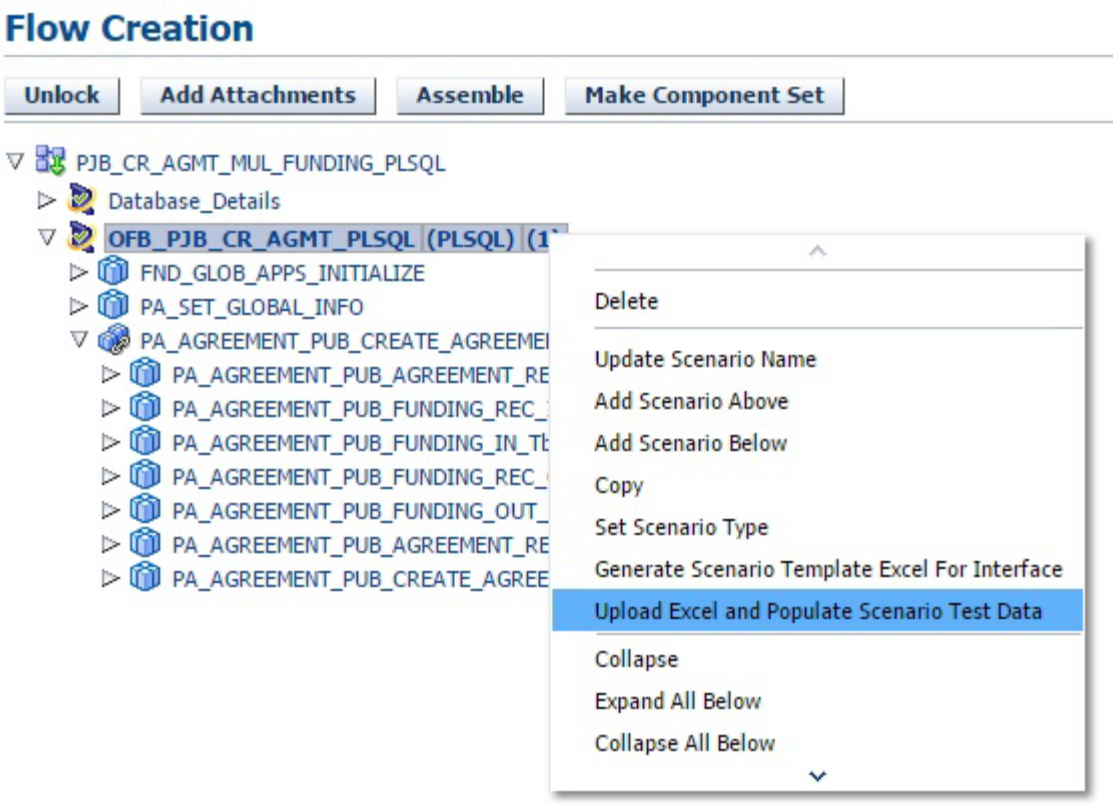
Test Data for User-Defined Components

Specify the data for all the user-defined components *except* the procedure component in the Multiple Test Data Excel spreadsheet and save the Excel spreadsheet.

Upload Excel and Populate Test Data

Right-click on the scenario name in the Flow creation pane and select **Upload Excel and Populate Scenario Test Data**.

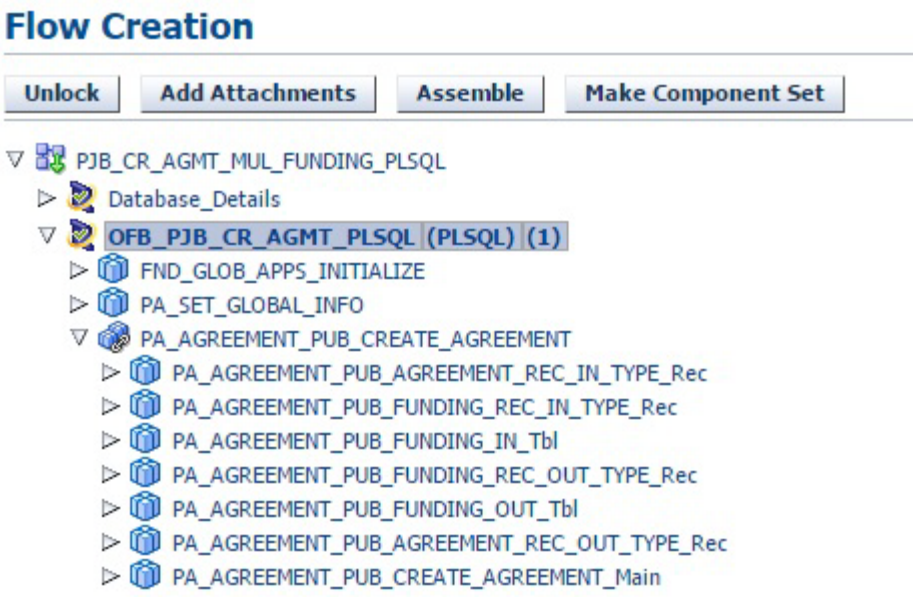
Figure 5–57 Upload Excel and Populate Scenario Test Data Menu option



On uploading the Excel spreadsheet, following set of the actions will be performed:

- The number of Iterations will be displayed next to the script name.
- Variables will be created for all the iterations specified in the excel based on the number of rows of data.
- Data will be stored in the Oracle Flow Builder application.

Figure 5–58 Example Flow After Uploading Excel Spreadsheet

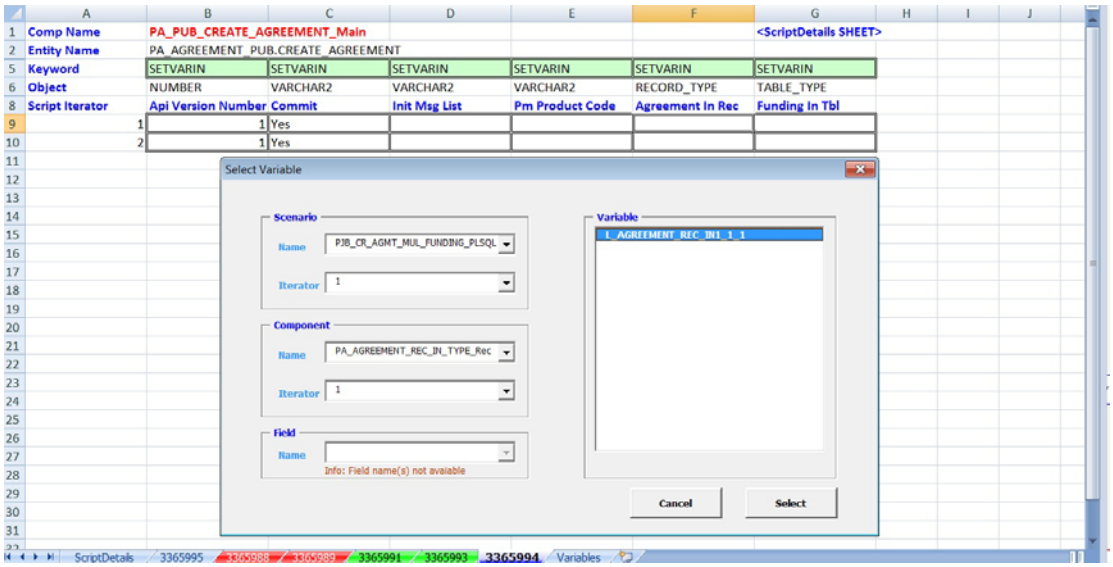


Test Data for Procedure Component in Excel

Specify the Test Data for the Procedure component in the Excel spreadsheet, as follows:

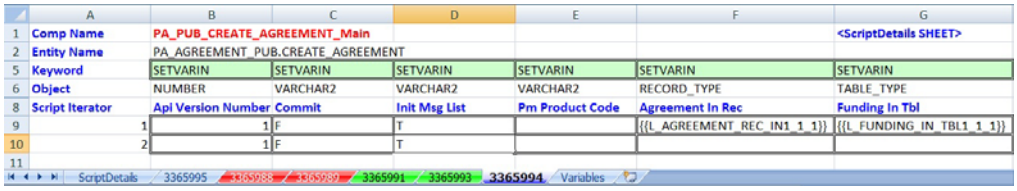
- Right-Click on the Scenario name in the Flow Structure pane and select **Generate Scenario Template Excel For Interface** to generate the Excel spreadsheet.
- Specify the Test Data for Standard Data types such as Varchar, Number, Boolean, etc, in the procedure component.
- Specify the Test Data for user defined data types of all iterations with a corresponding variable. Variables in the Excel spreadsheet can be specified as follows:
  - Enable Macros for the Excel spreadsheet.
  - Double-click on the cell where you need to provide the variable to open a dialog box for selecting variables.

Figure 5–59 Example of Variable Selection in Excel



– Provide details like Scenario name, Scenario iteration, component name, component iteration to filter exact variable. Select the variables from the variables pane.

Figure 5–60 Example Test Data For Procedure Component



- Upload the Excel spreadsheet into the scenario after providing data for the procedure component.

Test Data for Procedure Requirements

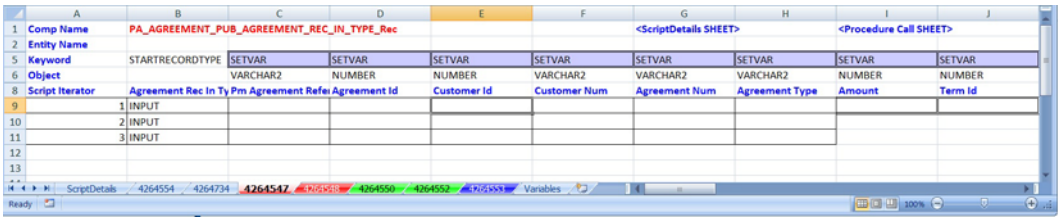
To pass user defined types like record/Table of Records to the procedure without providing any Test Data:

- Declare Record/Table of Records variable
- Pass the Record/Table of Records variable to the procedure.

This is performed in Oracle Flow Builder PLSQL Scenario with Multiple Test Data as follows:

1. Specify data as INPUT to START RECORD TYPE field in the Record/Table Sheet.

Figure 5–61 Example Test Data for Record Component



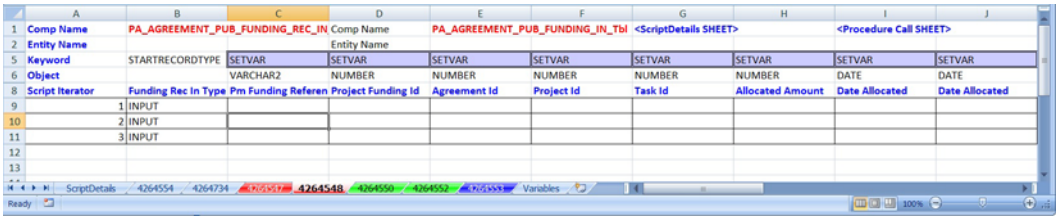
	A	B	C	D	E	F	G	H	I	J
1	Comp Name	PA_AGREEMENT_PUB_AGREEMENT_REC_IN_TYPE_Rec					<ScriptDetails SHEET>		<Procedure Call SHEET>	
2	Entity Name									
5	Keyword	STARTRECORDTYPE	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR
6	Object		VARCHAR2	NUMBER	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	NUMBER	NUMBER
8	Script Iterator	Agreement Rec In Ty	Pm Agreement Refe	Agreement Id	Customer Id	Customer Num	Agreement Num	Agreement Type	Amount	Term Id
9	1	INPUT								
10	2	INPUT								
11	3	INPUT								
12										
13										

ScriptDetails | 4264554 | 4264734 | 4264547 | 4264550 | 4264552 | Variables

Ready

2. Associate the record or table component variable with the corresponding field in the procedure component in Excel.

Figure 5–62 Example Test Data for Table of Records Component



	A	B	C	D	E	F	G	H	I	J
1	Comp Name	PA_AGREEMENT_PUB_FUNDING_REC_IN		Comp Name	PA_AGREEMENT_PUB_FUNDING_IN_Tbl		<ScriptDetails SHEET>		<Procedure Call SHEET>	
2	Entity Name			Entity Name						
5	Keyword	STARTRECORDTYPE	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR	SETVAR
6	Object		VARCHAR2	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	DATE	DATE
8	Script Iterator	Funding Rec In Type	Pm Funding Referen	Project Funding Id	Agreement Id	Project Id	Task Id	Allocated Amount	Date Allocated	Date Allocated
9	1	INPUT								
10	2	INPUT								
11	3	INPUT								
12										
13										

ScriptDetails | 4264554 | 4264734 | 4264548 | 4264550 | 4264552 | Variables

Ready

5.9.2 Adding Open Interface Flows

Oracle Flow Builder supports creation of flows for execution of Open Interfaces with Multiple Test Data. Oracle Flow Builder Open Interface flow can be created as follows:

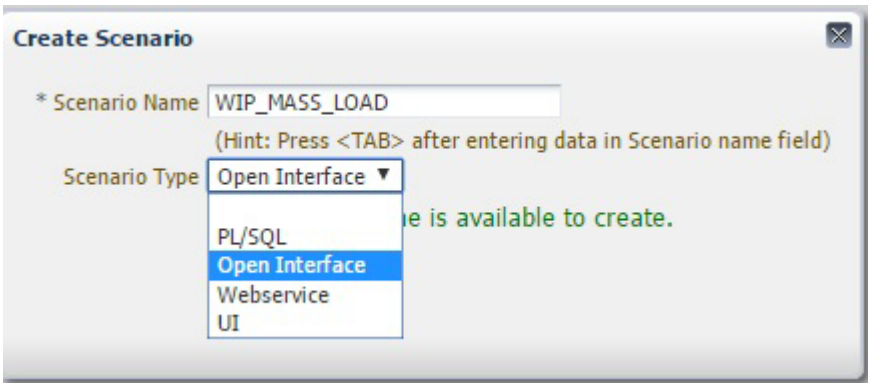
- Oracle Flow Builder scenario to provide database details.
- Oracle Flow Builder scenario to insert data into interface tables and execute one Open interface concurrent request.
- A separate scenario is to be created after the Open Interface Concurrent program execution for verification of data and to retrieve any data from the database using queries. The following components are required for verification and select query:
  - Verification Components - for Verification of data
  - Select Components - to retrieve data from the database

5.9.2.1 Open Interface Flow Scenario Structure

Create Scenario by right click on the flow and select **Add New Scenario**. Provide the following details in the Create Scenario dialog box as follows:

- Scenario Name
- Scenario Type as "Open Interface"

Figure 5–63 Create Scenario Details: Open Interface



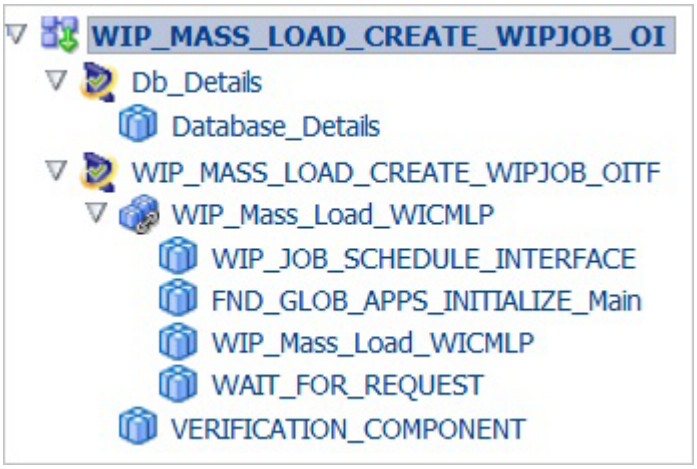
Scenario Type will be displayed next to the scenario names as "(OI)".

An Open Interface scenario should have following structure:

- A Component Set corresponds to a specific concurrent request. The Component Set should have the following:
  - Components to insert data into Interface database tables.
  - PLSQL procedure component to execute concurrent request.
- Drag any components such as FND_GLOB_APPS_INITIALIZE, etc. above the "Concurrent Request" component based on requirement.
- Drag "WAIT_FOR_REQUEST" component below the "Concurrent Request" component to wait till the concurrent request is completed.

The following figure shows an example of the Open Interface Flow structure:

Figure 5–64 Example Open Interface Flow Structure

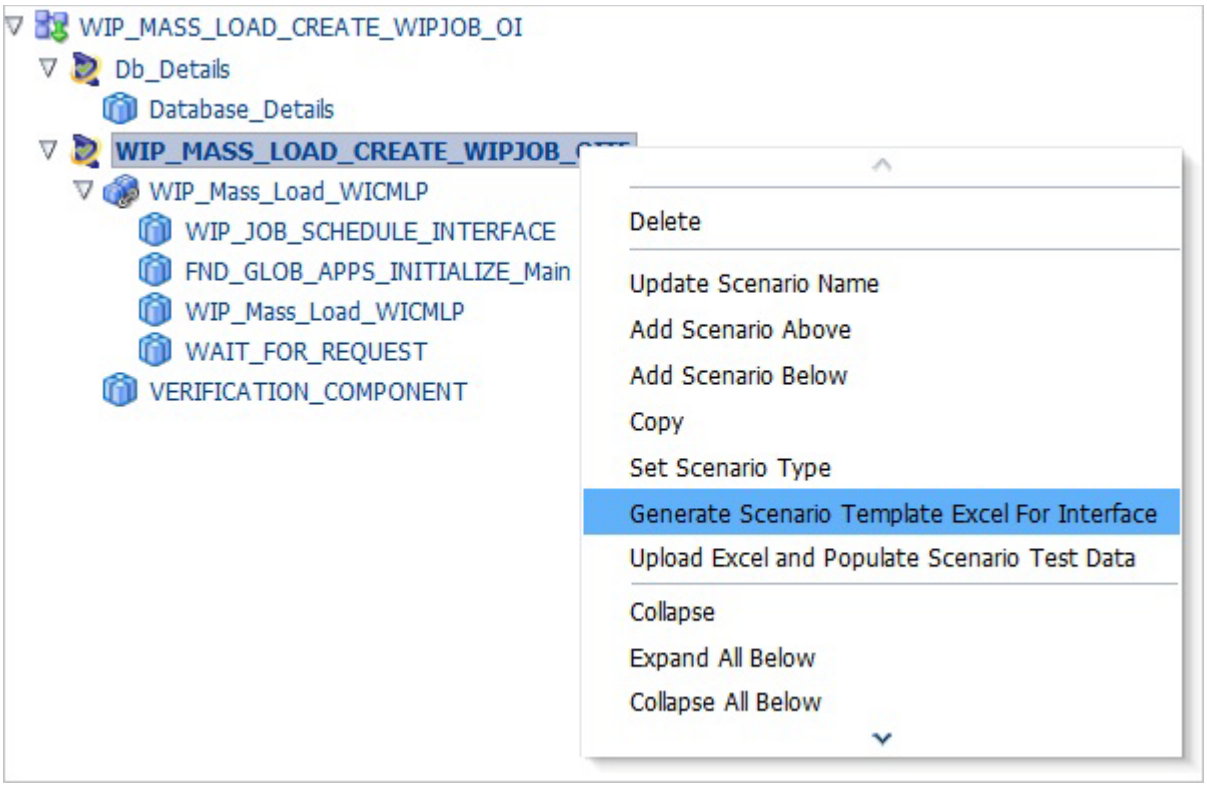


5.9.2.2 Test Data for Open Interface Scenario

Test Data for Open Interface flows should be specified in the Excel spreadsheet. To specify Test Data for Open Interface flows, do the following:

1. Right-click the Scenario name in the Flow Structure pane and select **Generate Scenario Template Excel For Interface**.

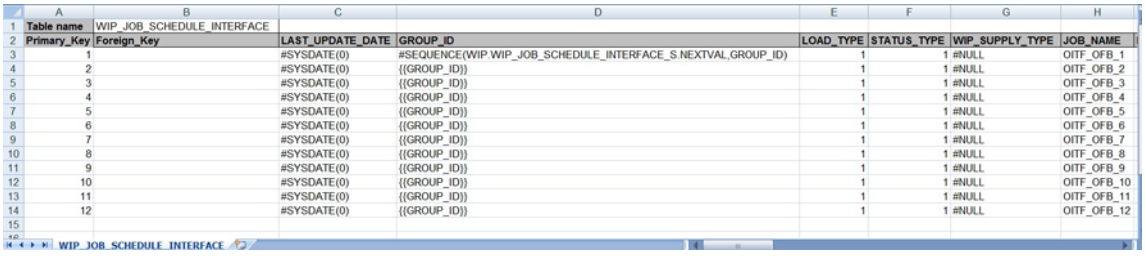
Figure 5–65 Generate Scenario Template Excel For Interface Menu Option



Selecting this menu option does the following:

- Generates the Excel spreadsheet for all the interface tables.
- Each sheet corresponds to one interface table.
- Each sheet has the following details:
  - Database table Name
  - Primary_Key & Foreign_Key
  - Database table columns

Figure 5–66 Example Excel Spreadsheet for Open Interface Flow Test Data



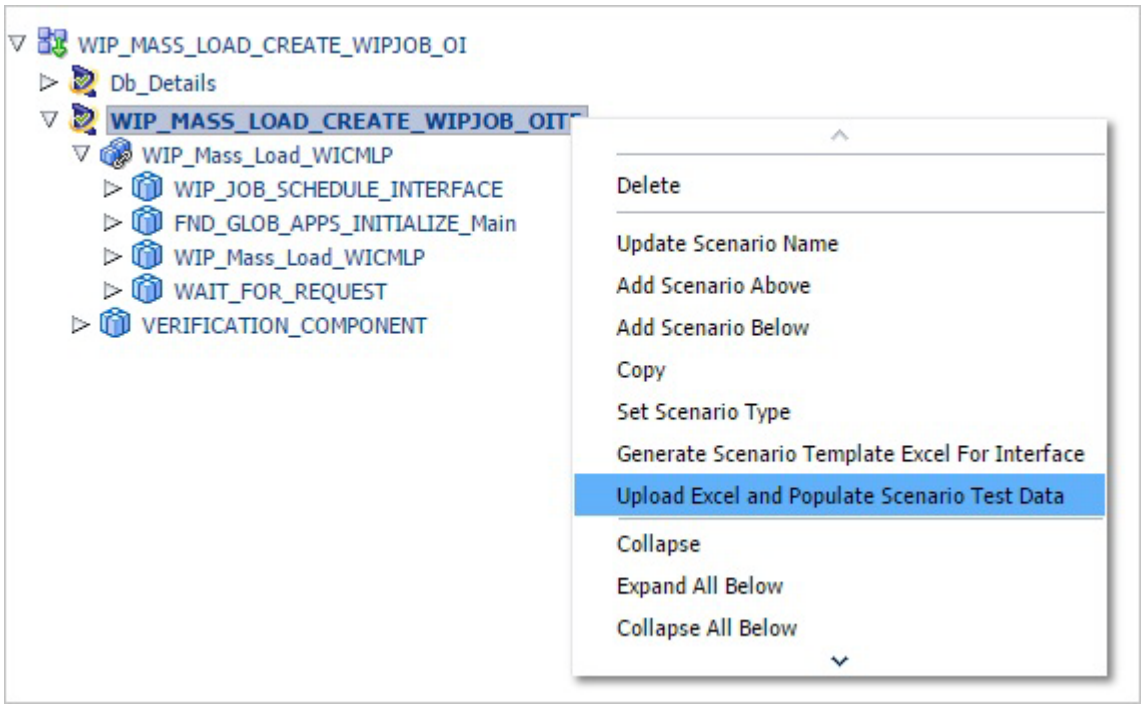
	A	B	C	D	E	F	G	H
1	Table name	WIP_JOB_SCHEDULE_INTERFACE						
2	Primary_Key	Foreign_Key	LAST_UPDATE_DATE	GROUP_ID	LOAD_TYPE	STATUS_TYPE	WIP_SUPPLY_TYPE	JOB_NAME
3	1		#SYSDATE(0)	#SEQUENCE(WIP.WIP_JOB_SCHEDULE_INTERFACE_S.NEXTVAL,GROUP_ID)	1	1	#NULL	OITF_OFB_1
4	2		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_2
5	3		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_3
6	4		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_4
7	5		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_5
8	6		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_6
9	7		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_7
10	8		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_8
11	9		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_9
12	10		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_10
13	11		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_11
14	12		#SYSDATE(0)	{{GROUP_ID}}	1	1	#NULL	OITF_OFB_12
15								

WIP_JOB_SCHEDULE_INTERFACE

2. Specify the data for interface tables. Primary_Key & Foreign_Key columns are used for user reference. Primary_Key field is a mandatory field and should be provided in sequence.
3. Save the Excel spreadsheet.

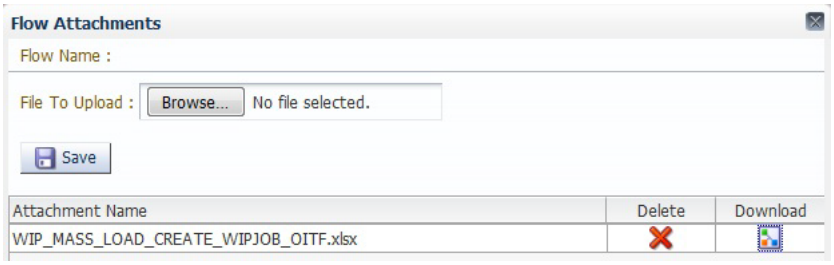
4. Right-click on the scenario name in the Flow creation pane and select **Upload Excel and Populate Scenario Test Data**.

Figure 5–67 Upload Excel and Populate Scenario Test Data Menu Option



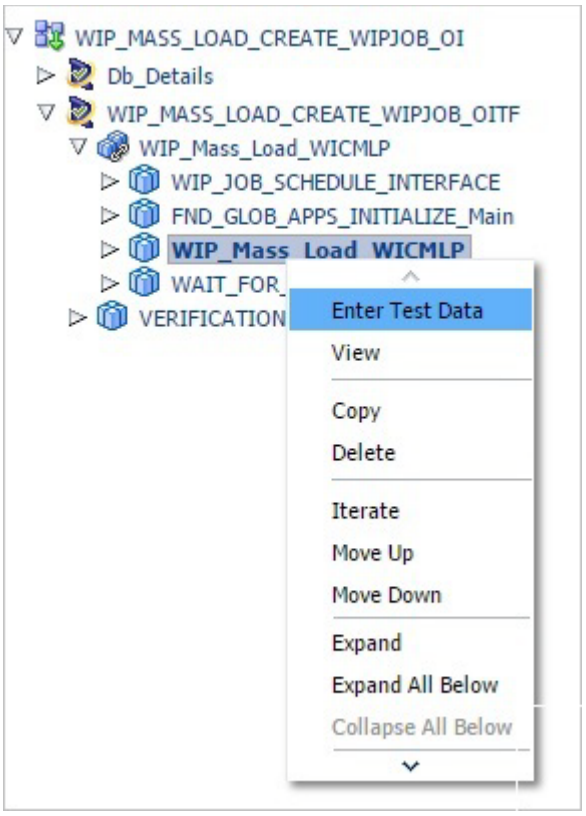
Upon uploading the Excel spreadsheet, the Excel file will be added to the Flow Attachments.

Figure 5–68 Excel File as Open Interface Flow Attachment



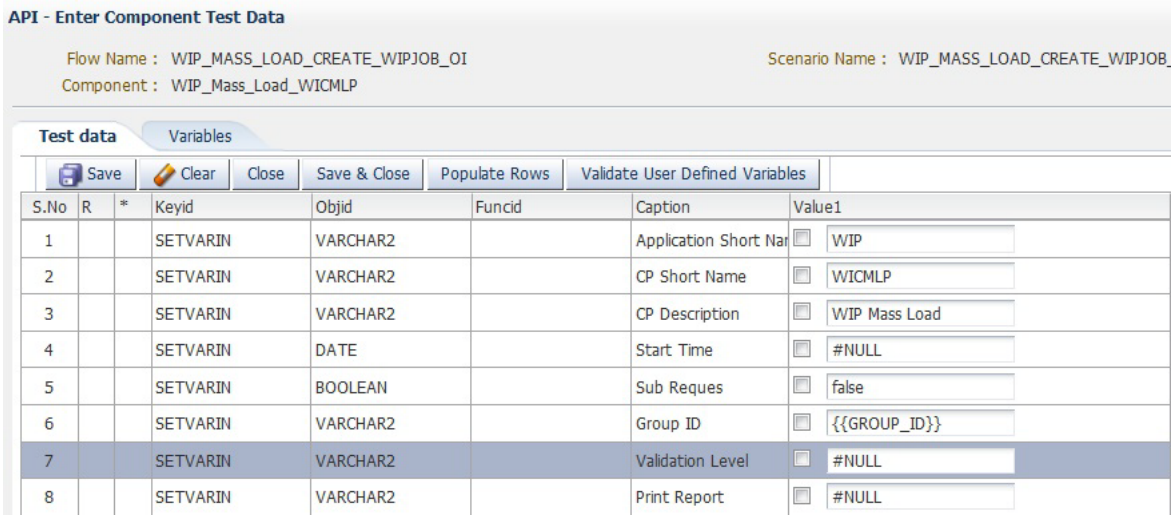
5. Right-click on the procedure component in the Open Interface scenario and select **Enter Test Data** from the menu.

Figure 5–69 Example Enter Test Data Menu Option for Open Interface Flow



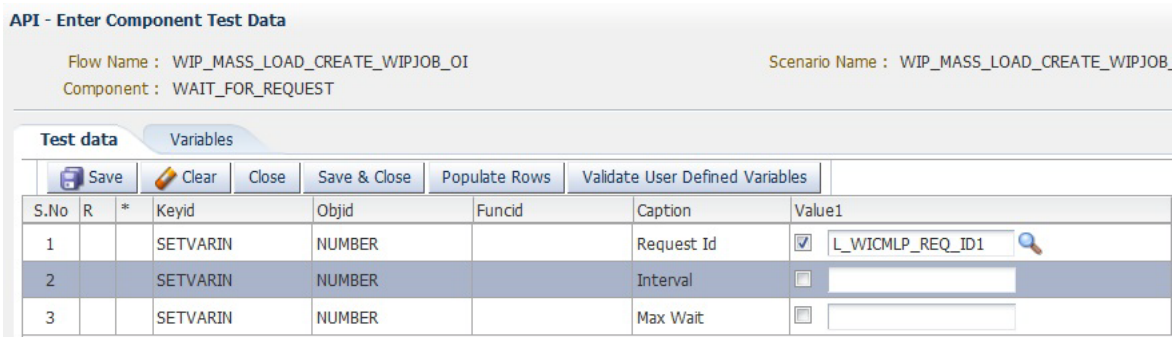
6. Specify the Test Data for the procedure as per the concurrent request.

Figure 5–70 Example Component Test Data for Concurrent Request



7. Specify the values for Request Id, Interval and Max Wait data field in Wait for Request component. This component is used to wait till the concurrent request is completed.

Figure 5–71 Example Component Test Data for Wait for Request



Notes:

The Open Interface template generated from the flow scenario always comes without Test Data.

The Open Interface template will be generated as *ScenarioName*.xlsx. Do not change the file name while uploading the Excel file to the flow.

5.9.2.3 Update Test Data for Open Interface

To update the Test Data for an Open Interface scenario:

1. Download the Excel spreadsheet that corresponds to the specified scenario from the Flow Attachments, as follows:
   - Click the **Add Attachments** button just above the flow name.
   - Download the Open Interface Excel spreadsheet.
2. Update the data in the Open Interface Excel spreadsheet.
3. Delete the Open Interface Excel spreadsheet in the flow attachments that corresponds to the scenario, as follows:
   - Click the **Add Attachments** button just above the flow name.
   - Delete the Open Interface Excel spreadsheet that corresponds to the scenario.
4. Upload the updated Open Interface Excel spreadsheet using the **Upload Excel and Populate Scenario Test Data** menu item at the scenario level.

5.9.2.4 Specifying Test Data for Open Interface

This section describes how to specify Test Data for Open Interface scenarios.

- All database table fields should be provided as String.
- Date values can be specified as #SYSDATE(+n). For example:
  - #SYSDATE(0) - Today's date
  - #SYSDATE(+1) - Tomorrow's date
  - #SYSDATE(-1) - Yesterday's date
- Null value can be specified as #NULL.
- Empty ("") value can be specified as #EMPTY.
- Space (" ") value can be specified as #SPACE.

- Rerunnable value can be specified using "?". "?" can be used anywhere in the existing data.
- A unique number with a database sequence can be specified as #SEQUENCE(*SequenceName*.NEXTVAL). For example:

  ```
  #SEQUENCE(WIP.WIP_INTERFACE_S.NEXTVAL)
  ```

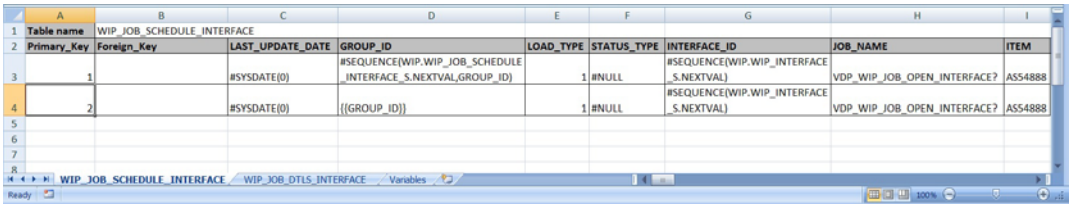
- A unique number with a database sequence and store the runtime database sequence in a variable for further usage in the same Excel template or another scenarios can be specified as #SEQUENCE(*SequenceName*.NEXTVAL, *VariableName*). For example:

  ```
  #SEQUENCE(WIP.WIP_JOB_SCHEDULE_INTERFACE_S.NEXTVAL,GROUP_ID)
  ```

  Variable Name can be used as {{*Varaible Name*}} in the same Excel spreadsheet (any sheet) or in the subsequent scenarios.

The following image shows an example of Open Interface Test Data:

Figure 5–72 Example Open Interface Test Data



	A	B	C	D	E	F	G	H	I
1	Table name	WIP_JOB_SCHEDULE_INTERFACE							
2	Primary_Key	Foreign_Key	LAST_UPDATE_DATE	GROUP_ID	LOAD_TYPE	STATUS_TYPE	INTERFACE_ID	JOB_NAME	ITEM
3	1		#SYSDATE(0)	#SEQUENCE(WIP.WIP_JOB_SCHEDULE_INTERFACE_S.NEXTVAL,GROUP_ID)	1	#NULL	#SEQUENCE(WIP.WIP_INTERFACE_S.NEXTVAL)	VDP_WIP_JOB_OPEN_INTERFACE?	AS54888
4	2		#SYSDATE(0)	{{GROUP_ID}}	1	#NULL	#SEQUENCE(WIP.WIP_INTERFACE_S.NEXTVAL)	VDP_WIP_JOB_OPEN_INTERFACE?	AS54888
5									
6									
7									
8									

5.9.3 Adding Webservice Flows

Oracle Flow Builder supports creation of flow for web services execution with multiple sets of Test Data and supports verification of response data. A Webservice flow can be created as follows:

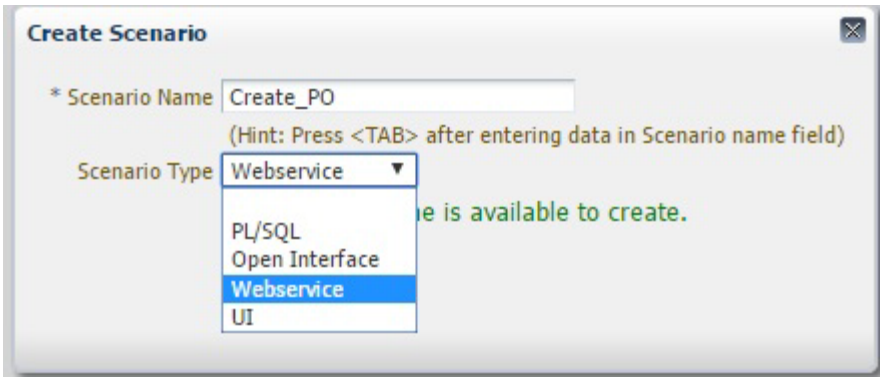
- A flow scenario to provide hostname and port details.
- Each Webservice payload and corresponding response verification as one scenario.

5.9.3.1 Webservice Flow Scenario Structure

Create a Scenario by right clicking on the flow and selecting **Add New Scenario**. Provide the following details in the Create Scenario dialog box, as follows:

- Scenario Name
- Scenario Type as "Webservice"

Figure 5–73 Create Scenario Details: Webservice



Scenario Type will be displayed next to the scenario names as "(WS)".

The Webservice scenario should have following structure:

- Add Component Sets related to the Webservice payload. The component set should have the following:
  - A component for WSDL and payload root structure details.
  - Components for Payload nodes having elements.
- Add Components related to Webservice response based on the requirements. Add all the output response components to the scenario which are required for the following:
  - Verification.
  - To retrieve the value from a response for further usage in the scenarios.

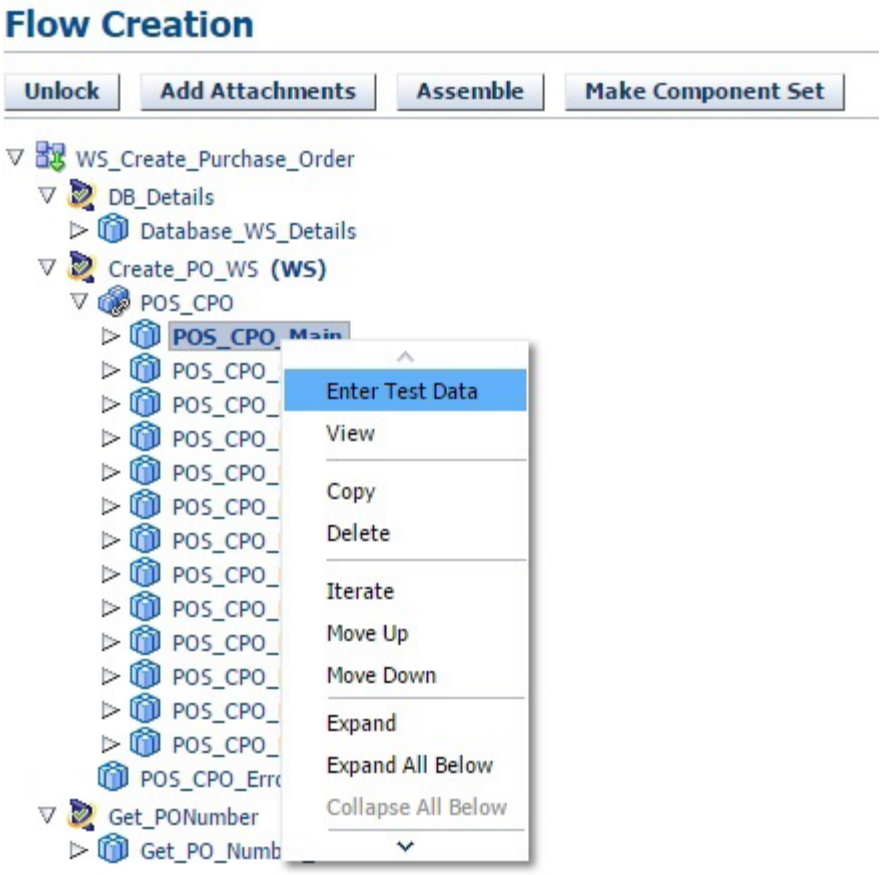
Figure 5–74 Example Webservice Flow Structure

5.9.3.2 Specifying Test Data for Webservice Flows

Specify the Test Data for Webservice flows as follows:

1. Right-click on a Payload structure component in the Webservice scenario and select **Enter Test Data** from the menu.

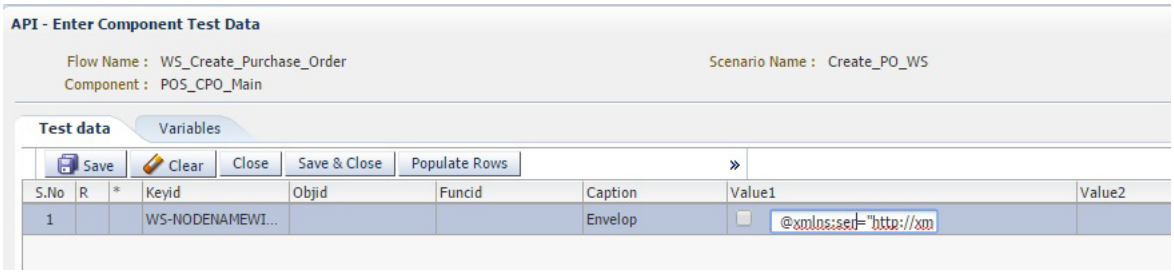
Figure 5–75 Example Enter Test Data Menu Option for Webservice Flow



2. Specify the Test Data for the envelop (specify attributes of the root node for the Envelop field). For example:

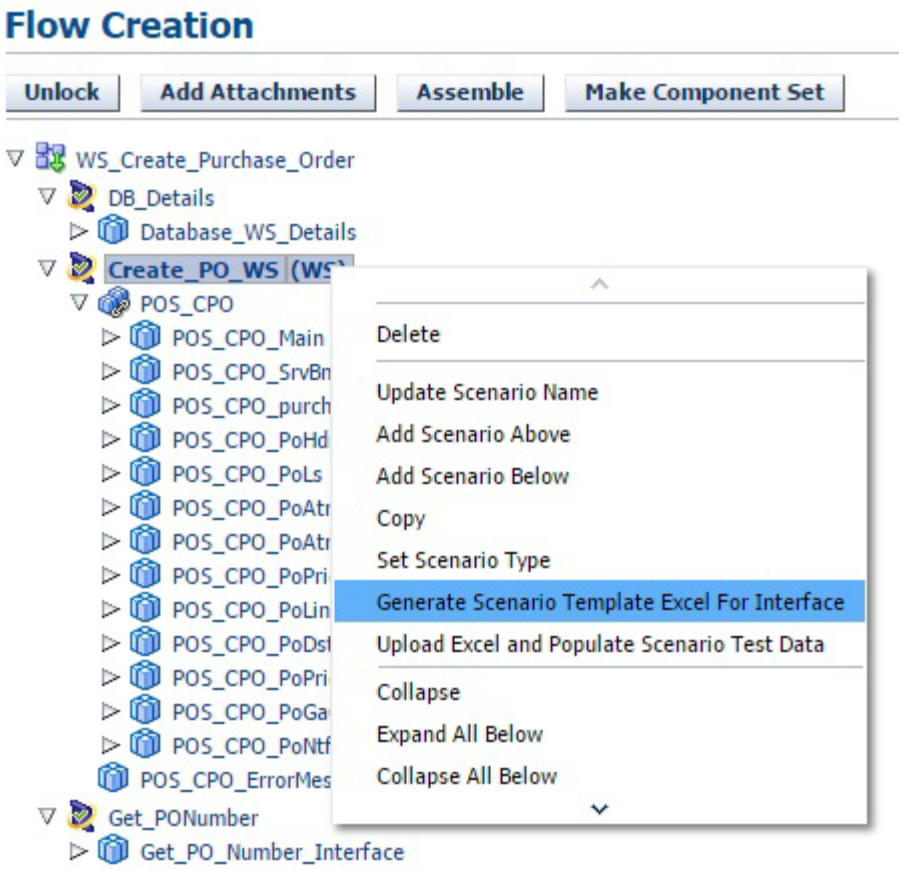
```
@xmlns:ser="http://xmlns.oracle.com/apps/fnd/ServiceBean",
@xmlns:ser1="http://xmlns.oracle.com/apps/po/service",
@xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

Figure 5–76 Example Test Data for Payload Component



3. Save and close.
4. Right-click on the Scenario name in the Flow Structure Webservice scenario pane and select **Generate Scenario Template Excel For Interface**.

Figure 5–77 Generate Scenario Template Excel for Interface Menu Option



Generating the Scenario Template Excel does the following:

- Generates an Excel spreadsheet for all the Webservice payload node components and WebService Response Components. Each "node component" in the scenario corresponds to one sheet in the Excel spreadsheet.

The Webservice Excel spreadsheet file will have the following details:

- **Main** sheet - this sheet will have details about the number of times payload needs to be executed.
  - Payload_Key indicates the number of times payload needs to be executed.
  - Primary_Key indicates unique number in that sheet.
  - Test Case Name indicates the purpose of the payload. This must always be provided for each payload.
  - Username and Password are used for Webservice security authentication. These values should be provided for each payload iteration.
  - AllowEmptyNode indicates inclusion of empty nodes (Nodes which doesn't have any data) in the generate payload. Provide data for this field as follows:

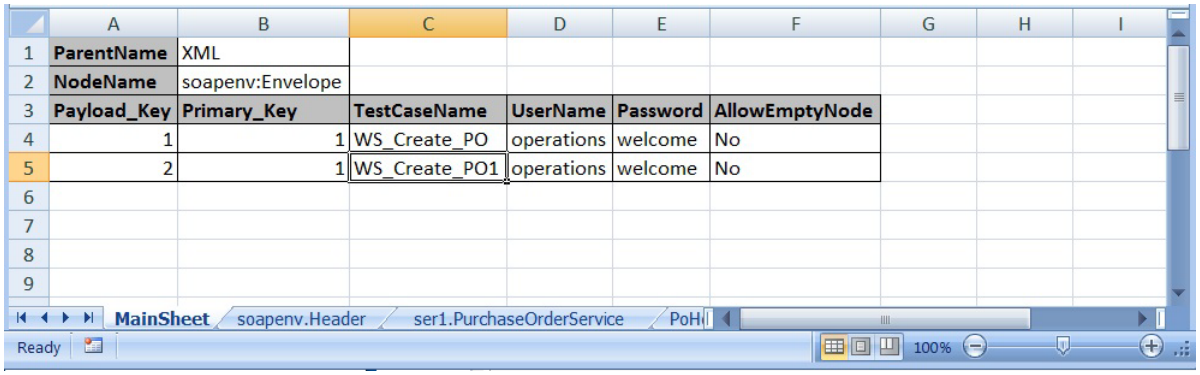
    YES - indicates Empty will be included in the payload even if data is not provided.

NO - indicates Empty nodes will be removed in the payload when data is not provided.

- Component sheets - each component from the Webservice scenario (payload & respose) will have one sheet in the Excel spreadsheet.
  - Each sheet will have "Parent Name" and "Node Name" of the payload node. These will come automatically into the Excel spreadsheet from the component for user reference.
- **Variables** sheet - this sheet will have all the variables from the first scenario to the current scenario.

5. Specify the following details in the Main sheet:

Figure 5–78 Example Main Sheet Test Data for Webservice Flow



	A	B	C	D	E	F
1	ParentName	XML				
2	NodeName	soapenv:Envelope				
3	Payload_Key	Primary_Key	TestCaseName	UserName	Password	AllowEmptyNode
4	1	1	WS_Create_PO	operations	welcome	No
5	2	1	WS_Create_PO1	operations	welcome	No

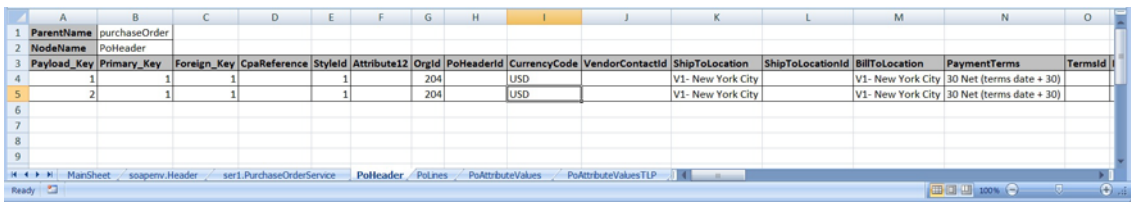
- Payload_key for each iteration of the payload.
- Provide Primary_Key corresponds to each payload.
- Provide TestCaseName corresponds to each payload. This must be provided for each payload.
- Provide UserName/Password, which are required as Webservice security authentication during payload execution.
- AllowEmptyNode indicates inclusion of empty nodes (Nodes which doesn't have any data) in the generate payload. Provide data for this field as follows:

  YES - indicates Empty will be included in the payload even if data is not provided.

  NO - indicates Empty nodes will be removed in the payload when data is not provided.

6. Specify the Payload data, as follows:

Figure 5–79 Example 1 PoHeader Sheet for Webservice Flow

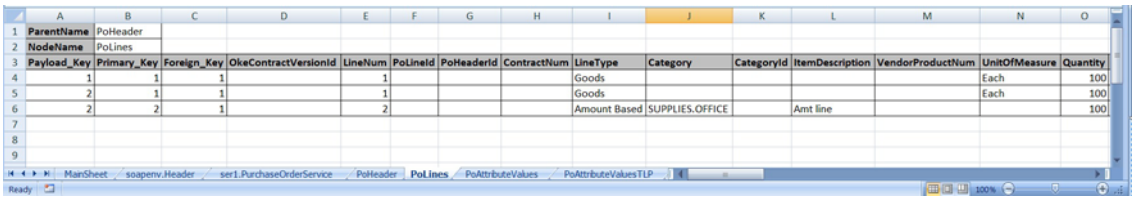


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	ParentName	purchaseOrder													
2	NodeName	PoHeader													
3	Payload_Key	Primary_Key	Foreign_Key	CpaReference	StyleId	Attribute12	OrgId	PoHeaderId	CurrencyCode	VendorContactId	ShipToLocation	ShipToLocationId	BillToLocation	PaymentTerms	TermsId
4	1	1	1		1		204		USD		V1- New York City		V1- New York City	30 Net (terms date + 30)	
5	2	1	1		1		204		USD		V1- New York City		V1- New York City	30 Net (terms date + 30)	

- Provide data for each node component with Payload_Key.

- Primary_Key and Foreign_Key to be provided based on node structure. These are to be provided corresponding to the payload.
- If the same node data is repeated multiple times, then provide multiple rows in the Test Data with the same Payload_Key.

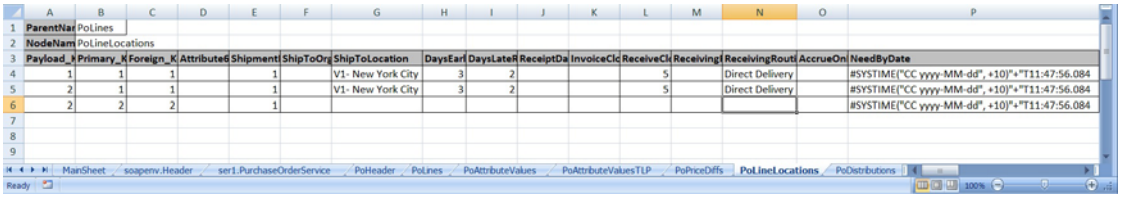
Figure 5–80 Example 2 PoLines Sheet for Webservice Flow



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	ParentName	PoHeader													
2	NodeName	PoLines													
3	Payload_Key	Primary_Key	Foreign_Key	OkeContractVersionId	LineNum	PoLineId	PoHeaderId	ContractNum	LineType	Category	CategoryId	ItemDescription	VendorProductNum	UnitOfMeasure	Quantity
4	1	1	1		1				Goods					Each	100
5	2	1	1		1				Goods					Each	100
6	2	2	1		2				Amount Based	SUPPLIES.OFFICE		Amt line			100
7															
8															
9															

MainSheet | soapenv.Header | ser1.PurchaseOrderService | PoHeader | PoLines | PoAttributeValues | PoAttributeValuesTLP

Figure 5–81 Example 3 PoLineLocations Sheet for Webservice Flow

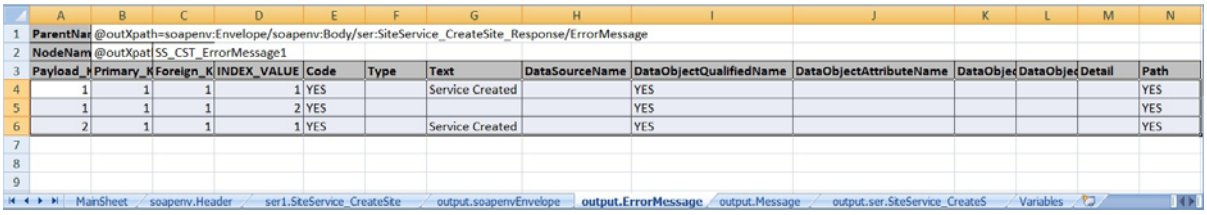


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	ParentNar	PoLines														
2	NodeNam	PoLineLocations														
3	Payload_K	Primary_K	Foreign_K	AttributeC	Shipment	ShipToOrg	ShipToLocation	DaysEarl	DaysLateR	ReceiptDa	InvoiceClc	ReceiveCl	ReceivingI	ReceivingRout	AccrueOn	NeedByDate
4	1	1	1		1		V1- New York City	3	2			5		Direct Delivery		#SYSTIME("CC yyyy-MM-dd", +10)"+"T11:47:56.084
5	2	1	1		1		V1- New York City	3	2			5		Direct Delivery		#SYSTIME("CC yyyy-MM-dd", +10)"+"T11:47:56.084
6	2	2	2		1											#SYSTIME("CC yyyy-MM-dd", +10)"+"T11:47:56.084
7																
8																
9																

MainSheet | soapenv.Header | ser1.PurchaseOrderService | PoHeader | PoLines | PoAttributeValues | PoAttributeValuesTLP | PoPriceDiffs | PoLineLocations | PoDistributions

7. Specify data to retrieve from payload response or verification of data in payload response.

Figure 5–82 Example Data to Retrieve/Verification of Data



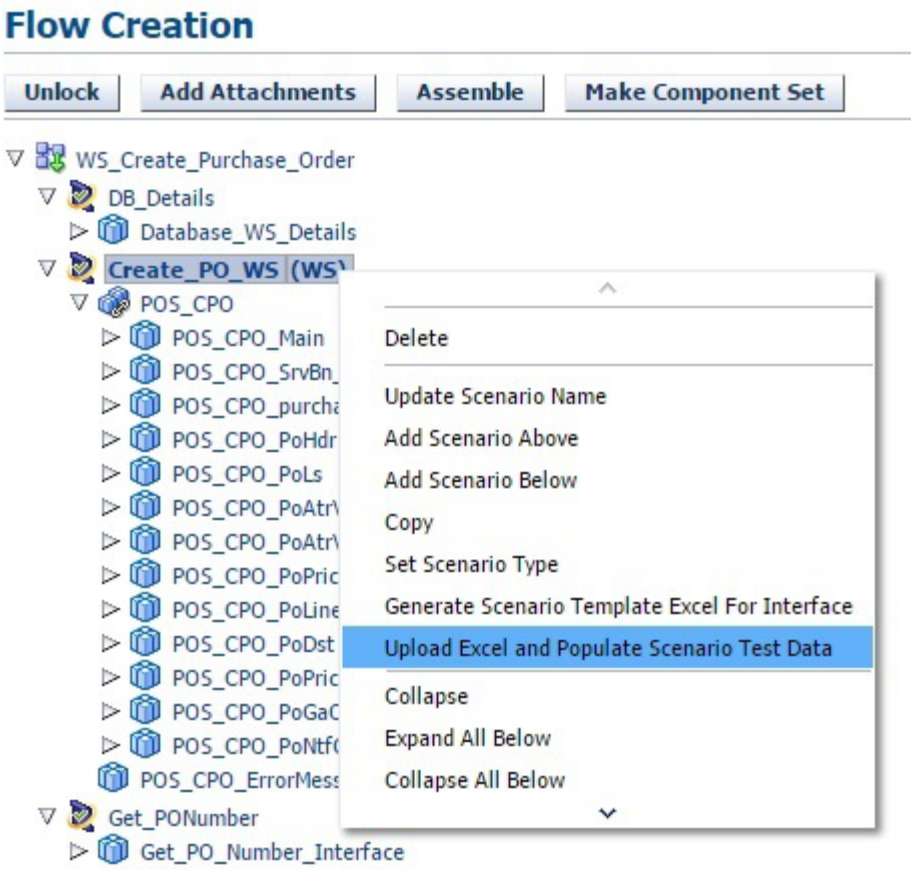
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ParentNar	@outXpath=soapenv:Envelope/soapenv:Body/ser:SiteService_CreateSite_Response/ErrorMessage												
2	NodeNam	@outXpat	SS_CST_ErrorMessage1											
3	Payload_K	Primary_K	Foreign_K	INDEX_VALUE	Code	Type	Text	DataSourceName	DataObjectQualifiedName	DataObjectAttributeName	DataObjec	DataObjec	Detail	Path
4	1	1	1	1	YES		Service Created		YES					YES
5	1	1	1	2	YES				YES					YES
6	2	1	1	1	YES		Service Created		YES					YES
7														
8														
9														

MainSheet | soapenv.Header | ser1.SiteService_CreateSite | output.soapenvEnvelope | output.ErrorMessage | output.Message | output.ser.SiteService_CreateS | Variables

- If user needs to verify any data for specific field from the output payload response, provide the data to be verified as Test Data in the corresponding field.
- If user needs to retrieve any data for specific field from the output payload response, then provide "YES" in the Test Data for the corresponding field.
- The column INDEX_VALUE in the output response components is used specify the index of the specific fields. This is to support repetition of the same node multiple times.

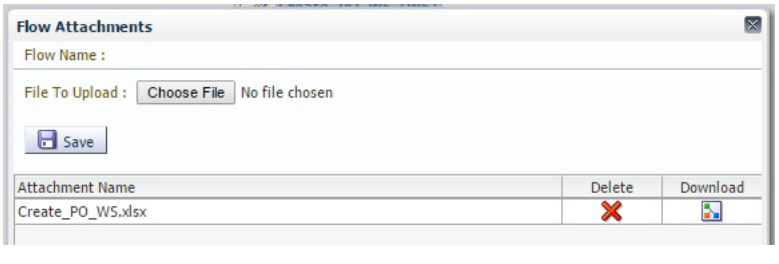
8. Save and close the Excel spreadsheet.
9. Upload the Excel spreadsheet by right-clicking on the Webservice scenario name in the Flow creation pane and select **Upload Excel and Populate Scenario Test Data**.

Figure 5–83 Upload Excel and Populate Scenario Test Data Menu Option



Upon uploading the Excel spreadsheet, the Excel file will be added to the Flow Attachments:

Figure 5–84 Excel File as Webservice Flow Attachment



5.9.3.3 Updating Test Data for Webservice Flows

To update Test Data for Webservice flows:

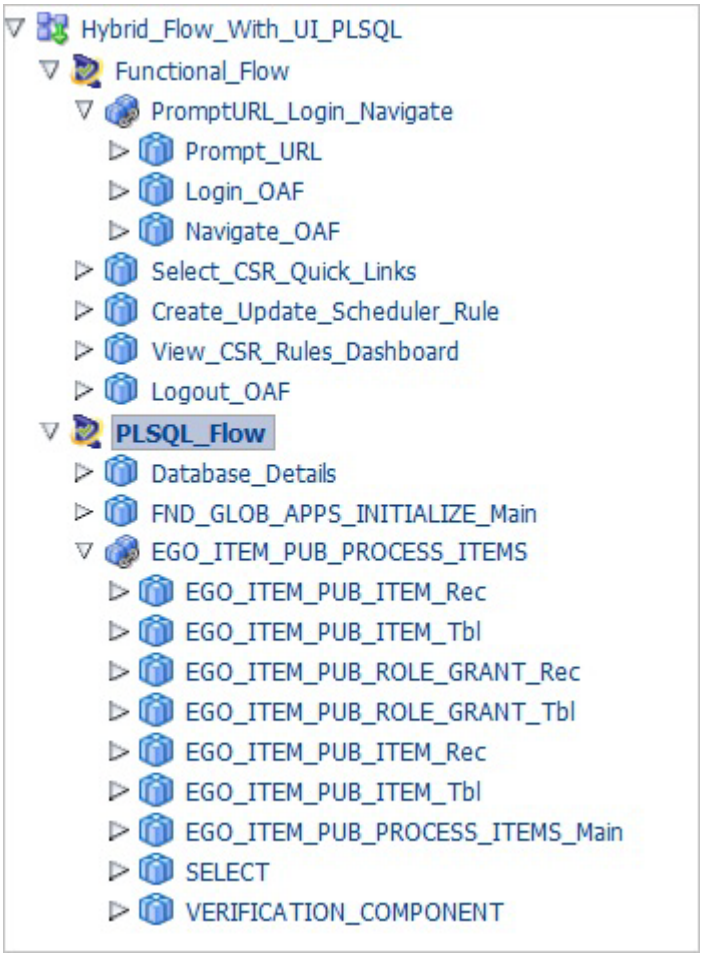
1. Download the Excel that corresponds to the specified scenario from the Flow Attachments, as follows:
   - Click **Add Attachments** just above the flow name.
   - Download the Webservice Excel spreadsheet file.
2. Update the data in the Webservice Excel spreadsheet file and save the file.

3. Delete the Webservice Excel spreadsheet file from the flow attachments that corresponds to the scenario, as follows:
   - Click **Add Attachments** just above the flow name.
   - Delete the Webservice Excel spreadsheet file that corresponds to the scenario.
4. Upload the updated Webservice Excel spreadsheet file using the **Upload Excel and Populate Scenario Test Data** menu item at the scenario level.

5.9.4 Adding Hybrid Flows

Oracle Flow Builder supports creation of flows with a combination of Functional Flow and PLSQL / Interface / Webservice flows.

Figure 5–85 Example Hybrid Flow



5.9.5 Best Practices

This section provides general guidelines on best practices and things to avoid when creating PLSQL flows.

5.9.5.1 Best Practice

General best practices are as follows:

1. Create a separate scenario to provide database details.

2. Provide the scenario type when creating any interface scenario. The scenario type is required to download the Multiple Test Data template for interfaces.
   - Data provided in the Excel spreadsheet should be in Text format.
3. Create separate scenario for verification and select components.
   - To validate data.
   - To retrieve any data from database using queries.
4. It is best to have a hybrid flow with a single set of Test Data.
5. Multiple Test Data is supported only for the following interfaces at the scenario level:
   - PLSQL
   - Open Interface
   - Webservice
6. Flow level Test Data Excel spreadsheet is used only for GUI flows.
   - You should use flow level Test Data Excel spreadsheet for GUI flows.
   - You should not use flow level Test Data Excel spreadsheet for interfaces like PLSQL, Open interface and Webservice with single or Multiple Test Data.
   - You should not use flow level Test Data Excel spreadsheet for hybrid flow.
7. When a flow is using Multiple Test Data scenario:
   - Enter the data for GUI scenario using the Oracle Flow Builder **Enter Test Data** option.
   - Do not download the Flow level Excel spreadsheet for entering Test Data.

5.9.5.2 Things to Avoid

Things to avoid are as follows:

1. Do not unlock the flow or logout from the Oracle Flow Builder when a Multiple Test Data template is downloaded for adding or updating data. Other users may modify the components.
2. Do not delete any rows in the PLSQL Multiple Test data Excel spreadsheet while updating data in Excel. Deletion of rows is not supported while updating data in PLSQL Multiple Test Data.
3. Do not include Verification and Select components in Multiple Test Data scenario.
4. Do not enter the Test Data using the Oracle Flow Builder **Enter Test Data** option for Multiple Test Data scenario components.

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**.

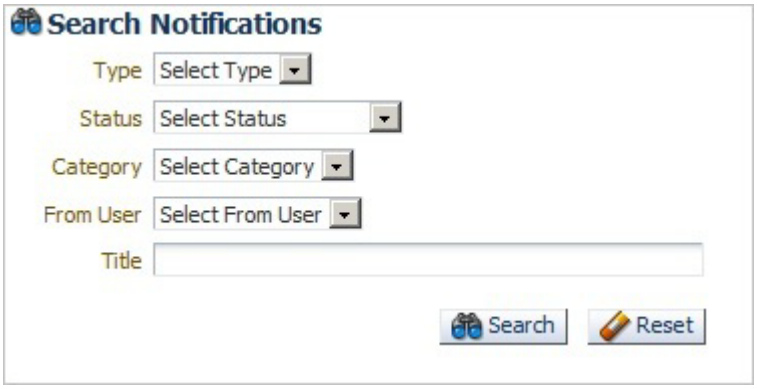
- To Do notifications: include notifications such as *Approved*, *Pending for Approval*, *Rejected*, and *Self Approved*. These notifications pertain to Components, User creation, or Product Family Access (PFAccess).
- FYI notifications: include notifications such as grant of Product Family Access, approval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

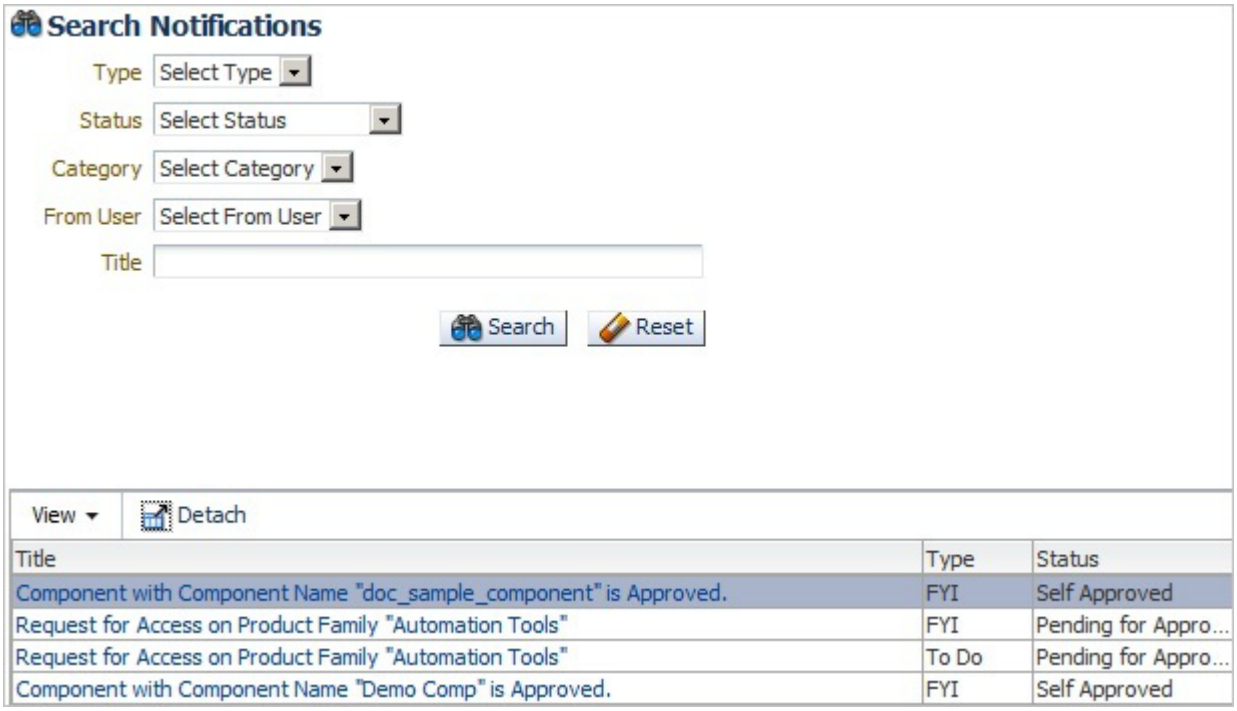
The Search pane of the Notifications page lets you search for specific notifications:

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.

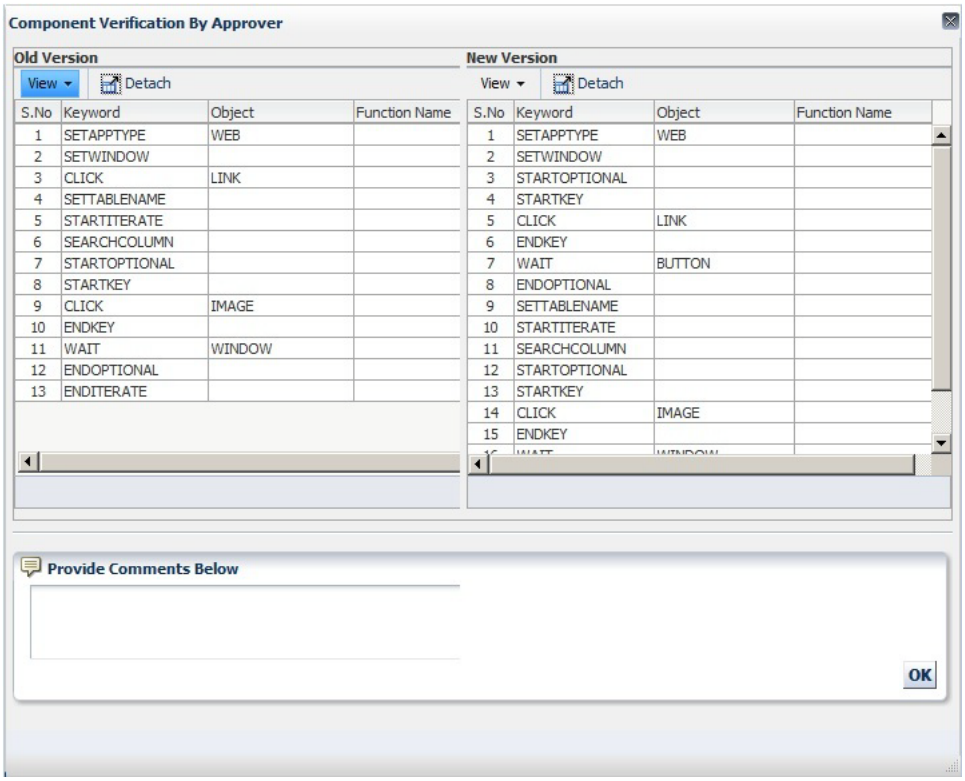
Figure 6–4 Component Details Actions

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.

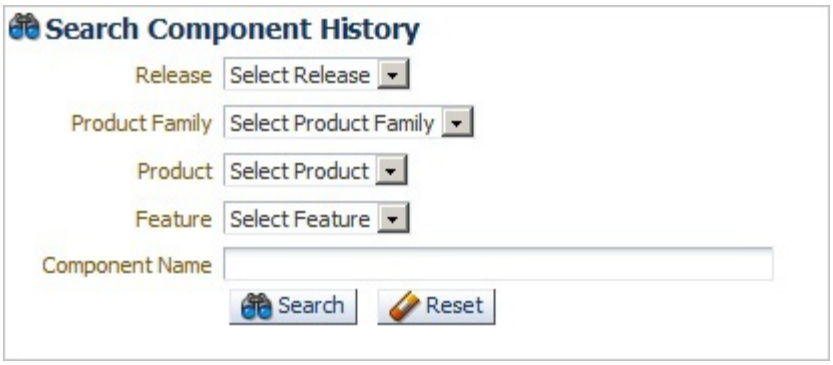
7.2 Searching and Viewing Component History

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:

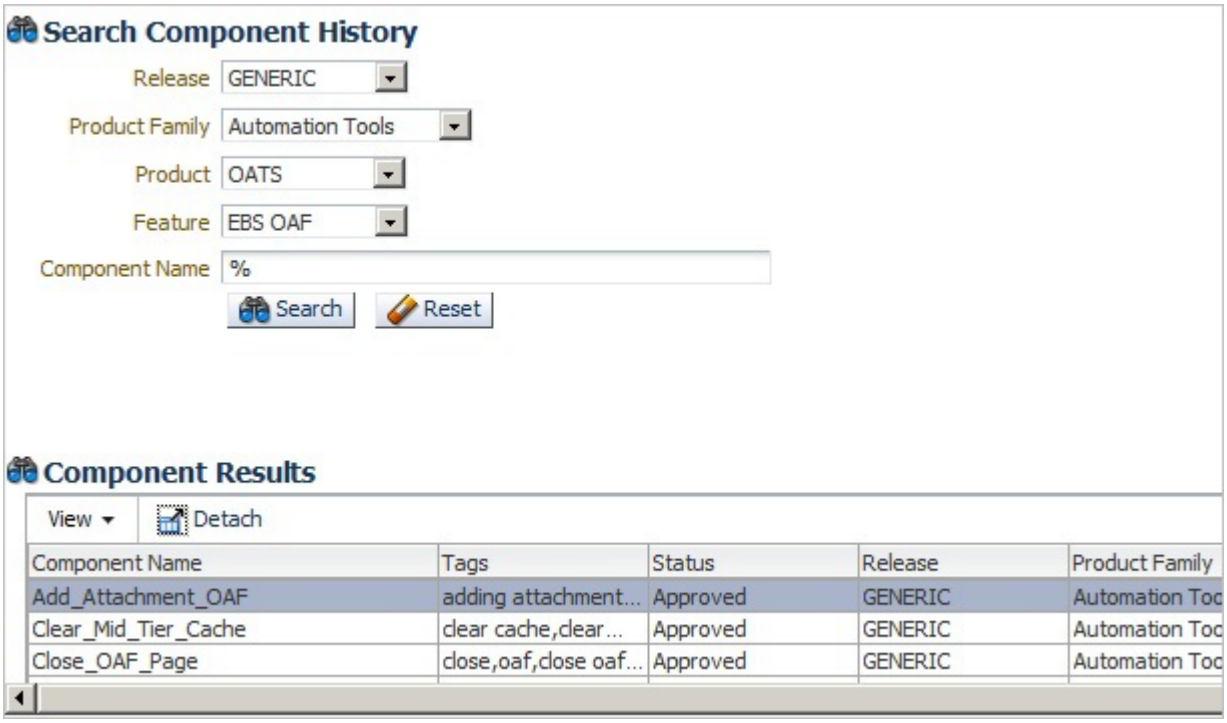
1. Log in and go to the History page.
2. From the **History** categories, select **Components**.

Figure 7–2 Search Component History Options



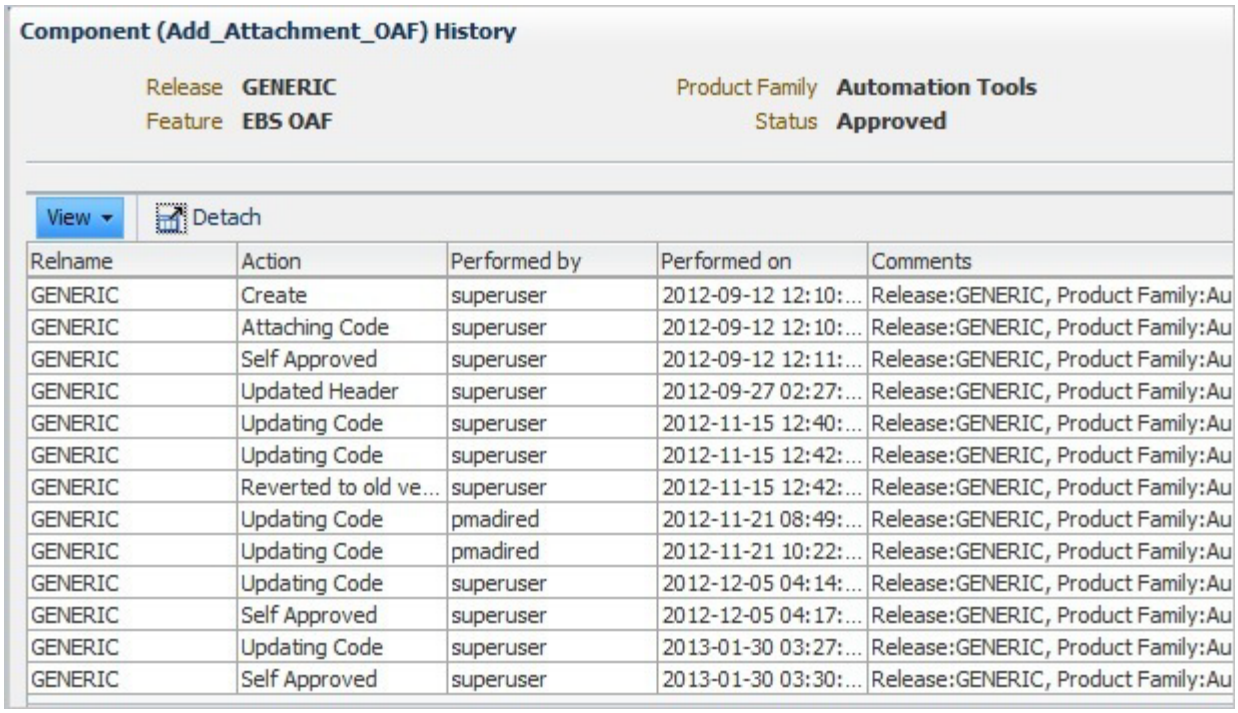
3. Select the Release, Product Family, Product, Feature, and Component name (or use % wildcard) to narrow the search criteria and click **Search** to view the history of the component:

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:

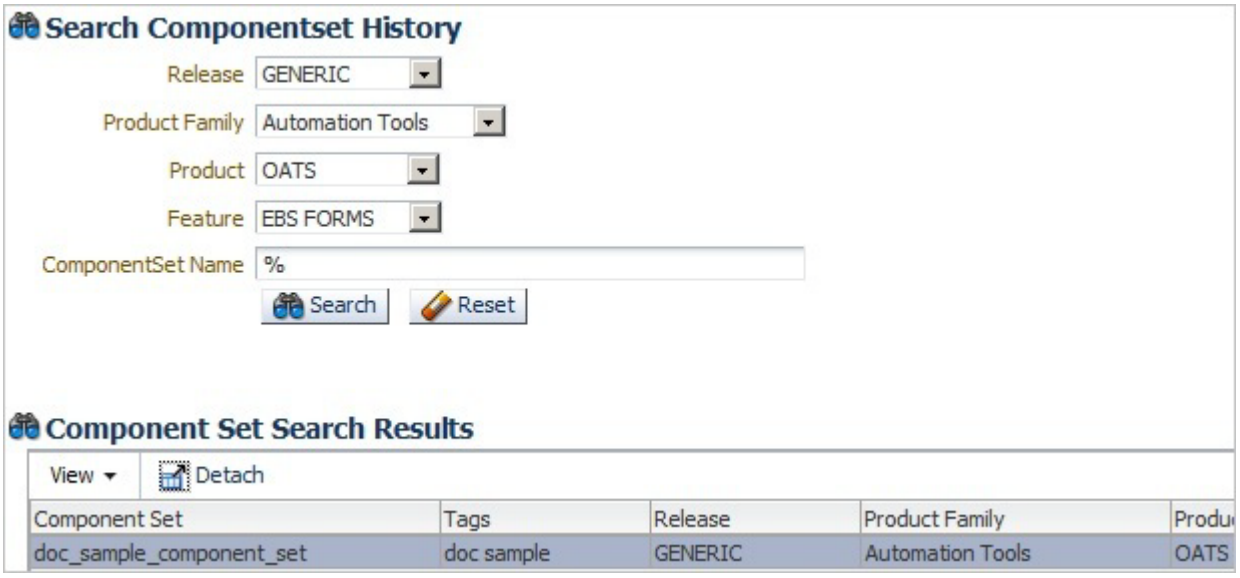
To view Component Set history:

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:

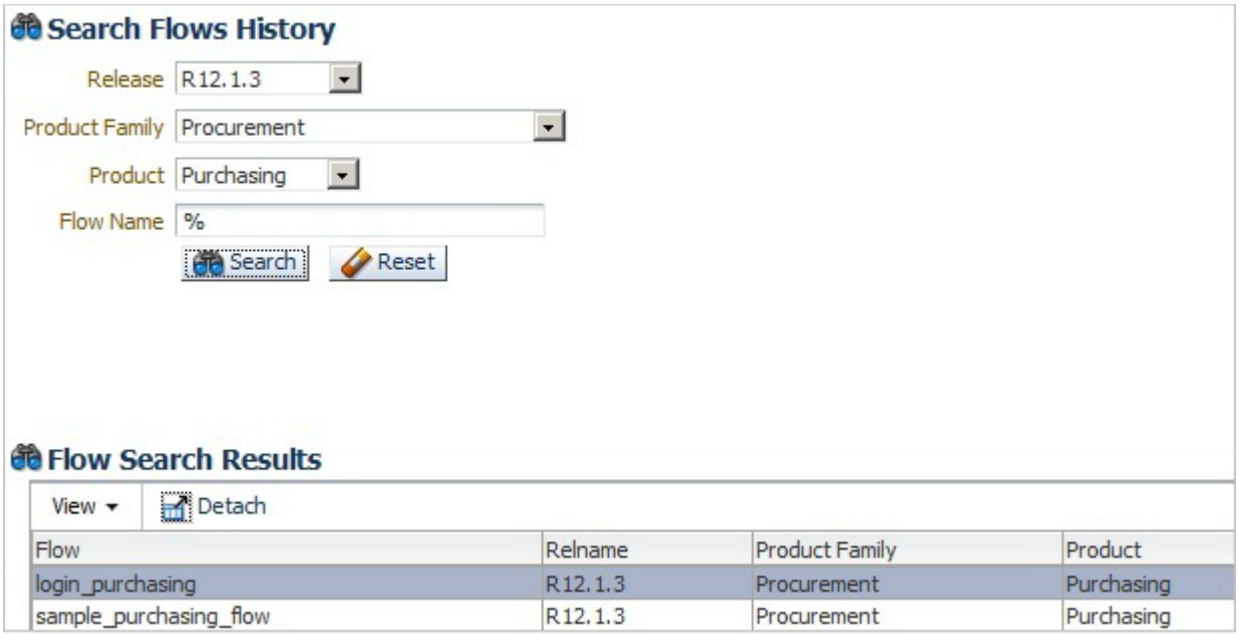
To view Flow history:

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

Figure 7–8 Search Flows History Options

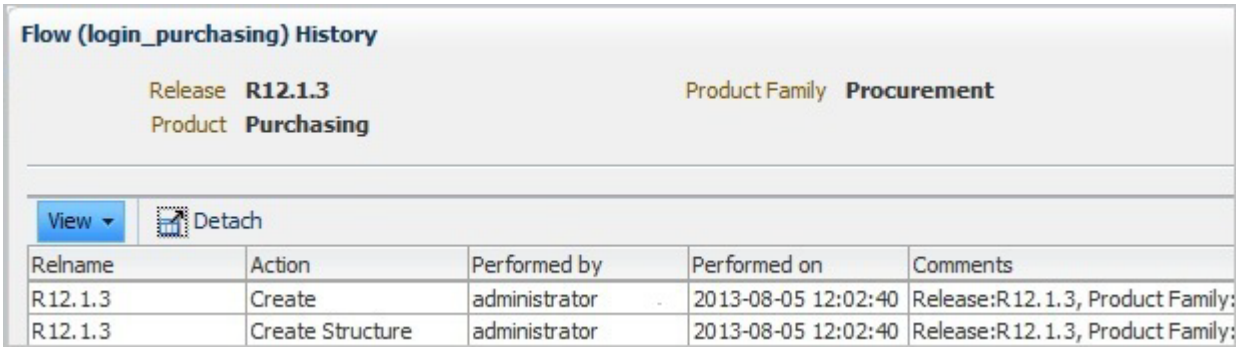
3. Select the Release, Product Family, Product, and Flow Name (or use % wildcard) to narrow the search criteria and click **Search** to view the history of the flow:

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:

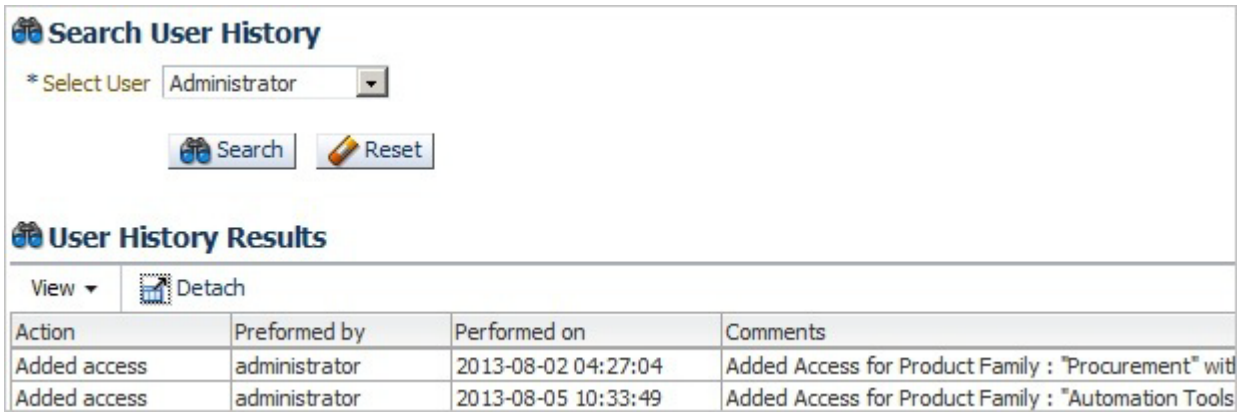
To view User history:

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.

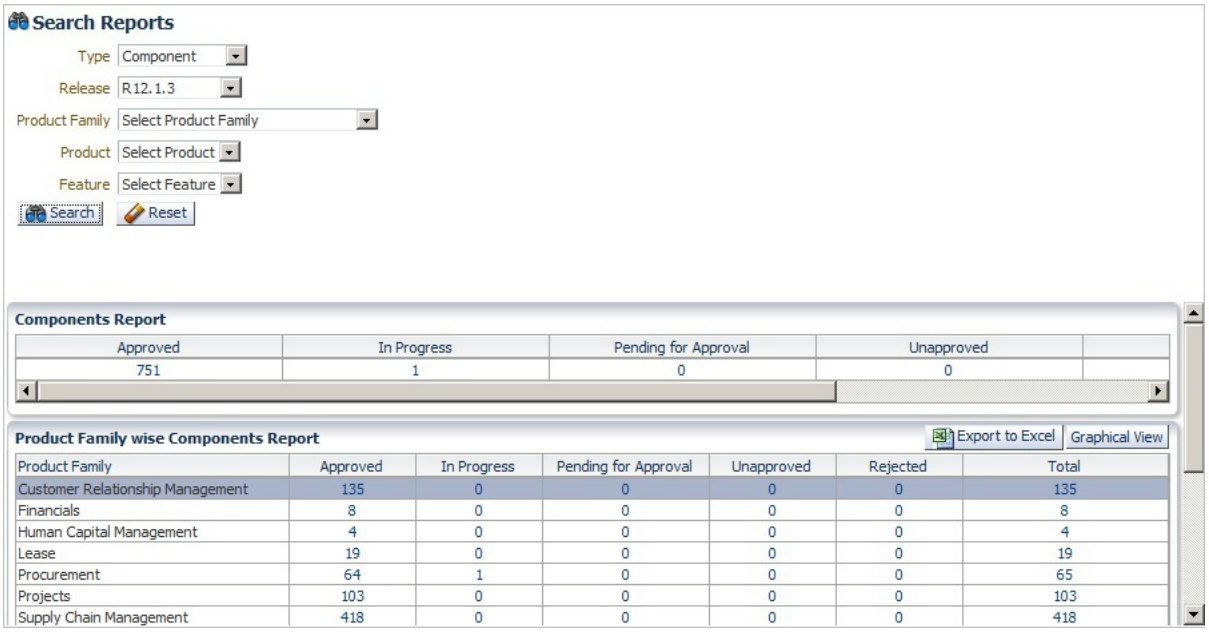
8.2 Searching and Generating Reports

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

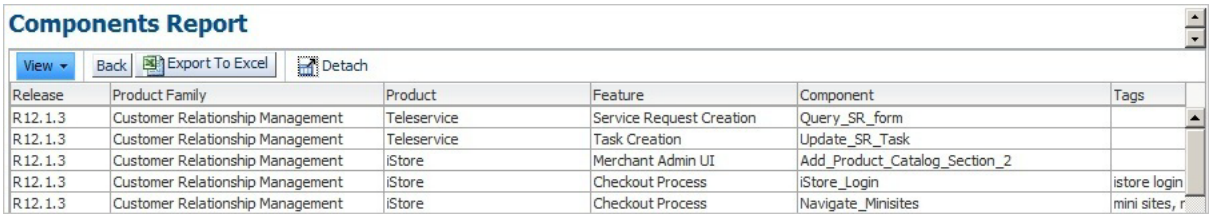
Select the Type, Release, Product Family, Product, and Feature to narrow the search criteria and click **Search** to generate the report:

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



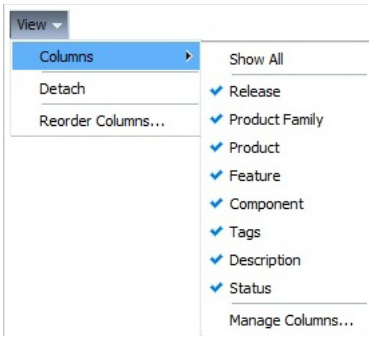
Click a value in a report table to show the report details in table format or click **Graphical View** to show the report details in graphical format. Figure 8–3 shows a sample components report in table format:

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

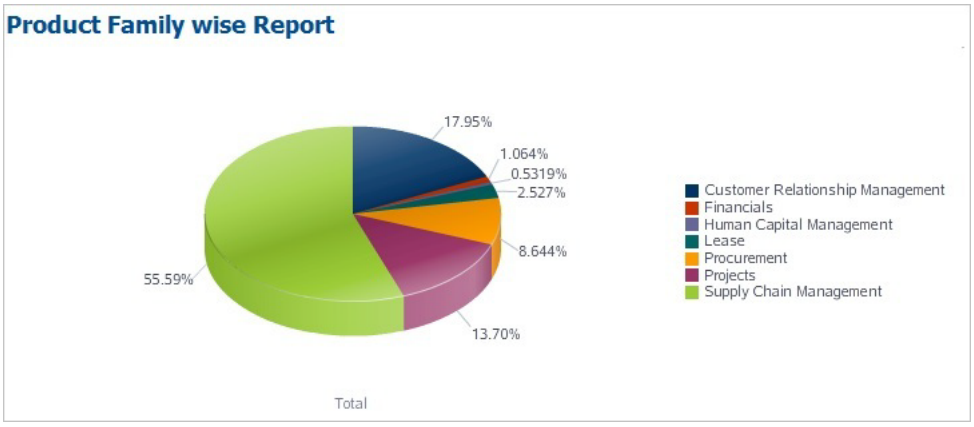


Figure 8–6 shows a sample Component Status report in graphical view:

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:

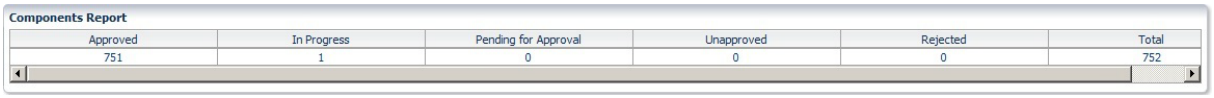
8.3.1 Generating a Component Totals Report

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


Components Report

Approved	In Progress	Pending for Approval	Unapproved	Rejected	Total
751	1	0	0	0	752

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


Components Report

View | Back | Export To Excel | Detach

Release	Product Family	Product	Feature	Component	Tags	Description
R12.1.3	Customer Relationship Management	Teleservice	Service Request Creation	Query_SR_form		SR can be queried
R12.1.3	Customer Relationship Management	Teleservice	Task Creation	Update_SR_Task		Update SR Task
R12.1.3	Supply Chain Management	Process MFG Product Develo...	Formula	Formula_Find	Formula	Find formula in forms
R12.1.3	Supply Chain Management	Process MFG Product Develo...	Formula	Create_Formula	Formula	Creating a formula in forms
R12.1.3	Supply Chain Management	Process MFG Product Develo...	Activity	Create_Activities	Activities	To create activities in forms
R12.1.3	Supply Chain Management	Process MFG Product Develo...	Routing	Create_Pr_Routings	Routing	Create process routing in forms
R12.1.3	Supply Chain Management	Flow Manufacturing	Flow Sch Creation Forms	Click_Flow_Schedule_Details	Flow Schedule Summary	This component is used to click Details b
R12.1.3	Customer Relationship Management	iStore	Merchant Admin UI	Add_Product_Catalog_Section_2		Adds product to a section
R12.1.3	Supply Chain Management	Order Management	Create Sales Order	Book_Sales_Order	Sales Order	Booking a Sales Order
R12.1.3	Supply Chain Management	Order Management	Create Sales Order	Create_SO_Header	Sales Order	Create Sales Order Header
R12.1.3	Supply Chain Management	Order Management	Create Sales Order	Create_SO_Line	Sales Order	Create Sales Order Line
R12.1.3	Supply Chain Management	Advanced Pricing	PriceList Line	Crt_PriceList_Lines	Advanced Pricing-Price List...	Create Price list Line details
R12.1.3	Supply Chain Management	MFG Process Execution	Setups	Process_Exe_Para	Setups	To set the values of Process Execution
R12.1.3	Supply Chain Management	MFG Process Execution	Find Batch /FPO	Find_Batches	Batch Details	To Find a Batch
R12.1.3	Supply Chain Management	Process Quality Management	Master Item Specification	Change_Spec_Status	Specification	Change specification status in forms
R12.1.3	Supply Chain Management	Inventory	Material workbench	Query_Material_Workbench	Query,Material Workbench	Query Material Workbench
R12.1.3	Customer Relationship Management	iStore	Checkout Process	iStore_Login	istore login	login using istore Customer UI URL
R12.1.3	Customer Relationship Management	iStore	Checkout Process	Navigate_Minisites	mini sites, navigate	Navigates to iStore Minisites
R12.1.3	Supply Chain Management	MFG Process Execution	Batch Actions	Batch_Actions_Release	Batch Actions	To Change the status of a batch to WIP
R12.1.3	Supply Chain Management	Item Master	Master Items	Verify_Master_Item_Pur_Tab	Master Items	NAV: Manufacturing and Distribution ma
R12.1.3	Customer Relationship Management	Customer Support Specialist	Task Creation	Create_Task_Csz_1		Creating task in normal way in CSZ
R12.1.3	Supply Chain Management	Flow Manufacturing	Workstation Parameters	Open_Workstation_Parameters	Oracle Applications, Seque...	This component is Used to Launch Work

Row Count: 751

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

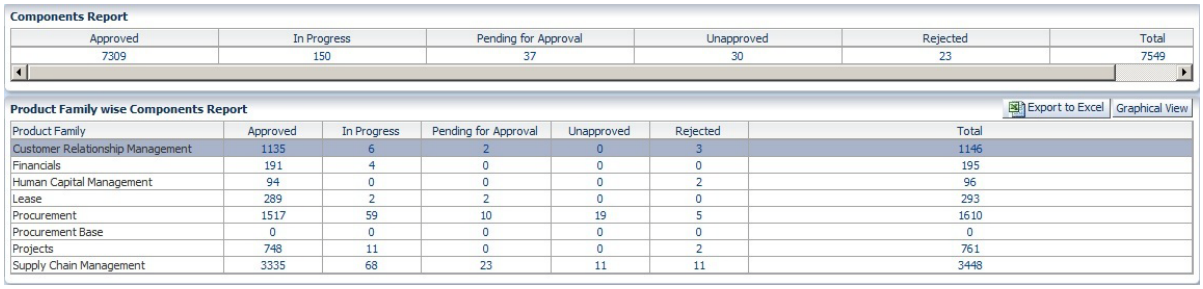
8.3.2 Generating a Component by Product Family Report

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

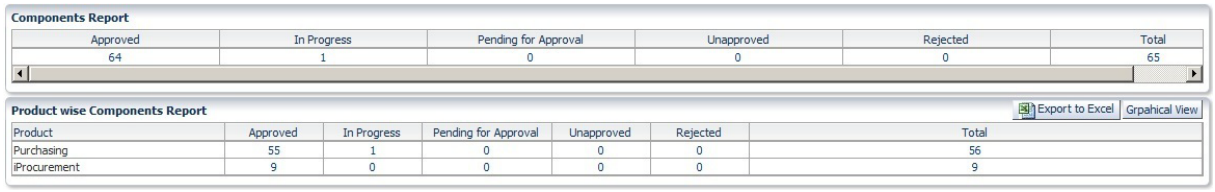
A components by product report shows the number of components in each status for each of the Products in a Product Family of a Release.

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.

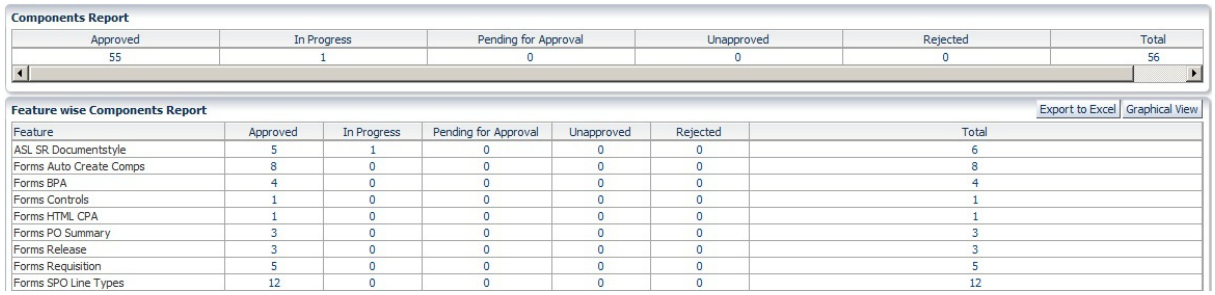
8.3.4 Generating a Component by Feature Report

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:

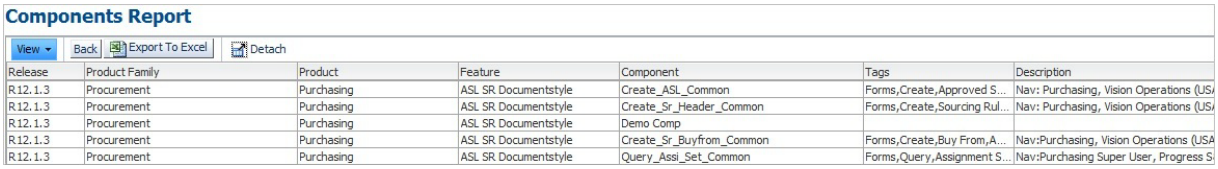
1. Log in and go to the Reports page.
2. Select **Component** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Select a product as the search **Product** setting.
6. Select a feature as the search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

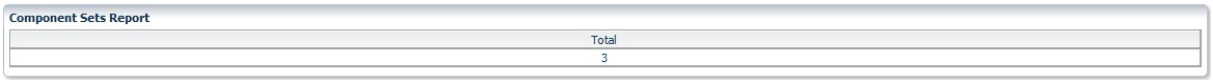
A component totals report shows the total number of components in the Oracle Flow Builder database.

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:

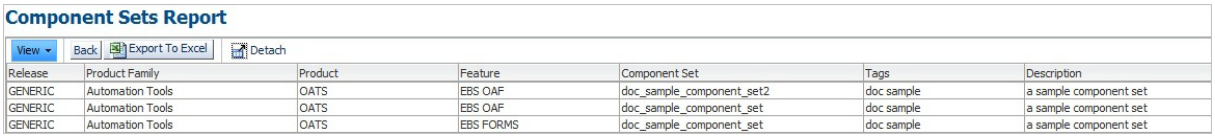
Figure 8–14 Component Set Totals Report in Table View



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.

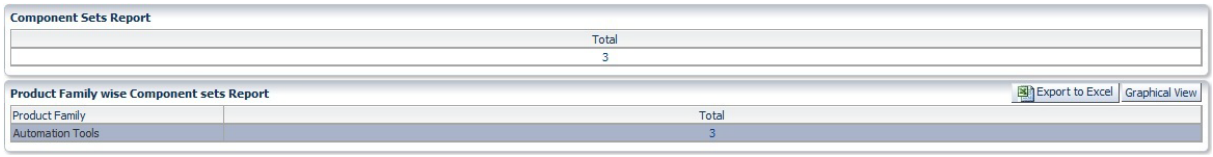
8.4.2 Generating a Component Set by Product Family Report

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.

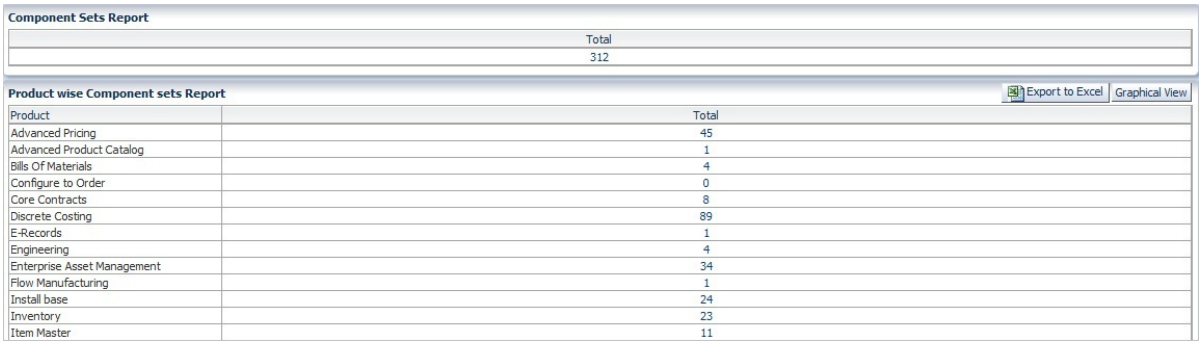
8.4.3 Generating a Component Set by Product Report

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.

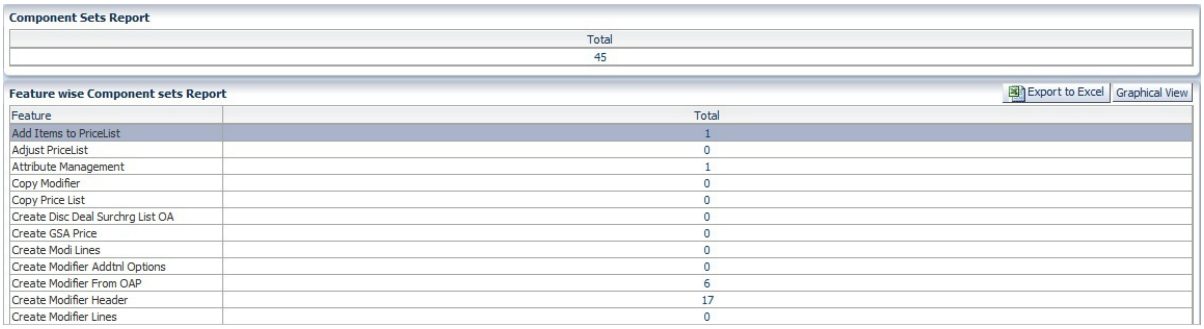
8.4.4 Generating a Component Set by Feature Report

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.

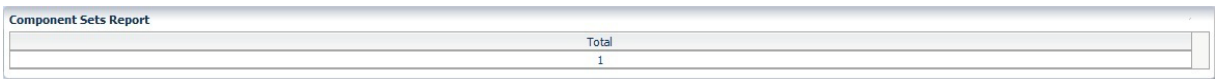
8.4.5 Generating a Component Set Report for a Specific Feature

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



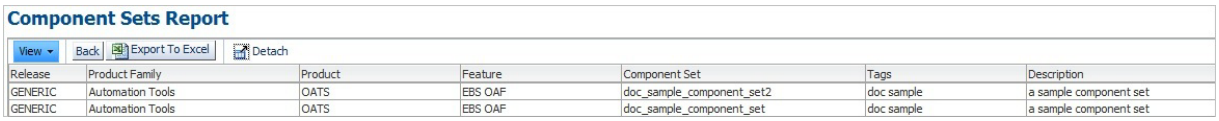
Component Sets Report

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:

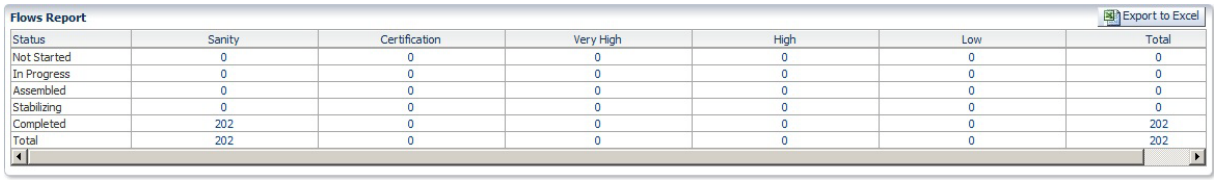
8.5.1 Generating a Flow Totals Report

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



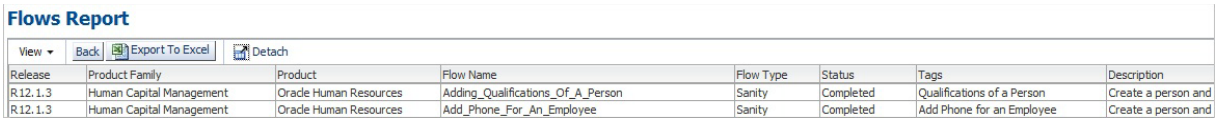
Flows Report — Export to Excel

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.

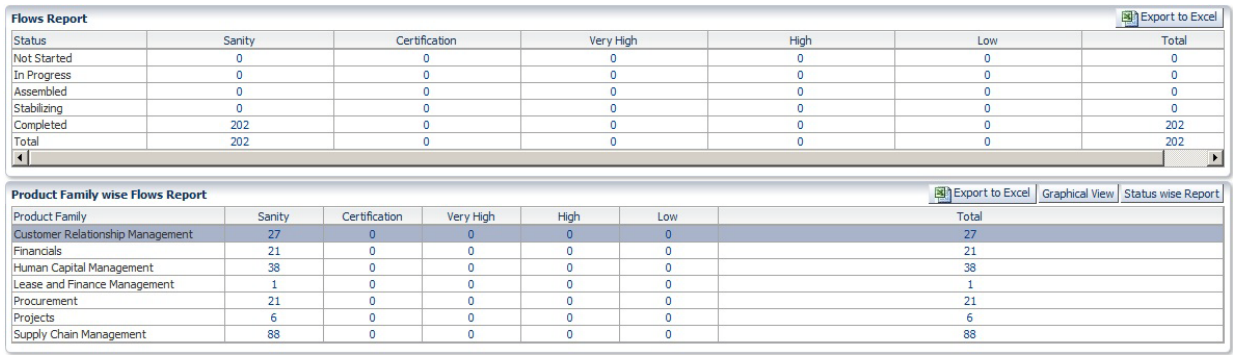
8.5.2 Generating a Flows by Product Family Report

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

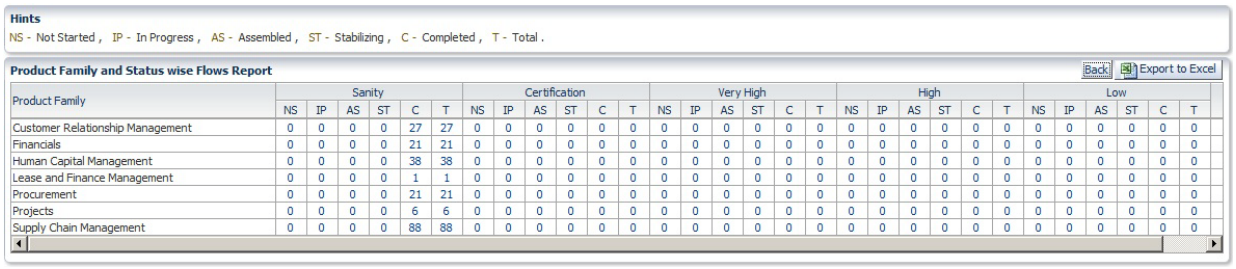
Status	Sanity	Certification	Very High	High	Low	Total
Not Started	0	0	0	0	0	0
In Progress	0	0	0	0	0	0
Assembled	0	0	0	0	0	0
Stabilizing	0	0	0	0	0	0
Completed	202	0	0	0	0	202
Total	202	0	0	0	0	202

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.

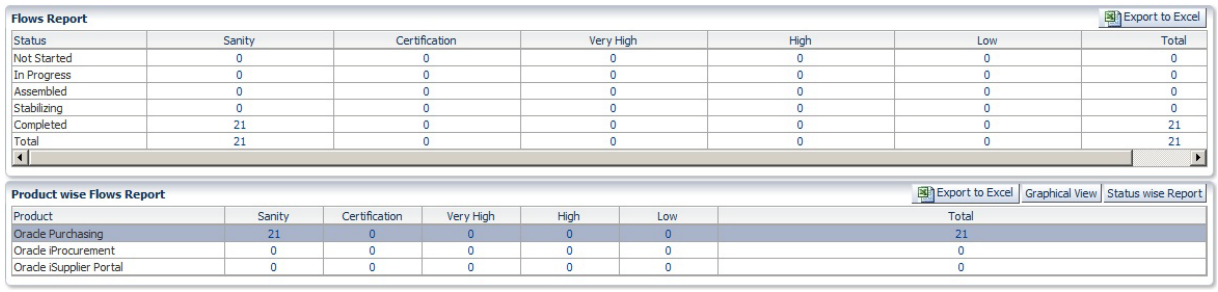
8.5.3 Generating a Flows by Product Report

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:

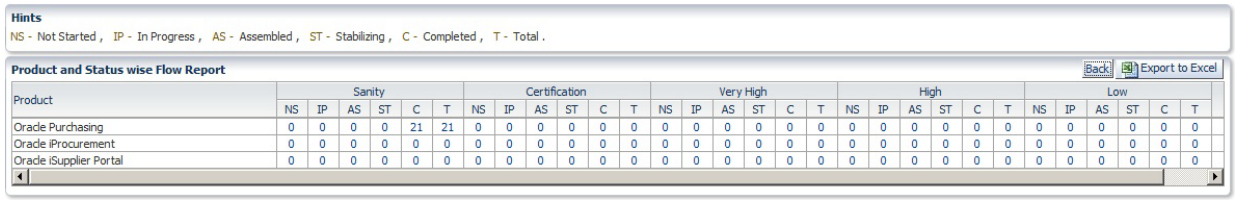
1. Log in and go to the Reports page.
2. Select **Flow** as the search **Type** setting.
3. Select a release as the search **Release** setting.
4. Select a product family as the search **Product Family** setting.
5. Leave or select **Select Product** as the search **Product** setting.
6. Leave or select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

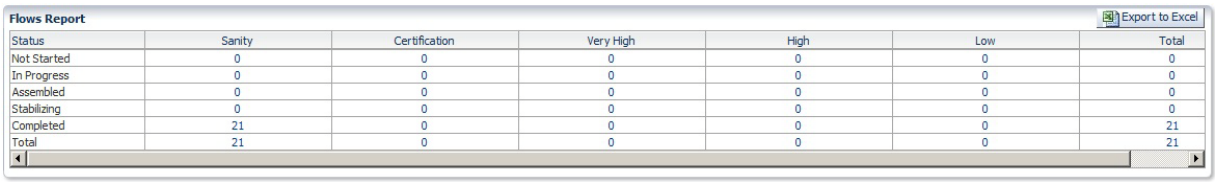
A flow by feature report shows the number of flows in each status for the flows in a specific product.

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

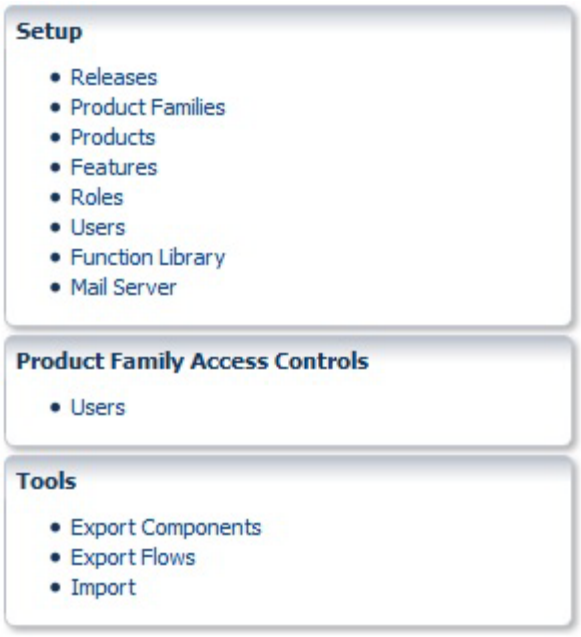
This section provides an overview of the administrative tasks with in the Oracle Flow Builder application. Administrator tasks can be performed by any user with administrator user privileges. However, the administrator user defined during setup of the application performs the initial administrative tasks. 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.
- Tools - import advanced pack

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
3	Importing Advanced Packs See Section 9.5, "Importing Components and Flows"	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

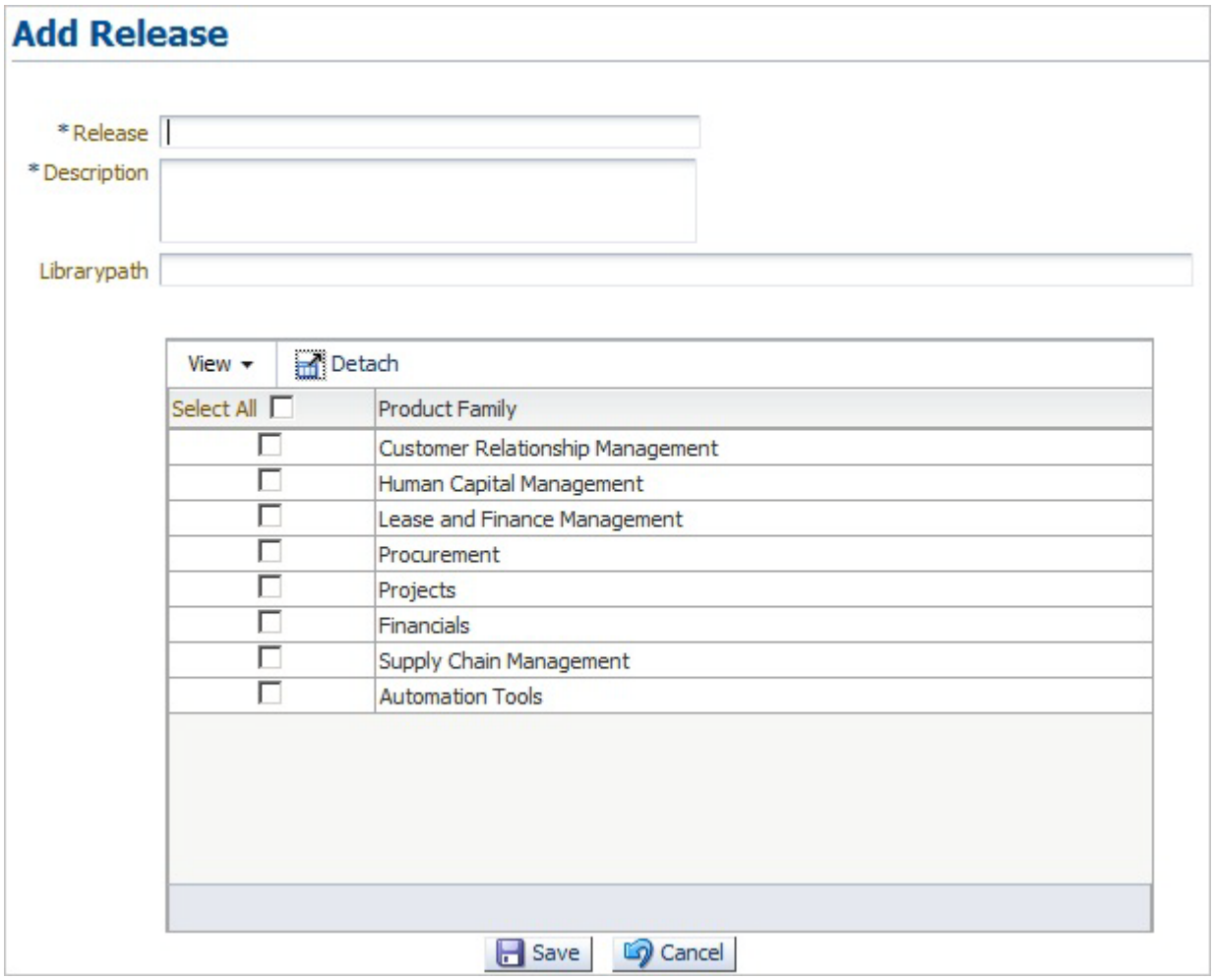
This section explains the procedures for administering the Releases within Oracle Flow Builder. Releases are the top level of the Component Tree hierarchy within the Oracle Flow Builder application.

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. Optionally, enter the path to a custom function library.

Note: For Oracle Flow Builder Release 12.4.0.2 only the default EBS function library is supported. Future releases may support additional function libraries.

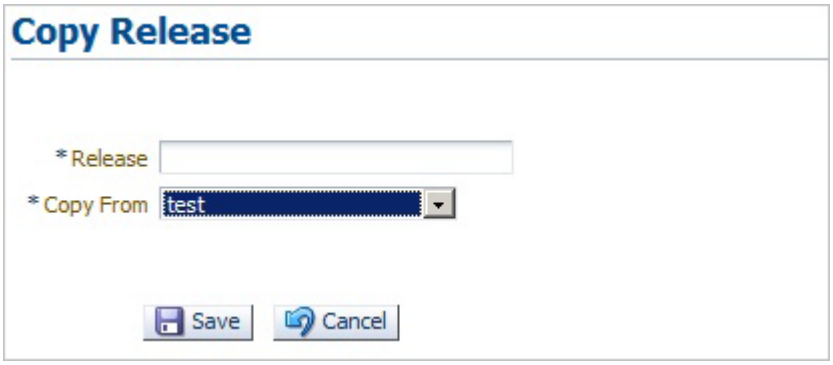
6. Select the product family or families to include under the Release in the Component Tree hierarchy and click **Save**.

9.2.1.2 Copying Releases

Follow these steps to copy an existing Release to a new Release in the Component Tree:

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

Figure 9–3 Copy Release Options



4. Define the new Release name.
5. Select the Release to copy from and click **Save**. Copying an existing release may take some time depending upon how many components, component sets, and flows are defined in the Release being copied.

   After clicking the **Save** button, the UI shows a waiting icon until the copy operation finishes. No other operation is allowed. When the Search Release page returns, it indicates the copy operation is finished.

   If you open Oracle Flow Builder in another instance of Explorer or another user opens an Explorer instance and attempts to copy a release at the same time, an error appears indicating there is a Copy Release procedure still running, and they must wait until it is finished.
6. After the copy operation finishes, log out of the Oracle Flow Builder application and log back in as an administrator to view the copied release structure.

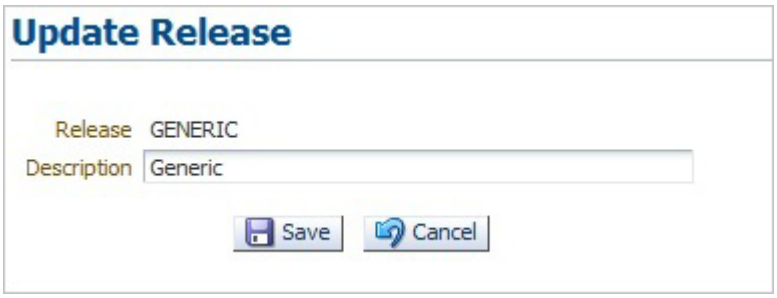
9.2.1.3 Updating Releases

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 9–4 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

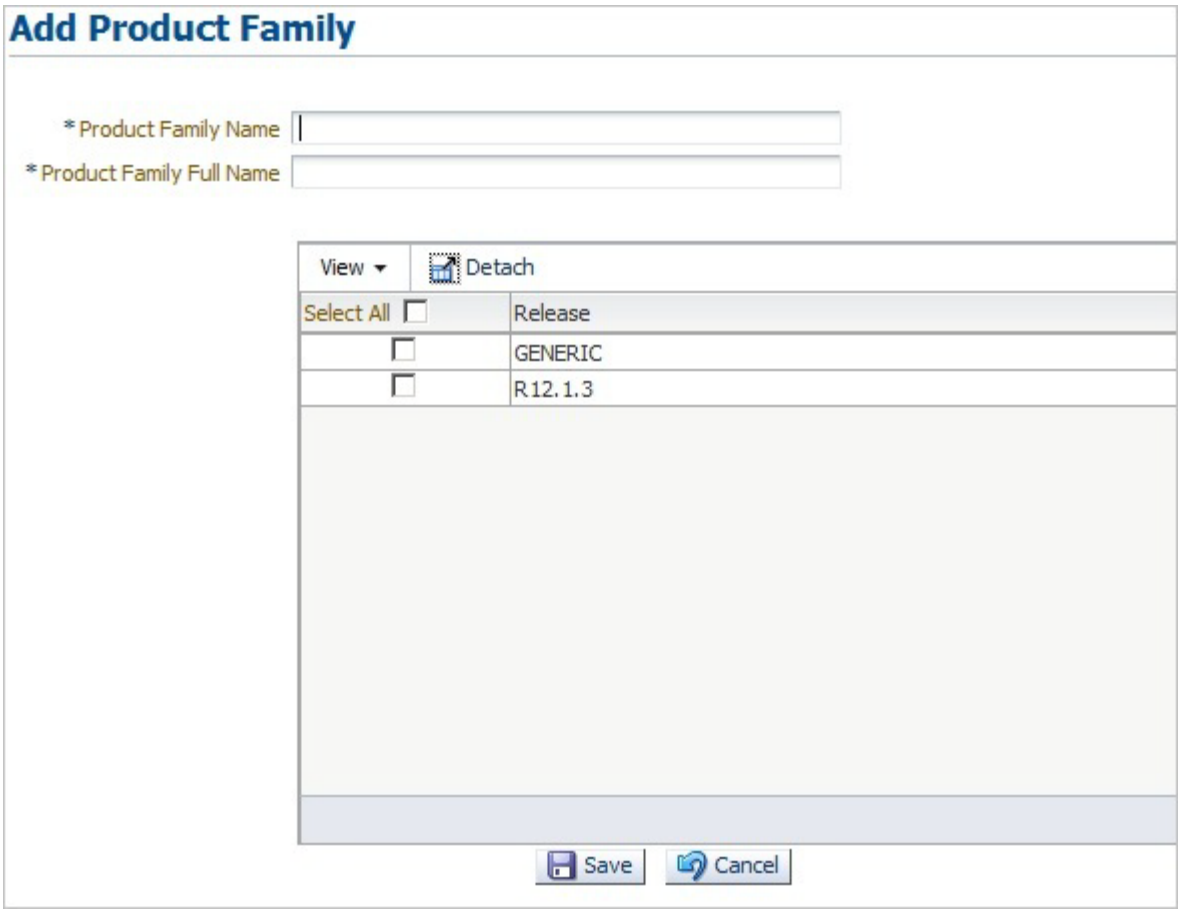
This section explains the procedures for administering the Product Families within Oracle Flow Builder. Product Families are defined to categorize the products and features in a Release hierarchy within the Component Tree.

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 9–5 Add Product Family Options



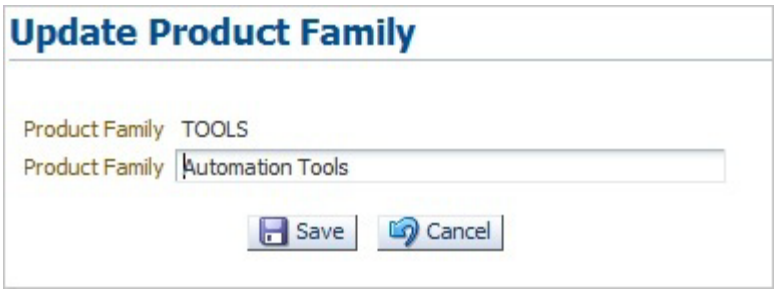
4. Define the Product Family Name and Product Family Full Name.
5. Select the Release(s) and click **Save**.

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 9–6 Update Product Family Options



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

9.2.3 Setting Up Products

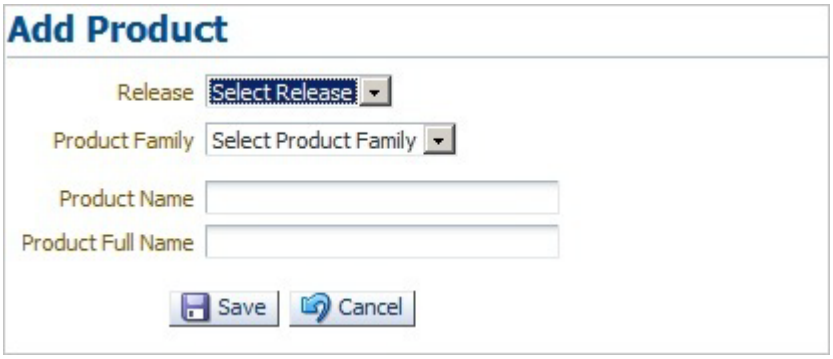
This section explains the procedures for administering the Products within Oracle Flow Builder. Products are defined to categorize the features in a Product Family hierarchy within the Component Tree.

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 9–7 Add Product Options



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

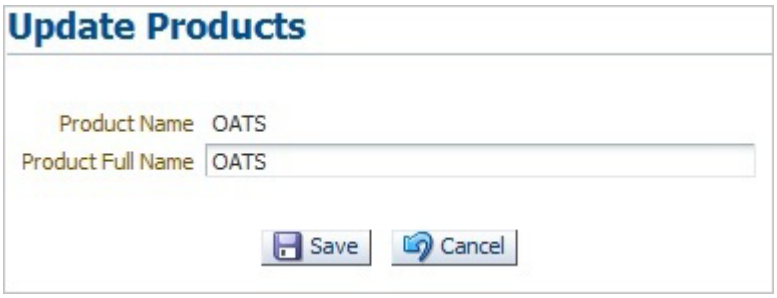
9.2.3.2 Updating Products

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 9–8 Update Product Options



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

9.2.4 Setting Up Features

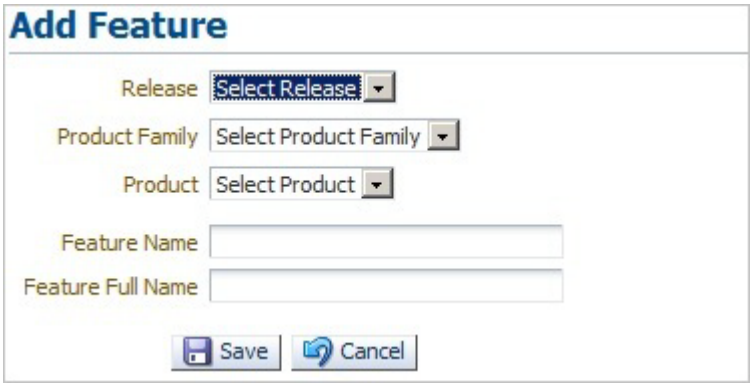
This section explains the procedures for administering the Features within Oracle Flow Builder. Features define specific features of a Product to test within the Component Tree.

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 9–9 Add Feature Options



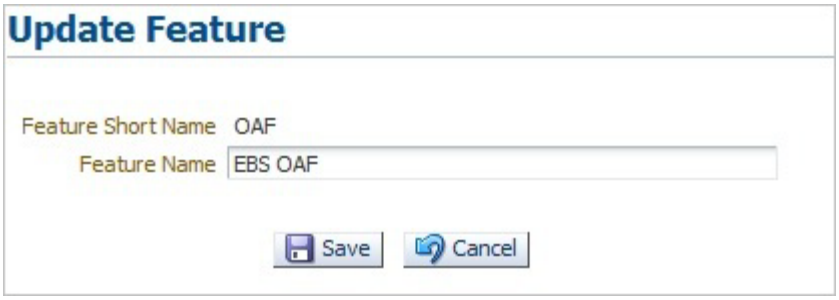
4. Select the Release.
5. Select the Product Family.
6. Select the Product.
7. Define the Feature Name and Feature Full Name and click **Save**.

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 9–10 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 9–11 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.

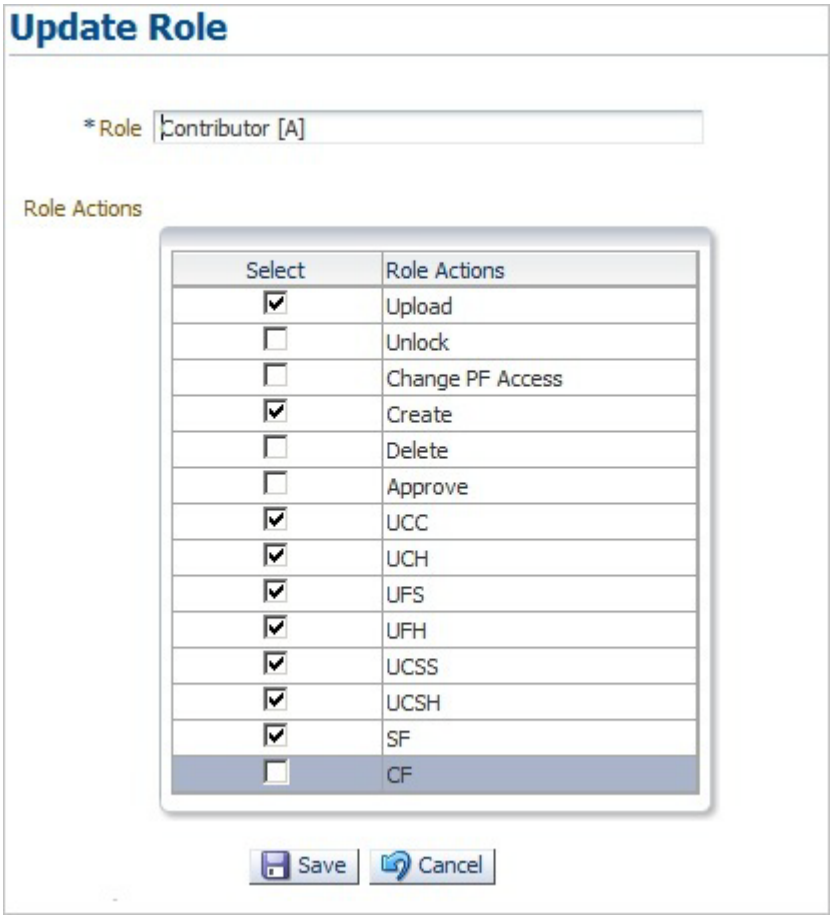
6. Click **Save** to add the user role.

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 9–12 Update Role Options



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

9.2.6 Setting Up Users

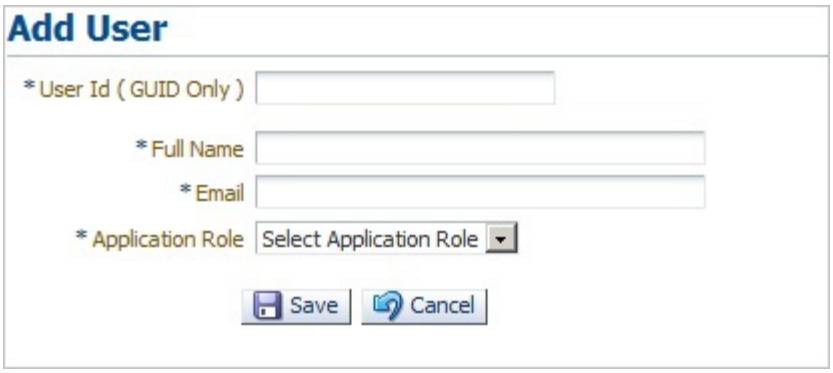
This section explains the procedures for administering the users within Oracle Flow Builder.

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 9–13 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.

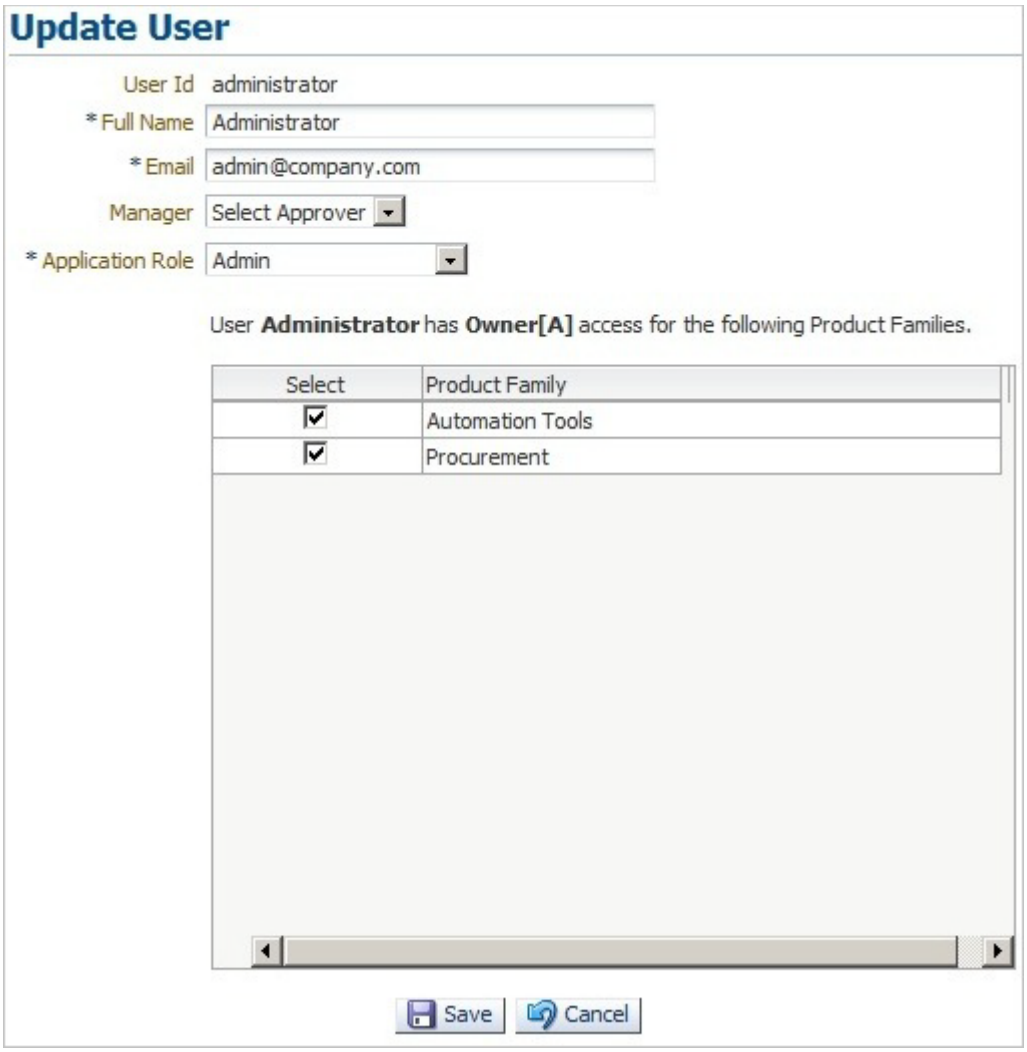
6. Click **Save** to add the user.

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 9–14 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:

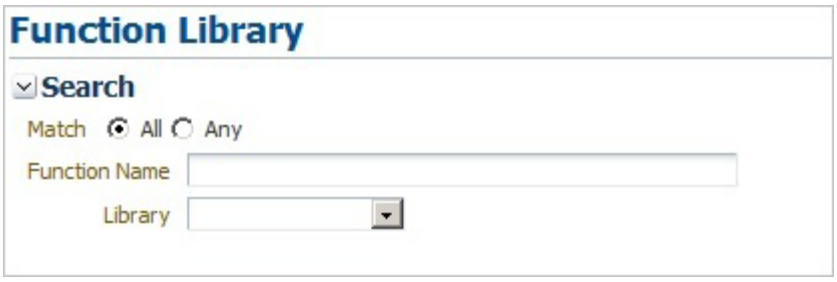
1. Start OpenScript.
2. From the **Tools** menu, select **Manage Repositories**.
3. Click **Add**.
4. Enter `OATS` as the **Name**.
5. Enter *<any-location-on-disk>* as the **Location**.
6. Click **OK**.
7. Copy the function libraries (`function-libs.zip`) included with the Oracle Flow Builder download 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 9–15 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
   ```

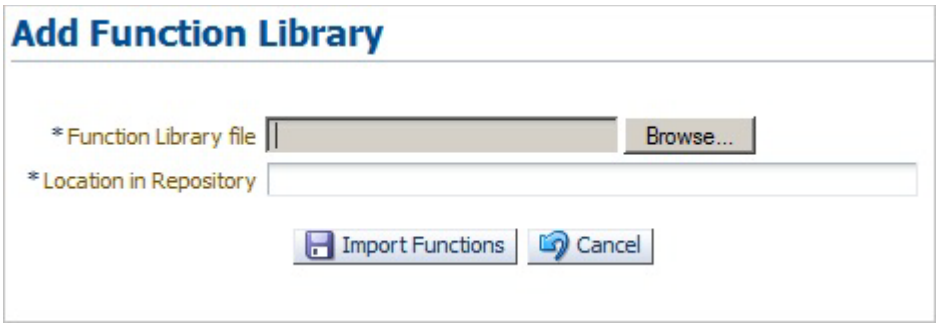
9. Click **Finish**.
10. Add your custom code to the function library script.
    - Use the script recorder to record steps.
    - Switch to the Java Code view and edit the code in the functions.
    - Functions declared public will be imported into Oracle Flow Builder.
    - Supporting functions should 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 9–16 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 Oracle 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 9–17 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 9–18 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 9–19 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**.

9.4 Exporting Components and Flows

An Administrator can export components, component sets, and flows from the Oracle Flow Builder instance. Exporting components and flows creates a .zip file archive of the selected components or flows that can be imported into anothe Oracle Flow Builder instance.

Importing and exporting components and flows can be used for the following:

- Moving from a test Oracle Flow Builder instance to a production Oracle Flow Builder instance.
- Moving from Oracle Flow Builder version x to Oracle Flow Builder version y.
- Make available Partner created components/flows a customer.
- Backing up.

The following tasks are performed by an the Oracle Flow Builder administrator:

- Exporting Components and Component Sets
- Exporting Flows

9.4.1 Exporting Components and Component Sets

Export provides a single export capability. There is no multi-select capabilty. If multi-select is required, you can do multiple exports or export by the level of Release, Product Family, or Product. Release, Product Family, or Product can be selected using the filter selections similar to Search. All components, sets, or flows selected by search will be exported.

To export components and component sets:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Tools**, click **Export Components**.
3. Select the Release.
4. Select the Product Family.
5. Select the Product.
6. Select the Feature.
7. Enter the ID of the componet or component set (or use % wildcard).
8. Click **Export**.
9. Click **Save File**.
10. Click **OK**.
11. Specify the location and file name where you want to save the file and click **Save**.

9.4.2 Exporting Flows

To export components and component sets:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Tools**, click **Export Flows**.
3. Select the Release.
4. Select the Product Family.
5. Select the Product.
6. Enter the ID of the flow (or use % wildcard).
7. Click **Export**.
8. Click **Save File**.
9. Click **OK**.
10. Specify the location and file name where you want to save the file and click **Save**.

9.5 Importing Components and Flows

An Administrator can import exported components and flows from another Oracle Flow builder instance or an Advanced Pack deliverable to apply the data to the Oracle Flow Builder application instance. Advanced Packs may contain additional components, component sets, and flows that can be included as add-ons or updates.

The following tasks are performed by an the Oracle Flow Builder administrator:

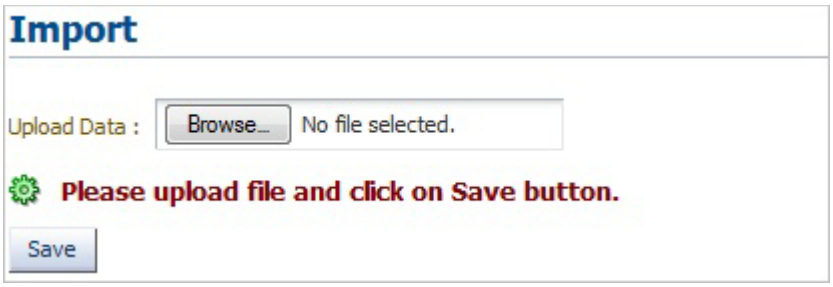
- Importing Components and Flows from Exported Files

9.5.1 Importing Components and Flows from Exported Files

To import exported components and flows or Advanced Pack data files:

1. Log in using Administrator credentials and go to the Administration page.
2. From **Tools**, click **Import**.

Figure 9–20 Import Options



3. Select **Browse**.
4. Select the Advanced Pack .zip file or the exported.
5. Click **Open**.
6. Click **Save**.

A

Keyword Reference

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

- Keywords and Objects List
- Web/EBS Forms Keywords and Objects
- PLSQL/Open Interface Keywords and Objects

A.1 Keywords and Objects List

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
ACTIVITYEVENT	Specify an Activity Event.	No objects required.
APPROVE	Approve the specified object.	The following objects are valid: ■ ALERT ■ CHOICEBOX ■ FLEXWINDOW ■ RESPONSEBOX
BEGINCALL	Specify the start of a Call Keyword set. Used with ENDCALL.	No objects required.
CALENDAREVENT	Specify a Calenadar Event.	No objects required.
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

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

Keywords	Description	Valid Objects
CANCEL	Cancel the specified object.	The following objects are valid: ■ ALERT ■ CHOICEBOX ■ FLEXWINDOW ■ RESPONSEBOX
CHECK	Check a checkbox object.	The following objects are valid: ■ CHECKBOX
CLICK	Click the specified object.	The following objects are valid: ■ ADFBUTTON ■ ADFCOMMANDMENU ■ ADFTOOLBARBUTTON ■ ALERTBUTTON ■ BUTTON ■ CHOICEBOXBUTTON ■ EDIT ■ FLEXCANCEL ■ FLEXCOMBINATION ■ FLEXOK ■ IMAGE ■ LINK ■ MENU ■ TAB ■ TOOLBAR
CLICKBUTTONK	Click the specified button object.	The following objects are valid: ■ DIALOG
CLICKICON	Click the specified icon object.	No objects required.
CLICKSEARCHICON	Click the specified search icon object.	No objects required.
CLOSE	Close the specified window object.	The following objects are valid: ■ ADFPANELWINDOW ■ DIALOG ■ WINDOW
COLLAPSE	Collapse the specified tree object.	The following objects are valid: ■ ADFPANELACCORDIAN ■ ADFPANELBOX ■ ADFPANELSPLITTER ■ TOOLBAR ■ TREE
COLLAPSENODE	Close the specified tree node object.	The following objects are valid: ■ TREE ■ TREELIST ■ TREETABLE

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

Keywords	Description	Valid Objects
DISPLAYCHANGE	Display the specified change.	No objects required.
ENDBLOCK	Specify the end of the Block Keyword set. Used with STARTBLOCK.	No objects required.
ENDCALL	Specify the end of the Call Keyword set. Used with BEGINCALL.	No objects required.
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.
ENDOBJECTTYPE	Specify the start of an Object Type Keyword set. Used with STARTOBJECTTYPE.	No objects required.
ENDOPTIONAL	Specify the end of an Optional Keyword set. Used with STARTOPTIONAL.	No objects required.
ENDQUERY	Specify the end of the query block Keyword set. Used with STARTQUERY.	No objects required.
ENDRECORDTYPE	Specify the end of a Record Type Keyword set. Used with STARTRECORDTYPE.	No objects required.
ENDRECOVERY	Specify the end of a Recovery Keyword set. Used with STARTRECOVERY.	No objects required.
ENDROWITERATOR	Specify the start of the iterate block Keyword set. Used with STARTROWITERATOR.	No objects required.
ENDTAB	Specify the end of a Tab Keyword set. Used with STARTTAB.	No objects required.
ENDTABLETYPE	Specify the end of a Table Type Keyword set. Used with STARTTABLETYPE.	No objects required.
ENDVARRAYTYPE	Specify the end of an VArray Type Keyword set. Used with STARTVARRAYTYPE.	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: ■ ADFBUTTON ■ ADFCOMBOLOV ■ ADFCOMMANDMENU ■ ADFMANYCHECKBOX ■ ADFMANYLISTBOX ■ ADFNAVIGATIONITEM ■ ADFNAVIGATIONPANE ■ ADFOUTPUTTEXT ■ ADFPANELACCORDIAN ■ ADFPANELBOX ■ ADFPANELHEADER ■ ADFPANELLABEL ■ ADFPANELSPLITTER ■ ADFPANELWINDOW ■ ADFTOOLBARBUTTON ■ ADFUPLOADFILE ■ ALERT ■ BUTTON ■ CHECKBOX ■ CHOICEBOX ■ DATE ■ DIALOG ■ EDIT ■ IMAGE ■ LINK ■ LIST ■ LISTBOX ■ LOV ■ MENU ■ RADIOBUTTON ■ SPREADTABLE ■ TAB ■ TABLE ■ TEXTAREA ■ TOOLBAR ■ TREE ■ TREETABLE ■ WINDOW

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

Keywords	Description	Valid Objects
EXPAND	Expand the specified ADF object.	The following objects are valid: ■ ADFPANELACCORDIAN ■ ADFPANELBOX ■ ADFPANELSPLITTER ■ TOOLBAR
EXPAND/COLLAPSE	Expand the specified ADF object.	The following objects are valid: ■ TABLE
EXPANDNODE	Expand the specified tree node object.	The following objects are valid: ■ TREE ■ TREELIST
FILTER	Filter the specified ADF object.	The following objects are valid: ■ TABLE
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

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

Keywords	Description	Valid Objects
FOCUS	Focus on the specified ADF object.	The following objects are valid: ■ ADFBUTTON ■ ADFCOMBOLOV ■ ADFCOMMANDMENU ■ ADFMANYCHECKBOX ■ ADFMANYLISTBOX ■ ADFNAVIGATIONITEM ■ ADFNAVIGATIONPANE ■ ADFOUTPUTTEXT ■ ADFPANELACCORDIAN ■ ADFPANELBOX ■ ADFPANELHEADER ■ ADFPANELLABEL ■ ADFPANELSPLITTER ■ ADFPANELWINDOW ■ ADFTOOLBARBUTTON ■ ADFUPLOADFILE ■ CHECKBOX ■ DATE ■ DIALOG ■ IMAGE ■ LINK ■ LIST ■ LISTBOX ■ MENU ■ RADIOBUTTON ■ TAB ■ TABLE ■ TOOLBAR ■ TREE ■ TREETABLE
FOCUSSHOW	Focus on and show the specified ADF object.	The following objects are valid: ■ MENU

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

Keywords	Description	Valid Objects
FUNCTIONCALL	Call a function from the specified function library.	The following libraries are valid: ■ cRMLIB ■ eBSLibrary ■ fINLIB ■ gENAPILIB ■ gENLIB ■ gENWSLIB ■ hRMSLIB ■ pRJTBLVERIFYLIB ■ pROCLIB ■ pROJLIB ■ sCMLIB ■ tELNETLIB ■ wEBTABLELIB
GET	Get the specified object.	The following objects are valid: ■ ALERT ■ CHOICEBOX ■ EDIT ■ FIELD ■ LINK ■ LIST ■ LISTBOX ■ SPREADCELL ■ STATUS ■ TAB ■ TEXT ■ TEXTAREA

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

Keywords	Description	Valid Objects
GETATTRIBUTE	Get the attributes of the specified object.	The following objects are valid: ■ ADFBUTTON ■ ADFCOMBOLOV ■ ADFCOMMANDMENU ■ ADFMANYCHECKBOX ■ ADFMANYLISTBOX ■ ADFNAVIGATIONITEM ■ ADFNAVIGATIONPANE ■ ADFOUTPUTTEXT ■ ADFPANELACCORDIAN ■ ADFPANELBOX ■ ADFPANELHEADER ■ ADFPANELLABEL ■ ADFPANELSPLITTER ■ ADFPANELWINDOW ■ ADFTOOLBARBUTTON ■ ADFUPLOADFILE ■ BUTTON ■ CHECKBOX ■ DATE ■ DIALOG ■ EDIT ■ IMAGE ■ LINK ■ LIST ■ LISTBOX ■ LOV ■ MENU ■ RADIOBUTTON ■ TAB ■ TABLE ■ TOOLBAR ■ TREE ■ TREETABLE ■ WINDOW
GETCELLDATA	Get the cell data of the specified table object.	The following objects are valid: ■ TABLE ■ TREETABLE
GETCELLDATABYROWINDEX	Get the cell data of the specified ADF table object by row index.	The following objects are valid: ■ TABLE ■ TREETABLE

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

Keywords	Description	Valid Objects
GETCOLUMNCOUNT	Get the column count of the specified ADF table object.	The following objects are valid: ■ TABLE ■ TREETABLE
GETCOLUMNHEADER	Get the column header of the specified ADF table object.	The following objects are valid: ■ TABLE ■ TREETABLE
GETITEMVALUES	Get the value of the specified list object.	The following objects are valid: ■ LIST
GETROWCOUNT	Get the row count of the specified ADF table or tree object.	The following objects are valid: ■ TABLE ■ TREE ■ TREETABLE
GETROWKEY	Get the row key of the specified ADF tree or treetable object.	The following objects are valid: ■ TREE ■ TREETABLE
GETSELECTEDTAB	Get the selected tab index of the specified ADF object.	The following objects are valid: ■ ADFNAVIGATIONPANE ■ TAB
GETVISIBLEROWCOUNT	Get the row count of the visible rows of the specified ADF table or tree object.	The following objects are valid: ■ TABLE ■ TREE ■ TREETABLE
GETVISIBLESTARTROWINDEX	Get the index of the starting visible row of the specified ADF table or tree object.	The following objects are valid: ■ TABLE ■ TREE ■ TREETABLE
HIDE	Hide the specified ADF menu object.	The following objects are valid: ■ MENU
INSERT	Insert Database Row data into a Database Table.	The following objects are valid: ■ TABLE
INVOKESOFTKEY	Invoke the soft key on the specified object.	The following objects are valid: ■ EDIT ■ SPREADTABLE
ISVISIBLE	Determines if the specified ADF dialog object is visible.	The following objects are valid: ■ DIALOG
LAUNCH	Launch the specified browser object.	The following objects are valid: ■ BROWSER
LEFTCLICK	Perform a left mouse click on the specified ADF tree or treetable object.	The following objects are valid: ■ TREE ■ TREETABLE

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

Keywords	Description	Valid Objects
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
MOVE	Perform a move action on the current object.	No objects required.
MOVEALL	Perform a move action on the current object.	No objects required.
NAVIGATE	Navigate to the specified URL.	No objects required.
POPUP	Perform a tab key press on the specified object.	The following objects are valid: ■ DATE
PRESSENTER	Perform a press Enter key action on the specified ADF object.	The following objects are valid: ■ ADFNAVIGATIONPANE ■ EDIT ■ IMAGE
PRESSTABKEY	Perform a tab key press on the specified object.	The following objects are valid: ■ EDIT
REMOVE	Remove the current ADF object.	No objects required.
REMOVEALL	Remove the all ADF objects.	No objects required.
RIGHTCLICK	Perform a right mouse click on the specified ADF tree or treetable object.	The following objects are valid: ■ TREE ■ TREETABLE
SCROLL	Scroll the specified ADF table or tree object.	The following objects are valid: ■ TABLE ■ TREE ■ TREETABLE
SCROLLTOROW	Scroll the specified ADF table or tree object.	The following objects are valid: ■ TABLE ■ TREE ■ TREETABLE
SEARCHBYDYNAMICCOLUMN	Search by dynamic column.	No objects required.
SEARCHCOLUMN	Search by column.	No objects required.
SEARCHEMPTYROW	Search for an empty column.	No objects required.

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

Keywords	Description	Valid Objects
SELECT	Perform a select action on the specified object.	The following objects are valid: ■ ADFLOV ■ ADFMANYLISTBOX ■ ADFNAVIGATIONPANE ■ COLUMN ■ LIST ■ LISTBOX ■ RADIOBUTTON ■ SEARCHLIST ■ TAB ■ TABLE ■ TREELIST
SELECTALLROWS	Perform a select all rows action on the specified spreadtable object.	The following objects are valid: ■ ADFMANYCHECKBOX ■ ADFMANYLISTBOX ■ 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 ■ TREETABLE
SELECTROW	Perform a select row action on the specified spreadtable object.	The following objects are valid: ■ SPREADTABLE ■ TABLE ■ TREETABLE
SELECTTAB	Perform a select tab action on the specified ADF object.	The following objects are valid: ■ TAB
SENDKEY	Perform a send key action on the specified edit object.	The following objects are valid: ■ EDIT
SET	Defines the database table column.	The following objects are valid: ■ COLUMN ■ CONDITION
SETAPPTYPE	Set the application type. This is typically the first keyword specified in the component code.	The following application types are valid: ■ ADF ■ FORMFLEX ■ FORMS ■ JTT ■ TELNET ■ WEB
SETCURRENTROW	Set the current row.	No objects required.

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

Keywords	Description	Valid Objects
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.
SETTEXT	Set the text on the specified object.	The following objects are valid: ■ ADFUPLOADFILE ■ DATE ■ DYNAMICEDIT ■ EDIT ■ FIELD ■ PASSWORD ■ RESPONSEBOX ■ TEXTAREA
SETEXTAUTOCOMPLETE	Set the text on the specified edit object with auto complete.	The following objects are valid: ■ EDIT
SETVALUE	Set the value of the specified ADF combo list of values object.	The following objects are valid: ■ ADFCOMBOLOV
SETVAR	Set the value of the parameters of the User Defined Data type.	The following objects are valid: ■ BOOLEAN ■ CHAR ■ DATE ■ INT ■ INTEGER ■ LONG ■ NUMBER ■ OBJECT_TYPE ■ RECORD_TYPE ■ VARCHAR ■ VARCHAR2

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

Keywords	Description	Valid Objects
SETVARIN	Set the value of the parameter as an input parameter of the database procedure or function.	The following objects are valid: ■ BOOLEAN ■ CHAR ■ DATE ■ INT ■ INTEGER ■ LONG ■ NUMBER ■ OBJECT_TYPE ■ RECORD_TYPE ■ TABLE_TYPE ■ VARCHAR ■ VARCHAR2 ■ VARRAY_TYPE
SETVARINOUT	Set the value of the parameter as used as both input and output parameter of the database procedure or function.	The following objects are valid: ■ BOOLEAN ■ CHAR ■ DATE ■ INT ■ INTEGER ■ LONG ■ NUMBER ■ OBJECT_TYPE ■ RECORD_TYPE ■ TABLE_TYPE ■ VARCHAR ■ VARCHAR2 ■ VARRAY_TYPE
SETVAROUT	Set the value of the parameter as an output parameter of the database procedure or function.	The following objects are valid: ■ BOOLEAN ■ CHAR ■ DATE ■ INT ■ INTEGER ■ LONG ■ NUMBER ■ OBJECT_TYPE ■ RECORD_TYPE ■ TABLE_TYPE ■ VARCHAR ■ VARCHAR2 ■ VARRAY_TYPE

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

Keywords	Description	Valid Objects
SETWINDOW	Set the window.	No objects required.
SHOW	Show the specified ADF PanelAccordian object.	The following objects are valid: ■ ADFPANELACCORDIAN
SHOWDROPDOWN	Show the specified ADF PanelAccordian object.	The following objects are valid: ■ ADFCOMBOLOV
SORT	Sort the specified ADF table object.	The following objects are valid: ■ TABLE
STARTBLOCK	Specify the actual PLSQL procedure call Keyword set. Used with ENDBLOCK.	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.
STARTOBJECTTYPE	Specify the start of an Object Type Keyword set. Used with ENDOBJECTTYPE.	No objects required.
STARTOPTIONAL	Specify the start of an Optional Keyword set. Used with ENDOPTIONAL.	No objects required.
STARTQUERY	Specify the start the query block Keyword set. Query block is used to perform actions on database tables. Used with ENDQUERY.	No objects required.
STARTRECORDTYPE	Specify the start of a Record Type Keyword set. Used with ENDRECORDTYPE.	No objects required.
STARTRECOVERY	Specify the start of a Recovery Keyword set. Used with ENDRECOVERY.	No objects required.
STARTROWITERATOR	Specify the start of the iterate block Keyword set. Used with ENDROWITERATOR.	No objects required.
STARTTAB	Specify the start of a Tab Keyword set. Used with ENDTAB.	No objects required.
STARTTABLETYPE	Specify the end of a Table Type Keyword set. Used with ENDTABLETYPE.	No objects required.
STARTVARRAYTYPE	Specify the start of a VArray Type Keyword set. Used with ENDVARRAYTYPE.	No objects required.
STARTXLTBLVERIFY	Specify the start of an Excel table verify Keyword set. Used with ENDXLTBLVERIFY.	No objects required.

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

Keywords	Description	Valid Objects
STARTXLVERIFY	Specify the start of an Excel verify Keyword set. Used with ENDXLVERIFY.	No objects required.
UNCHECK	Uncheck the specified checkbox object.	The following objects are valid: ■ CHECKBOX
UNSELECT	Unselect the specified listbox object.	The following objects are valid: ■ ADFMANYLISTBOX ■ LIST ■ LISTBOX
UNSELECTTALL	Unselect all of the the specified checkbox or listbox objects.	The following objects are valid: ■ ADFMANYCHECKBOX ■ ADFMANYLISTBOX
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

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

Keywords	Description	Valid Objects
WAITFORPAGE	Wait for the specified ADF object.	The following objects are valid: ■ ADFBUTTON ■ ADFCOMBOLOV ■ ADFCOMMANDMENU ■ ADFMANYCHECKBOX ■ ADFMANYLISTBOX ■ ADFNAVIGATIONITEM ■ ADFNAVIGATIONPANE ■ ADFOUTPUTTEXT ■ ADFPANELACCORDIAN ■ ADFPANELBOX ■ ADFPANELHEADER ■ ADFPANELLABEL ■ ADFPANELSPLITTER ■ ADFPANELWINDOW ■ ADFTOOLBARBUTTON ■ ADFUPLOADFILE ■ CHECKBOX ■ DATE ■ DIALOG ■ EDIT ■ IMAGE ■ LINK ■ LIST ■ LISTBOX ■ MENU ■ RADIOBUTTON ■ TAB ■ TABLE ■ TOOLBAR ■ TREE ■ TREETABLE
WS-ENDSTRUCTURE	Specify the end the Web service structure Keyword set. Used with WS-STARTSTRUCTURE.	No objects required.
WS-NODENAME	Specify the node name and its path from the payload.	No objects required.
WS-NODENAMEWITHATTRIBUTE	Specify the node name with attributes of the node from the payload.	No objects required.
WS-OPERATIONNAME	Specify the operation to be executed in the WSDL.	No objects required.
WS-PARENT	Specify the parent node for a specific node from the payload.	No objects required.

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

Keywords	Description	Valid Objects
WS-PROCESSWSREQUEST	Specify the start the payload structure.	No objects required.
WS-SETWEBSERVICENAME	Specify the WSDL details for the specific payload.	No objects required.
WS-SETXMLELEMENT	Specify the node element.	No objects required.
WS-STARTSTRUCTURE	Specify the start the Web service structure Keyword set. Used with WS-ENDSTRUCTURE.	No objects required.

A.2 Web/EBS Forms Keywords and Objects

The following sections list the Keywords and valid objects available to use to specify component code for Web and EBS Forms applications.

A.2.1 ACTIVATE

This keyword is used to set focus on the popup dialog box in EBS forms application.

This keyword is used with these objects:

- Alert
- Choicebox
- Window

A.2.1.1 ALERT

The object type Alert for the Activate keyword is used to set focus on an Alert box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name - Name of the Alert box
- Attribute value - Name of the Alert box

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ACTIVATE	ALERT	Name of the Alert box.	Name of the Alert box
Example	ACTIVATE	ALERT	Caution	Caution

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Alert box	Select the option as True to generate code			
Example	Caution	True			

Reference Image

None.

A.2.1.2 CHOICEBOX

The object type Choice box for the Activate keyword is used to set focus on a Choice box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Choice box
- Attribute value – Name of the Choice box (Optional)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ACTIVATE	CHOICEBOX	Name of the Choice box	Name of the Choice box
Example	ACTIVATE	CHOICEBOX	Note	Note

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Choice box	Select the option as True to generate code			
Example	Note	True			

A.2.1.3 WINDOW

The object type Window for the Activate keyword is used to set focus on an EBS forms or flex window.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the window
- Attribute value – Attribute value of the window (Name property from the xpath). Attribute value is optional for flex window.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ACTIVATE	WINDOW	Name of the Window	Attribute value of the Window
Example	ACTIVATE	WINDOW	Line Payments	LINE_PAYMENTS

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Window	Select the option as True to generate code			
Example	Line Payments	True			

A.2.2 APPROVE

This keyword is used to acknowledge a popup dialog box.

This keyword is used with these objects:

- Alert
- Choicebox
- Flexwindow
- Responsebox

A.2.2.1 ALERT

The object type Alert for the Approve keyword is used to acknowledge an Alert box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Alert button
- Attribute Value – Name of the Alert button

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	APPROVE	ALERT	Name of the alert button	Name of the Alert button
Example	APPROVE	ALERT	OK	OK

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of Alert button	Select the option as True to generate code			
Example	Ok	True			

Reference Image

None.

A.2.2.2 CHOICEBOX

The object type Choicebox for the Approve keyword is used to acknowledge a Choice box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Choice box button
- Attribute Value – Name of the Choice box button

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	APPROVE	CHOICEBOX	Name of the Choice box button	Name of the Choice box button
Example	APPROVE	CHOICEBOX	Yes	Yes

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of Choice box button	Select the option as True to generate code			
Example	Yes	True			

Reference Image

None.

A.2.2.3 FLEXWINDOW

The object type Flexwindow for the Approve keyword is used to acknowledge a Flex window in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Flex window button

- Attribute Value – Name of the Flex window button

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	APPROVE	FLEXWINDOW	Name of the Flex window button	Name of the Flex window button
Example	APPROVE	FLEXWINDOW	Ok	Ok

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Flex window button	Select the option as True to generate code			
Example	Ok	True			

Reference Image

None.

A.2.2.4 RESPONSEBOX

The object type Responsebox for the Approve keyword is used to acknowledge a Response box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Response box button
- Attribute Value – Name of the Response box button

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	APPROVE	RESPONSEBOX	Name of the Response box button	Name of the Response box button
Example	APPROVE	RESPONSEBOX	Ok	Ok

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Response box button	Select the option as True to generate code			

	Display Name	Value 1	Value 2	Value 3	Value 4
Example	Ok	True			

Reference Image

None.

A.2.3 CANCEL Keyword

This keyword is used to perform a Cancel operation on an object.

This keyword is used with these objects:

- Alert
- Choicebox
- Flexwindow
- Responsebox

A.2.3.1 ALERT

The object type Alert for the Cancel keyword is used to cancel an Alert box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Alert button
- Attribute Value – Name of the Alert button

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CANCEL	ALERT	Name of the Alert button	Name of the Alert button
Example	CANCEL	ALERT	Cancel	Cancel

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Alert button	Select the option as True to generate code			
Example	Cancel	True			

Reference Image

None.

A.2.3.2 CHOICEBOX

The object type Choicebox for the Cancel Keyword is used to cancel a Choice box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Choice box button
- Attribute Value – Name of the Choice box button

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CANCEL	CHOICEBOX	Name of the Choice box button	Name of the Choice box button
Example	CANCEL	CHOICEBOX	No	No

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Choice box button	Select the option as True to generate code			
Example	No	True			

Reference Image

None.

A.2.3.3 FLEXWINDOW

The object type Flexwindow for the Cancel keyword is used to cancel a Flex window in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Flex window button
- Attribute Value – Name of the Flex window button

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CANCEL	FLEXWINDOW	Name of the Flex window button	Name of the Flex window button
Example	CANCEL	FLEXWINDOW	Cancel	Cancel

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Flex window button	Select the option as True to generate code			
Example	Cancel	True			

Reference Image

None.

A.2.3.4 RESPONSEBOX

The object type Responsebox for the Cancel keyword is used to cancel a Response box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Response box button
- Attribute Value – Name of the Response box button

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CANCEL	RESPONSEBOX	Name of the Response box button	Name of the Response box button
Example	CANCEL	RESPONSEBOX	Cancel	Cancel

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Response box button	Select the option as True to generate code			
Example	Cancel	True			

Reference Image

None.

A.2.4 CHECK

This keyword is used to perform a check or uncheck operation on a Check box.

A.2.4.1 CHECKBOX

The object type Checkbox for the Check keyword is used to perform a check or uncheck operation on a checkbox.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Check box
- Attribute value – Attribute value of the Check box (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CHECK	CHECKBOX	Name of the Checkbox	Attribute value of the Checkbox
Example	CHECK	CHECKBOX	Forward	PO_APPROVE_WF_CHECKBOX_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the checkbox	Select True or False			
Example	Forward	True			

Reference Image

None.

A.2.5 CLICK

This keyword is used to perform a click operation on an object.

This keyword is used with these objects:

- Button
- ADFButton
- AlertButton
- ChoiceboxButton
- Edit
- Element
- Flexcancel
- Flexcombination
- Flexok
- Image

- Link
- Tab
- Toolbar

A.2.5.1 BUTTON

The object type Button for the Click keyword is used to perform a click event on a Button.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the button
- Attribute value – Attribute value of the button (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following tablelists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	BUTTON	Name of the button	Attribute value of the button
Example	CLICK	BUTTON	Approve	Approve

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the button	Select True or False			
Example	Approve	True			

Reference Image

None.

A.2.5.2 ADFBUTTON

The object type ADFButton for the Click keyword is used to perform a click event on an ADFButton.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the ADFbutton
- Attribute value – Attribute value of the ADFbutton (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	ADFBUTTON	Name of the ADFButton	Attribute value of the ADFButton
Example	CLICK	ADFBUTTON	Add	r1:r1:pc1:cb8

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the button	Select True or False			
Example	Add	True			

Reference Image

None.

A.2.5.3 ALERTBUTTON

The object type AlertButton for the Click keyword is used to perform a click event on an Alert Dialog in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Alert Dialog button
- Attribute value – Name of the Alert Dialog button

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	ALERTBUTTON	Name of the Alert button	Name of the Alert button
Example	CLICK	ALERTBUTTON	OK	OK

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the button	Select True or False			
Example	OK	True			

Reference Image

None.

A.2.5.4 CHOICEBOXBUTTON

The object type ChoiceboxButton for the Click keyword is used to perform a click event on a Choice box button in EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Choice box button
- Attribute value – Name of the Choice box button.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	CHOICEBOXBUTTON	Name of the Choicebox button	Name of the Choicebox button
Example	CLICK	CHOICEBOXBUTTON	Yes	Yes

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the choicebox	Select True or False			
Example	Yes	True			

Reference Image

None.

A.2.5.5 EDIT

The object type Edit for the Click keyword is used to perform a click event on an Edit field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field
- Attribute value – Attribute value of the Edit field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	EDIT	Name of the Edit field	Attribute value of the Edit field

	Keyword	Object	Display Name	Attribute Value
Example	CLICK	EDIT	Results	ROUT_RESULTS_ CURRENT_ RECORD_ INDICATOR_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the edit field	Select True or False			
Example	Results	True			

Reference Image

None.

A.2.5.6 ELEMENT

The object type Element for the Click keyword is used to perform a click event on any web element.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Element.
- Attribute value – Complete Xpath of the Element.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	>CLICK	ELEMENT	Name of the Element	XPath of the Element
Example	CLICK	ELEMENT	Results	/web:window[@title ='Oracle Lease and Finance Management: Update Operational Options']/web:document[@index='0']/web:form[@id='DefaultFormName' or @name='DefaultFormName' or @index='0']/web:input_ checkbox[@id='DepreciateYN']

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Element	Select True or False			
Example	Results	True			

Reference Image

None.

A.2.5.7 FLEXCANCEL

The object type FlexCancel for the Click keyword is used to perform a click event on a Flex field cancel button in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Flex field Cancel button.
- Attribute value – None.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	FLEXCANCEL	Name of the Flexfield Cancel button	None
Example	CLICK	FLEXCANCEL	Cancel	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Flex field Cancel button	Select True or False			
Example	Cancel	True			

Reference Image

None.

A.2.5.8 FLEXCOMBINATION

The object type Flexcombination for the Click keyword is used to perform a click event on a combination button in a Flex window in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Flex Combination button.
- Attribute value – None.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	FLEXCOMBINATION	Name of the Flex Combination	None
Example	CLICK	FLEXCOMBINATION	Combination	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Flex Combination	Select True or False			
Example	Combination	True			

Reference Image

None.

A.2.5.9 FLEXOK

The object type FlexOk for the Click keyword is used to perform a click event on an OK button in a Flex Window in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the OK button in Flex window
- Attribute value – Name of the OK button in Flex window

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	FLEXOK	Name of the Flexfield OK button	None
Example	CLICK	FLEXOK	OK	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Flex field OK button	Select True or False			
Example	OK	True			

Reference Image

None.

A.2.5.10 IMAGE

The object type Image for the Click keyword is used to perform a click event on a Web image.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the image
- Attribute value – Attribute value of the image (properties from the xpath like alt, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	IMAGE	Name of the Image	Attribute value of the Image
Example	CLICK	IMAGE	Update	Update

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the image	Select True or False			
Example	Update	True			

Reference Image

None.

A.2.5.11 LINK

The object type Link for the Click keyword is used to perform a click event on a Web link.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the link
- Attribute value – Attribute value of the link (properties from the xpath like name, id, text, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	LINK	Name of the Link	Attribute value of the Link
Example	CLICK	LINK	Return to Contracts	Return to Contracts

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the link	Select True or False			
Example	Return to Contracts	True			

Reference Image

None.

A.2.5.12 TAB

The object type Tab for the Click keyword is used to perform a click event on a Tab in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Tab
- Attribute value – Attribute value of the Tab (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	TAB	Name of the Tab	Attribute value of the Tab
Example	CLICK	TAB	Main	ATTRIBUTE_ GROUPS_REGIONS

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the tab	Select True or False			
Example	Main	True			

Reference Image

None.

A.2.5.13 TOOLBAR

The object type Toolbar for the Click keyword is used to perform a click event on a Toolbar in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Toolbar button
- Attribute value – Attribute value of the EBS forms window (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLICK	TOOLBAR	Name of the Toolbar	Attribute value of the EBS forms window
Example	CLICK	TOOLBAR	Save	COST_PLUS_ STRUCTURE

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Toolbar	Select True or False			
Example	Save	True			

Reference Image

None.

A.2.6 CLOSE

This keyword is used to close the window.

A.2.6.1 WINDOW

The object type Window for the Close keyword is used to close the window.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Window
- Attribute value – Attribute value of the Window (properties from the xpath like name, id, title, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	CLOSE	WINDOW	Name of the Window	Attribute value of the Window
Example	CLOSE	WINDOW	Process Routing (*) Step Line	title='FPDS-NG : SRIDEVI_MCC [IDV]'

Test Data

None.

Reference Image

None.

A.2.7 COLLAPSE

This keyword is used to collapse a node in the tree in an EBS forms application.

A.2.7.1 TREE

The object type Tree for the Collapse keyword is used to collapse a node in the tree.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Tree
- Attribute value – Attribute value of the Tree (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	COLLAPSE	TREE	Name of the Tree	Attribute value of the Tree
Example	COLLAPSE	TREE	Recent Documents	APPTREE_NAV_ TREE_ NAVIGATOR_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Tree	Node name			
Example	Recent Documents	Organizations | AM1			

Reference Image

None.

A.2.8 COLLAPSENODE

This keyword is used to collapse a node in the tree in EBS forms application.

A.2.8.1 TREE

The object type Tree for the Collapsenode keyword is used to collapse a node in the tree.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Tree
- Attribute value – Attribute value of the Tree (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	COLLAPSENODE	TREE	Name of the Tree	Attribute value of the Tree
Example	COLLAPSENODE	TREE	Recent Documents	APPTREE_NAV_ TREE_ NAVIGATOR_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Tree	Node name			
Example	Recent Documents	Organizations | AM1			

Reference Image

None.

A.2.9 ENDCATCH

This keyword is used to end the Catch block.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDCATCH	None	None	None
Example	ENDCATCH	None	None	None

Test Data
None.

Reference Image
None.

A.2.10 ENDGROUP
This keyword is to end the Group block.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDGROUP	None	None	None
Example	ENDGROUP	None	None	None

Test Data
None.

Reference Image
None.

A.2.11 ENDITERATE
This keyword is used to end the Iterate block.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDITERATE	None	None	None

	Keyword	Object	Display Name	Attribute Value
Example	ENDITERATE	None	None	None

Test Data
None.

Reference Image
None.

A.2.12 ENDKEY
This keyword is used to end the Key block.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDKEY	None	None	None
Example	ENDKEY	None	None	None

Test Data
None.

Reference Image
None.

A.2.13 ENDOPTIONAL
This keyword is used to end the Optional block.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDOPTIONAL	None	None	None
Example	ENDOPTIONAL	None	None	None

Test Data
None.

Reference Image
None.

A.2.14 ENDRECOVERY
This keyword is used to end the Recovery block.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDRECOVERY	None	None	None
Example	ENDRECOVERY	None	None	None

Test Data
None.

Reference Image
None.

A.2.15 ENDTAB
This keyword is used to end the Tab block.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDTAB	None	None	None
Example	ENDTAB	None	None	None

Test Data
None.

Reference Image

None.

A.2.16 ENDXLTBLVERIFY

This keyword is used to end the block having table verification with Excel data.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDXLTBLVERIFY	None	None	None
Example	ENDXLTBLVERIFY	None	None	None

Test Data

None.

Reference Image

None.

A.2.17 ENDXLVERIFY

This keyword is used to end the block having verification with Excel data.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDXLVERIFY	None	None	None
Example	ENDXLVERIFY	None	None	None

Test Data

None.

Reference Image

None.

A.2.18 EXPANDNODE

This keyword is used to expand a node in the tree in an EBS forms application.

A.2.18.1 TREE

The object type Tree for the Expandnode keyword is used to expand a node in the tree.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Tree
- Attribute value – Attribute value of the Tree (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	EXPANDNODE	TREE	Name of the Tree	Attribute value of the Tree
Example	EXPANDNODE	TREE	Assets With Work	APPTREE_NAV_ TREE_ NAVIGATOR_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the tree	Node path			
Example	Assets With Work	Forecast Work | All Assets			

Reference Image

None.

A.2.19 FIREEVENTBLUR

This keyword is used to trigger blur event on the object in a Web application.

This keyword is used with these objects:

- Checkbox
- Edit
- List
- Listbox
- Radiobutton
- Textarea

A.2.19.1 CHECKBOX

The object type Checkbox for the Fireeventblur keyword is used to trigger the blur event on the checkbox.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Checkbox
- Attribute value – Attribute value of the Checkbox (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTBLUR	CHECKBOX	Name of the Checkbox	Attribute value of the Checkbox
Example	FIREEVENTBLUR	CHECKBOX	Apply To Lower Levels Flag	ApplyToLowerLevelsFlag

Test Data

None.

Reference Image

None.

A.2.19.2 EDIT

The object type Edit for Fireeventblur keyword is used to trigger blur event on the Edit field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field
- Attribute value – Attribute value of the Edit field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTBLUR	EDIT	Name of the Check box	Attribute value of the Check box
Example	FIREEVENTBLUR	EDIT	Sale Order	SchCompletionSONum

Test Data

None.

Reference Image

None.

A.2.19.3 LIST

The object type List for the Fireeventblur keyword is used to trigger a blur event on the List.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List
- Attribute value – Attribute value of the List (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTBLUR	LIST	Name of the LIST	Attribute value of the List
Example	FIREEVENTBLUR	LIST	Select	Select

Test Data

None.

Reference Image

None.

A.2.19.4 LISTBOX

The object type Listbox for the Fireeventblur keyword is used to trigger a blur event on the Listbox.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Listbox
- Attribute value – Attribute value of the Listbox (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTBLUR	LISTBOX	Name of the Listbox	Attribute value of the Listbox
Example	FIREEVENTBLUR	LISTBOX	Request Type	name='RequestType'

Test Data

None.

Reference Image

None.

A.2.19.5 RADIOBUTTON

The object type Radiobutton for the Fireeventblur keyword is used to trigger a blur event on the Radiobutton.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Radiobutton
- Attribute value – Attribute value of the Radiobutton (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTBLUR	RADIOBUTTON	Name of the Radiobutton	Attribute value of the Radiobutton
Example	FIREEVENTBLUR	RADIOBUTTON	Select	Select

Test Data

None.

Reference Image

None.

A.2.19.6 TEXTAREA

The object type Textarea for Fireeventblur keyword is used to trigger blur event on the Textarea.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Textarea
- Attribute value – Attribute value of the Textarea (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTBLUR	TEXTAREA	Name of the Text area	Attribute value of the Text area
Example	FIREEVENTBLUR	TEXTAREA	Description	description

Test Data

None.

Reference Image

None.

A.2.20 FIREEVENTONCHANGE

This keyword is used to trigger onchange event on the object. This keyword is used with these objects:

- Checkbox
- Edit
- List
- Listbox
- Radiobutton
- Textarea

A.2.20.1 CHECKBOX

The object type Checkbox for the Fireeventonchange keyword is used to trigger an onchange event on the Checkbox.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Checkbox
- Attribute value – Attribute value of the Checkbox (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTONCHANGE	CHECKBOX	Name of the Checkbox	Attribute value of the Checkbox
Example	FIREEVENTONCHANGE	CHECKBOX	Organization	orgSelectLov

Test Data

None.

Reference Image

None.

A.2.20.2 EDIT

The object type Edit for the Fireeventonchange keyword is used to trigger an onchange event on the Edit field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field

- Attribute value – Attribute value of the Edit field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTONCHANGE	EDIT	Name of the Edit field	Attribute value of the Edit field
Example	FIREEVENTONCHANGE	EDIT	Organization	orgSelectLov

Test Data

None.

Reference Image

None.

A.2.20.3 LIST

The object type List for the Fireeventonchange keyword is used to trigger an onchange event on the List.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List
- Attribute value – Attribute value of the List (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTONCHANGE	LIST	Name of the List	Attribute value of the List
Example	FIREEVENTONCHANGE	LIST	To Step	ToStep

Test Data

None.

Reference Image

None.

A.2.20.4 LISTBOX

The object type Listbox for the Fireeventonchange keyword is used to trigger an onchange event on the Listbox.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Listbox
- Attribute value – Attribute value of the Listbox (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTONCHANGE	LISTBOX	Name of the Listbox	Attribute value of the Listbox
Example	FIREEVENTONCHANGE	LISTBOX	Parent Element	Parent Element

Test Data

None.

Reference Image

None.

A.2.20.5 RADIOBUTTON

The object type Radiobutton for the Fireeventonchange keyword is used to trigger an onchange event on the Radiobutton.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Radiobutton
- Attribute value – Attribute value of the Radiobutton (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTONCHANGE	RADIOBUTTON	Name of the Radiobutton	Attribute value of the Radiobutton
Example	FIREEVENTONCHANGE	RADIOBUTTON	Organization	orgSelectLov

Test Data

None.

Reference Image

None.

A.2.20.6 TEXTAREA

The object type Textarea for the Fireeventonchange keyword is used to trigger an onchange event on the Textarea.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Textarea
- Attribute value – Attribute value of the Textarea (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	FIREEVENTONCHANGE	TEXTAREA	Name of the Textarea	Attribute value of the Textarea
Example	FIREEVENTONCHANGE	TEXTAREA	Item Description	ItemDescription

Test Data

None.

Reference Image

None.

A.2.21 FUNCTIONCALL

This keyword is used to execute a function specified in the object.

This keyword is used with these objects:

- cRMLIB
- eBSLibrary
- gENLIB
- pROCLIB
- pROJLIB
- sCMLIB
- tELNETLIB

A.2.21.1 cRMLIB

The object type cRMLIB for the Functioncall keyword is used to execute a function in the cRMLIB library.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the field where function call is made

- Attribute value – Attribute value of the field where function call is made (properties from the xpath like name, id etc). Providing attribute value depends on the function called in the 'Function Name' field.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Function Name
Usage	FUNCTIONCALL	cRMLIB	Name of the field	Attribute value of the field	Name of the crmlib function
Example	FUNCTIONCALL	cRMLIB	Configure	Configure	clickConfigure

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the field	Select True or False			
Example	Configure	True			

Reference Image

None.

A.2.21.2 eBSLibrary

The object type eBSLibrary for the Functioncall keyword is used to execute a function in the eBSLibrary.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the field where function call is made
- Attribute value – Attribute value of the field where function call is made (properties from the xpath like name, id etc). Providing an attribute value depends on the function called in the 'Function Name' field.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Function Name
Usage	FUNCTIONCALL	eBSLibrary			Name of the eBSLibrary function
Example	FUNCTIONCALL	eBSLibrary			Oracle_close_all_browsers

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	None	None			
Example	None	None			

Reference Image

None.

A.2.21.3 gENLIB

The object type gENLIB for the Functioncall keyword is used to execute a function in the gENLIB library.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the field where function call is made
- Attribute value – Attribute value of the field where function call is made (properties from the xpath like name, id etc). Providing an attribute value depends on the function called in the 'Function Name' field.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Function Name
Usage	FUNCTIONCALL	gENLIB	Name of the field	Name/Attribute value of the field	Name of the genlib function
Example	FUNCTIONCALL	gENLIB	Save	Save	webClickButton

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the field	Select True or False			
Example	Save	True			

Reference Image

None.

A.2.21.4 pROCLIB

The object type pROCLIB for the Functioncall keyword is used to execute a function in the pROCLIB library.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the field where function call is made

- Attribute value – Attribute value of the field where function call is made (properties from the xpath like name, id etc).Providing attribute value depends on the function called in the 'Function Name' field.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Function Name
Usage	FUNCTIONCALL	pROCLIB			Name of the proclib function
Example	FUNCTIONCALL	pROCLIB			encryptURL

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	None	None			
Example	None	None			

Reference Image

None.

A.2.21.5 pROJLIB

The object type pROJLIB for the Functioncall keyword is used to execute a function in the pROJLIB library.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the field where function call is made
- Attribute value – Attribute value of the field where function call is made (properties from the xpath like name, id etc).Providing attribute value depends on the function called in the 'Function Name' field.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Function Name
Usage	FUNCTIONCALL	pROJLIB			Name of the projlib function
Example	FUNCTIONCALL	pROJLIB			webImgVerfy Checkbox

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	None	None			
Example	None	None			

Reference Image

None.

A.2.21.6 sCMLIB

The object type sCMLIB for the Functioncall keyword is used to execute a function in the sCMLIB library.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the field where function call is made
- Attribute value – Attribute value of the field where function call is made (properties from the xpath like name, id etc). Providing attribute value depends on the function called in the 'Function Name' field.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Function Name
Usage	FUNCTIONCALL	sCMLIB			Name of the scmlib function
Example	FUNCTIONCALL	sCMLIB			selectTimeCardPeriod

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	None	None			
Example	None	None			

Reference Image

None.

A.2.21.7 tELNETLIB

The object type tELNETLIB for the Functioncall keyword is used to execute a function in the tELNETLIB library.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the field where function call is made

- Attribute value – Attribute value of the field where function call is made (properties from the xpath like name, id etc). Providing attribute value depends on the function called in the 'Function Name' field.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Function Name
Usage	FUNCTIONCALL	tELNETLIB	Name of the field	Attribute value of the field	Name of the telnetlib function
Example	FUNCTIONCALL	tELNETLIB			Close

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Display name of the field	Select True or False			
Example	Button	True			

Reference Image

None.

A.2.22 GET

This keyword is used to get the data from the specified object:

- Alert
- Choicebox
- Edit
- Field
- List
- Listbox
- Status
- Text
- Textarea

A.2.22.1 ALERT

The object type Alert for the Get keyword is used to get the Alert message from an EBS forms Alert Dialog in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Alert
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GET	ALERT	Name of the Alert Dialog	None	Output variable
Example	GET	ALERT	Alert	None	alertMsg

Test Data

None.

Reference Image

None.

A.2.22.2 CHOICEBOX

The object type Choicebox for the Get keyword is used to get the Alert message from an EBS forms Choice box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Choice box
- Attribute value –None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GET	CHOICEBOX	Name of the Choice box	None	Output variable
Example	GET	CHOICEBOX	Note	None	choiceMsg

Test Data

None.

Reference Image

None.

A.2.22.3 EDIT

The object type Edit for the Get keyword is used to get the value from an Edit field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field
- Attribute value – Attribute value of the Edit field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GET	EDIT	Name of the Edit field	Attribute value of the Edit field	Output variable
Example	GET	EDIT	Operating Unit	Operatingunit	opUnit

Test Data

None.

Reference Image

None.

A.2.22.4 FIELD

The object type Field for the Get keyword is used to get the value from EBS telent for the specified field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	GET	FIELD	None	None
Example	GET	FIELD	None	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Field Name	Output Variable for Field Value			
Example	LPN	subLPN			

Reference Image

None.

A.2.22.5 LIST

The object type List for the Get keyword is used to get the selected value from a List.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List
- Attribute value – Attribute value of the List (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GET	LIST	Name of the List	Attribute value of the List	Output variable
Example	GET	LIST	Destination Type	PO_ DISTRIBUTIONS_ DESTINATION_TYPE_0	destType

Test Data

None.

Reference Image

None.

A.2.22.6 LISTBOX

The object type ListBox for the Get keyword is used to get the selected value from a Listbox.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List box.
- Attribute value – Attribute value of the List box (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GET	LISTBOX	Name of the List box	Attribute value of the List box	Output variable
Example	GET	LISTBOX	Actions	ActionListTop	actions

Test Data

None.

Reference Image

None.

A.2.22.7 STATUS

The object type Status for the Get keyword is used to get the StatusBar message in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GET	STATUS	None	None	Output variable
Example	GET	STATUS	None	None	statusMsg

Test Data

None.

Reference Image

None.

A.2.22.8 TEXT

The object type Text for the Get keyword is used to get the text available between two texts from the specified area in an EBS Web application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Text
- Attribute value – Attribute should be provided as follows:
  - Before text should be provided as @before='Before Text'
  - After text should be provided as @afer='After Text'

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GET	TEXT	Name of the Text	Text location	Output variable

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Example	GET	TEXT	Auction Number	@before='Auction ',@after=' has been'	auctionNumber

Test Data

None.

Reference Image

None.

A.2.22.9 TEXTAREA

The object type TextArea for the Get keyword is used as follows:

- Get the value in text field in an EBS forms application
- Get the value in the Text Area in aWeb application

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the TextField/TextArea.
- Attribute value – Attribute value of the TextField/TextArea (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GET	TEXTAREA	>Name of the TextField/TextArea	Attribute value of the TextField/TextArea	Output variable
Example	GET	TEXTAREA	Description	auctDescription	description

Test Data

None.

Reference Image

None.

A.2.23 GETATTRIBUTE

This keyword is used to get the specified attribute of the following objects:

- Button
- Checkbox
- Edit

- List
- LOV
- Radiobutton
- Window

A.2.23.1 BUTTON

The object type Button for the GetAttribute keyword is used to get the specified attribute value of a button.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the button
- Attribute value – Attribute value of the button (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GETATTRIBUTE	BUTTON	Name of the button	Attribute value of the button	Output variable
Example	GETATTRIBUTE	BUTTON	Apply	ApplyButton	applyProperty

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the button	Provide attribute name			
Example	Apply	Name			

Reference Image

None.

A.2.23.2 CHECKBOX

The object type Checkbox for the GetAttribute keyword is used to get the specified attribute value of a Checkbox.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name/Label of the Checkbox
- Attribute value – Attribute value of the Checkbox (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GETATTRIBUTE	CHECKBOX	Name of the Check box	Attribute value of the Check box	Output variable
Example	GETATTRIBUTE	CHECKBOX	Consumption Advice	FIND_CONSIGNED_CONSUMPTION_FLAG_0	Consumption Property

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Check box	Provide attribute name			
Example	Consumption Advice	Name			

Reference Image

None.

A.2.23.3 EDIT

The object type Edit for the GetAttribute keyword is used to get the specified attribute value of an Edit field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field
- Attribute value – Attribute value of the Edit (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GETATTRIBUTE	EDIT	Name of the Edit field	Attribute value of the Edit field	Output variable
Example	GETATTRIBUTE	EDIT	Amount	name='amount'	AmountProperty

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Edit field	Provide attribute name			
Example	Amount	value			

Reference Image

None.

A.2.23.4 LIST

The object type List for the GetAttribute keyword is used to get the specified attribute value of a list in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List
- Attribute value – Attribute value of the List (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GETATTRIBUTE	LIST	Name of the List	Attribute value of the list	Output variable
Example	GETATTRIBUTE	LIST	Destination Type	PO_ DISTRIBUTIONS_ DESTINATION_TYPE_0	destinationTypeProperty

Test Data

None.

Reference Image

None.

A.2.23.5 LOV

The object type LOV for the GetAttribute keyword is used to get the specified attribute value of a Select Box/List Of Values.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Select box / List of values
- Attribute value – Attribute value of the Select box / List of Values (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GETATTRIBUTE	LOV	Name of the LOV	Attribute value of the LOV	Output variable
Example	GETATTRIBUTE	LOV	Organization	Organization	OrganizationProperty

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the LOV	Provide attribute name			
Example	Organization	value			

Reference Image

None.

A.2.23.6 RADIOBUTTON

The object type Radio button for the GetAttribute keyword is used to get the specified attribute value of a radio button.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name/Label of the Radio button
- Attribute value – Attribute value of the Radio button (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GETATTRIBUTE	RADIOBUTTON	Name of the Radio button	Attribute value of the Radio button	Output variable
Example	GETATTRIBUTE	RADIOBUTTON	Select	id='select'	SelectProperty

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Radio button	Provide attribute name			

	Display Name	Value 1	Value 2	Value 3	Value 4
Example	Select	value			

Reference Image

None.

A.2.23.7 WINDOW

The object type Window for the GetAttribute keyword is used to get the specified attribute value of the window.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Window
- Attribute value – Attribute value of the Window (properties from the xpath like name, id, text, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GETATTRIBUTE	WINDOW	Name of the Window	Attribute value of the Window	Output variable
Example	GETATTRIBUTE	WINDOW	Agreements	title='Agreements'	AgreementProperty

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the window	Provide attribute name			
Example	Agreements	text			

Reference Image

None.

A.2.24 GETITEMVALUES

This keyword is used to get the value of a List in an EBS forms application.

A.2.24.1 LIST

The object type List for the GetItemValues keyword is used to get the value of the selected item in the EBS forms list.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List
- Attribute value – Attribute value of the List (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	GETITEMVALUES	LIST	Name of the List	Attribute value of the list	Output variable
Example	GETITEMVALUES	LIST	Destination Type	PO_ DISTRIBUTIONS_ DESTINATION_TYPE_0	destinationType

Test Data

None.

Reference Image

None.

A.2.25 INVOKESOFTKEY

This keyword is used to invoke soft key like 'NEXT_FIELD' in an EBS forms application.

This keyword is used with these objects:

- Edit
- Spreadtable

A.2.25.1 EDIT

The object type Edit for the InvokeSoftKey keyword is used to invoke a soft key such as 'NEXT_FIELD' on the edit field in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field
- Attribute value – Attribute value of the Edit field (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	INVOKESOFTKEY	EDIT	Name of the Edit field	Attribute value of Edit field
Example	INVOKESOFTKEY	EDIT	Status	name='status'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Edit field	Provide Soft key			
Example	Status	'NEXT_FIELD'			

Reference Image

None.

A.2.25.2 SPREADTABLE

The object type Spread table for the InvokeSoftKey keyword is used to invoke a soft key such as 'NEXT_FIELD' on the Spread Table in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Spread Table
- Attribute value – Attribute value of the Spread Table (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	INVOKESOFTKEY	SPREADTABLE	Name of the Spread table	Attribute of the Spread table
Example	INVOKESOFTKEY	SPREADTABLE	Lots	RESULTS_GRID_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Spread table	Provide Soft key			
Example	Lots	NEXT_FIELD			

Reference Image

None.

A.2.26 LAUNCH

This keyword is used to launch the browser.

A.2.26.1 BROWSER

The object type Browser for the Launch keyword is used to launch the browser.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name –None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	LAUNCH	BROWSER	None	None
Example	LAUNCH	BROWSER	None	None

Test Data

None.

Reference Image

None.

A.2.27 MAXIMIZE

This keyword is used to maximize the browser.

A.2.27.1 WINDOW

The object type Window for the Maximize keyword is used to maximize the browser window.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the window
- Attribute value – Attribute value of the Window (properties from the xpath like title)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	MAXIMIZE	WINDOW	Name of the window	Attribute value of the window
Example	MAXIMIZE	WINDOW	iStore	iStore

Test Data

None.

Reference Image

None.

A.2.28 MAXVISIBLELINES

This keyword is used to provide the number of the visible lines in the forms table in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – Number of visible lines in the EBS forms application

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	MAXVISIBLELINES	None	None	No. of lines visible in the forms table
Example	MAXVISIBLELINES	None	None	5

Test Data

None.

Reference Image

None.

A.2.29 MENUSELECT

This keyword is used to select the specified menu item in an EBS forms application.

This keyword is used with these objects:

- Contextmenu
- Mainmenu
- Tree

A.2.29.1 CONTEXTMENU

The object type Context menu for the MenuSelect keyword is used to activate the pop-up menu of the Oracle Forms text field object and select the specified menu item in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Text field
- Attribute value – Attribute value of the Text field (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	MENUSELECT	CONTEXTMENU	Name of the Text field	Attribute value of the Text field
Example	MENUSELECT	CONTEXTMENU	Context Menu	ORDER_ OPERATING_ UNIT_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Text field	Menu Item			
Example	Context Menu	Schedule			

Reference Image

None.

A.2.29.2 MAINMENU

The object type Main Menu for the MenuSelect keyword is used to select the specified menu in an EBS forms window in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Menu to select in the EBS forms window
- Attribute value – Menu to select in the EBS forms window

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	MENUSELECT	MAINMENU	Menu	Menu
Example	MENUSELECT	MAINMENU	View | Query By Example | Run	View | Query By Example | Run

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Menu	Select True or False			
Example	View | Query By Example | Run	True			

Reference Image

None.

A.2.29.3 TREE

The object type Tree for the MenuSelect keyword is to performs the following in an EBS forms application.

- Selects the node in the EBS forms tree
- Activates the popup menu on the selected node
- Selects the menu item in the popup

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the EBS forms tree
- Attribute value – Attribute value of the EBS forms tree (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	MENUSELECT	TREE	Name of the EBS forms tree	Attribute value of the EBS forms tree
Example	MENUSELECT	TREE	Menu for Add Planning Node	APPTREE_NAV_ TREE_ NAVIGATOR_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the EBS forms tree	Tree Node path	Menu item in the popup		
Example	Menu for Add Planning Node	Spares Planning | Warehouses	Add Subinventory		

Reference Image

None.

A.2.30 MINIMIZE

This keyword is used to minimize the browser

A.2.30.1 WINDOW

The object type Window for the Minimize keyword is used to minimize the browser window.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Web Window

- Attribute value – Attribute value of the Web window (properties from the xpath like title)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	MINIMIZE	WINDOW	Name of the Window	Attribute value of the Window
Example	MINIMIZE	WINDOW	Auction	Auction

Test Data

None.

Reference Image

None.

A.2.31 PRESSTABKEY

This keyword is used to send the press tab key event on the Edit field.

A.2.31.1 EDIT

The object type Edit for the PressTabKey keyword is used to send the press tab key event on the Edit field

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name - Name of the Edit field
- Attribute value - Attribute value of the Edit field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	PRESSTABKEY	EDIT	Name of the Edit field	Attribute value of the Edit field.
Example	PRESSTABKEY	EDIT	User Name	usernameField

Test Data

None.

Reference Image

None.

A.2.32 SEARCHBYDYNAMICCOLUMN

This keyword is used to find the row number in a table using table column and it's value which are to be provided in the testdata.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Table Name
- Attribute value – None.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SEARCHBYDYNAMICCOLUMN	None	Table Name	None
Example	SEARCHBYDYNAMICCOLUMN	None	Awards Table	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Table Name	Table Column Name	Column Value		
Example	Awards Table	Supplier	Grainger		

Reference Image

None.

A.2.33 SEARCHCOLUMN

This keyword is used to find the row number in a table using table column and it's value.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Column Name
- Attribute value – Column Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SEARCHCOLUMN	None	Name of Column	Name of the Column
Example	SEARCHCOLUMN	None	Requisition	Requisition

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Column	Column Value			
Example	Requisition	1234			

Reference Image

None.

A.2.34 SEARCHEMPTYROW

This keyword is used to find the empty row number in a table using table name and it's column.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Column Name
- Attribute value – Column Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SEARCHEMPTYROW	None	Name of Column	Name of the Column
Example	SEARCHEMPTYROW	None	Requisition	Requisition

Test Data

None.

Reference Image

None.

A.2.35 SELECT

This keyword is used to perform a Select operation on the specified object.

This keyword is used with these objects:

- List
- Listbox
- Radiobutton
- Treelist

A.2.35.1 LIST

The object type List for the Select keyword is used to select the specified value from the list.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List
- Attribute value – Attribute value of the List (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SELECT	LIST	Name of the List	Attribute value of the List
Example	SELECT	LIST	Class	GROUPS_SYSTEM_LINKAGE_FUNCTION_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the List	Value to Select			
Example	Class	Task Owning Org			

Reference Image

None.

A.2.35.2 LISTBOX

The object type List Box for the Select keyword is used to select the specified value from the list box.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List box
- Attribute value - Attribute value of the List Box (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SELECT	LISTBOX	Name of the List box	Attribute value of the List box
Example	SELECT	LISTBOX	Class	GROUPS_SYSTEM_LINKAGE_FUNCTION_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the List box	Value to Select			
Example	Class	Task Owning Org			

Reference Image

None.

A.2.35.3 RADIOBUTTON

The object type Radio Button for the Select keyword is used to select the radio button.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name/Label of the Radio button
- Attribute value - Attribute value of the Radio button (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SELECT	RADIOBUTTON	Name of the Radio button	Attribute value of the Radio button
Example	SELECT	RADIOBUTTON	Alphanumeric	name='AWARD_ SETUP_MANUAL_ AWARD_NUM_ TYPE_ ALPHANUMERIC_ 0'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Radio button	Value to Select			
Example	Alphanumeric	True			

Reference Image

None.

A.2.35.4 TREELIST

The object type TreeList for the Select keyword is used to select item in an EBS forms TreeList in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the TreeList
- Attribute value – Attribute value of the TreeList (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SELECT	TREELIST	Name of the TreeList	Attribute value
Example	SELECT	TREELIST	Navigator	NAVIGATOR_LIST_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the TreeList	Tree List Item			
Example 1	Navigator	Flexfield | Key			
Example 2	Navigator	Contract Groups			

Reference Image

None.

A.2.36 SELECTALLROWS

This keyword is used to select all the rows in a Spread table.

A.2.36.1 SPREADTABLE

The object type Spread Table for the SelectAllRows keyword is used to select all the rows in a Spread table in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Table Name
- Attribute value – Attribute value of the Spread Table (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SELECTALLROWS	SPREADTABLE	Table Name	Attribute value of the Spread Table

	Keyword	Object	Display Name	Attribute Value
Example	SELECTALLROWS	SPREADTABLE	Lots	RESULTS_GRID_0

Test Data

None.

Reference Image

None.

A.2.37 SELECTNODE

This keyword is used to select the specified node of an EBS forms tree in an EBS forms application.

A.2.37.1 TREE

The object type Tree for the SelectNode keyword is used to select the specified node of an EBS forms tree in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Tree
- Attribute value – Attribute value of the Tree (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SELECTNODE	TREE	Name of the Tree	Attribute value of the Tree
Example	SELECTNODE	TREE	Warehouse Node Path	APPTREE_NAV_ TREE_ NAVIGATOR_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the node	Enter the node path to select			
Example	Warehouse Node Path	Spares Planning | War ehouses			

Reference Image

None.

A.2.38 SELECTROW

This keyword is used to select the specified row in EBS forms spread table.

A.2.38.1 SPREADTABLE

The object type SpreadTable for the SelectRow keyword is used to select specified row in an EBS forms spread table.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SELECTROW	SPREADTABLE	None	None
Example	SELECTROW	SPREADTABLE	None	None

Test Data

None.

Reference Image

None.

A.2.39 SENDKEY

This keyword is used to click and set the text in a text field.

A.2.39.1 EDIT

The object type Edit for the SendKey keyword is used to click and set the text in a text field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the text field label
- Attribute value – Attribute value of the Text field (properties from the xpath like id, name, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SENDKEY	EDIT	Name of Text field Label	Attribute value of the Text field
Example	SENDKEY	EDIT	Attribute	Attribute_Name

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Text field	Text field Value			
Example	Attribute	Red			

Reference Image

None.

A.2.40 SETAPPTYPE

This keyword is used to set the Application type.

This keyword is used with these objects:

- ADF
- Formflex
- Forms
- JTT
- PLSQL_OI
- Telnet
- Web

A.2.40.1 ADF

The object type ADF for the SetAppType keyword is used to set the application type as ADF.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETAPPTYPE	ADF	None	None
Example	SETAPPTYPE	ADF	None	None

Test Data

None.

Reference Image

None.

A.2.40.2 FORMFLEX

The object type FORMFLEX for the SetAppType keyword is used to set the application type as EBS forms Flex in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETAPPTYPE	FORMFLEX	None	None
Example	SETAPPTYPE	FORMFLEX	None	None

Test Data

None.

Reference Image

None.

A.2.40.3 FORMS

The object type Forms for the SetAppType keyword is used to set the application type as EBS forms in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETAPPTYPE	FORMS	None	None
Example	SETAPPTYPE	FORMS	None	None

Test Data

None.

Reference Image

None.

A.2.40.4 JTT

The object type JTT for the SetAppType keyword is used to set the application type as Wireless Application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETAPPTYPE	JTT	None	None
Example	SETAPPTYPE	JTT	None	None

Test Data

None.

Reference Image

None.

A.2.40.5 PLSQL_OI

The object type PLSQL_OI for the SetAppType keyword is used to set the application type as PLSQL or Open Interface.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETAPPTYPE	PLSQL_OI	None	None
Example	SETAPPTYPE	PLSQL_OI	None	None

Test Data

None.

Reference Image

None.

A.2.40.6 TELNET

The object type Telnet for the SetAppType keyword is used to set the application type as Mobile.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETAPPTYPE	TELNET	None	None
Example	SETAPPTYPE	TELNET	None	None

Test Data

None.

Reference Image

None.

A.2.40.7 WEB

The object type WEB for the SetAppType keyword is used to set the application type as Web Application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETAPPTYPE	WEB	None	None
Example	SETAPPTYPE	WEB	None	None

Test Data

None.

Reference Image

None.

A.2.41 SETCURRENTROW

This keyword is used to set the row number with the row number which was returned by component code specified above this Setcurrentrow keyword.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETCURRENTROW	None	None	None
Example	SETCURRENTROW	None	None	None

Test Data

None.

Reference Image

None.

A.2.42 SETFOCUS

This keyword is used to set focus on specified objects.

This keyword is used with these objects:

- Edit
- Firstrecord
- Textarea
- Treelist

A.2.42.1 EDIT

The object type Edit for the SetFocus keyword is used to set focus on an Edit field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field
- Attribute value – Attribute value of the Edit field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETFOCUS	EDIT	Name of the Edit field	Attribute value of the Edit field.
Example	SETFOCUS	EDIT	User Name	usernameField

Test Data

None.

Reference Image

None.

A.2.42.2 FIRSTRECORD

The object type FirstRecord for the SetFocus keyword is used to set focus on first record in the window by selecting menu 'View | Record | First' in the EBS forms window in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the EBS forms Window (Optional)
- Attribute value – Attribute value of the EBS forms Window (properties from the xpath like name, id, etc.) (Optional)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETFOCUS	FIRSTRECORD	Name of the Window	Attribute value of the Window
Example	SETFOCUS	FIRSTRECORD	Lookup Codes	FND_LOOKUP_ VALUES_LOOKUP_ CODE_0

Test Data

None.

Reference Image

None.

A.2.42.3 TEXTAREA

The object type Textarea for the SetFocus keyword is used to set focus on the following:

- Set focus on Textfield in an EBS forms application
- Set focus on Textarea in a Web application

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the TextField or TextArea

- Attribute value – Attribute value of the TextField or TextArea (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETFOCUS	TEXTAREA	Name of the TextField/TextArea	Attribute value of the TextField/TextArea
Example	SETFOCUS	TEXTAREA	Description	description

Test Data

None.

Reference Image

None.

A.2.42.4 TREELIST

The object type Tree List for the SetFocus keyword is used to set focus on the specified item in an EBS Forms tree list in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Tree list
- Attribute value – Attribute value of the Tree list (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETFOCUS	TREELIST	Name of the Tree list	Attribute value of the Tree list
Example	SETFOCUS	TREELIST	Navigator	NAVIGATOR_LIST_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Tree list	Tree List Item			
Example	Navigator	Flexfield|Key			

Reference Image

None.

A.2.43 SETLINE

This keyword is used to set the row number of the table to perform further actions or verifications on the specified row.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Table Column
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETLINE	None	Column Name	None
Example	SETLINE	None	Item Line	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Column Name	Enter the line number			
Example	Item Line	2			

Reference Image

None.

A.2.44 SETSPREADTABLE

This keyword is used to identify an EBS forms spread table.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the EBS forms Spread Table
- Attribute value – Attribute value of the Spread table (properties from the xpath like name)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETSPREADTABLE	None	Spread Table Name	Attribute value of the spread table.
Example	SETSPREADTABLE	None	Assignee	TASK_GRID_TASK_BEAN_0

Test Data

None.

Reference Image

None.

A.2.45 SETTABLENAME

This keyword is used to identify the table by providing one of the column names as the table name.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Table Column
- Attribute value – Name of the Table Column

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETTABLENAME	None	Column Name	Column Name
Example	SETTABLENAME	None	Select	Select

Test Data

None.

Reference Image

None.

A.2.46 SETTEXT

This keyword is used to set text in the specified Object field.

This keyword is used with these objects:

- Dynamicedit
- Edit
- Password
- Responsebox
- Textarea

A.2.46.1 DYNAMICEDIT

The object type Dynamic Edit for the SetText keyword is used to set the value in EBS Flex Fields in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Flex field

- Attribute value – Attribute value should be provided as specified below:
  - Flex field Label name as @label = 'Flex Field Label'
  - Flex field Column Name as @column = 'Column Name'

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETTEXT	DYNAMICEDIT	Name of the Flex field	Flex filed Row, Column values
Example	SETTEXT	DYNAMICEDIT	Item Low	@label='Item',@column='Low'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Flex field	Value			
Example	Item Low	GMF_STD_ING1			

Reference Image

None.

A.2.46.2 EDIT

The object type Edit for the SetText keyword is used to set text in an edit field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field
- Attribute value – Attribute value of the Edit field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETTEXT	EDIT	Name of the Edit field	Attribute value of the Edit field.
Example	SETTEXT	EDIT	User Name	usernameField

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the edit field	Enter the value to set in the edit field			
Example	User Name	operations			

Reference Image

None.

A.2.46.3 FIELD

The object type Field for the SetText keyword is used to set the text in the EBS Telnet Service.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETTEXT	FIELD	None	None
Example	SETTEXT	FIELD	None	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Telnet Field	Enter the value to set in the telnet window.			
Example	Telnet Field	PR4			

Reference Image

None.

A.2.46.4 PASSWORD

The object type Password for the SetText keyword is used to set the password in a password field in an EBS web application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Password field

- Attribute value – Attribute value of the Password field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETTEXT	PASSWORD	Name of the Password field	Attribute value of the Password field.
Example	SETTEXT	PASSWORD	Password	passwordField

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Password field	Enter the value to set in the password field			
Example	Password	welcome			

Reference Image

None.

A.2.46.5 RESPONSEBOX

The object type ResponseBox for the SetText keyword is used to set the text in EBS Forms Response box in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Response box field
- Attribute value – Name of the Response box field

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETTEXT	RESPONSEBOX	Name of the Response box field	Name of the Response box field
Example	SETTEXT	RESPONSEBOX	Block	Block

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Response box field	Value to enter in Response box			

	Display Name	Value 1	Value 2	Value 3	Value 4
Example	Block	Authorize First Installment			

Reference Image

None.

A.2.46.6 TEXTAREA

The object type Textarea for the SetText keyword is used to set text in the following objects.

- Textfield in an EBS Forms application
- Textarea in an EBS Web application

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the TextField or TextArea
- Attribute value – Attribute value of the TextField or TextArea (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETTEXT	TEXTAREA	Name of the TextField/TextArea	Attribute value of the TextField/TextArea
Example	SETTEXT	TEXTAREA	Contact Information	ContactInfo

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the TextField/TextArea	Enter the value to set in the textarea field			
Example	Contact Information	Test info			

Reference Image

None.

A.2.47 SETWINDOW

This keyword is used to set the window so that all the elements following this keyword will use the window name specified using Setwindow.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Window
- Attribute value – Attribute value of the Window (properties from the xpath like name, title, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETWINDOW	None	Name of the window	Attribute value of the window
Example 1	SETWINDOW	None	Agreements	Agreements
Example 2	SETWINDOW	None	Awards	title='Awards'
Example 3	SETWINDOW	None	Implementation Options	name='OPTIONS'

Test Data

None.

Reference Image

None.

A.2.48 STARTCATCH

This keyword is used to start the Catch block.

- It should be used within the STARTRECOVERY block.
- It will handle the exceptions raised by actions specified in the recovery block.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTCATCH	None	None	None
Example	STARTCATCH	None	None	None

Test Data

None.

Reference Image

None.

A.2.49 STARTGROUP

This keyword is used to start the Group block.

- Group block is used to group a set of actions.
- All the group actions depend upon the value provided for the first action in the Group block.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTGROUP	None	None	None
Example	STARTGROUP	None	None	None

Test Data

None.

Reference Image

None.

A.2.50 STARTITERATE

This keyword is used to start the Iterate block. Iterate block is used to perform actions on tables.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTITERATE	None	None	None
Example	STARTITERATE	None	None	None

Test Data

None.

Reference Image

None.

A.2.51 STARTKEY

This keyword is used to start the Key block.

- A Key block can have only one action to be performed.
- A Key block should be available within an Optional block.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the action to be performed in the Key block
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTKEY	None	Name of the action to be performed in Key block	None
Example	STARTKEY	None	File|Save	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the action to be performed in Key block	Provide True or False			
Example	File|Save	True			

Reference Image

None.

A.2.52 STARTOPTIONAL

This keyword is used to start the Optional block. Actions defined in the Optional block will be performed based on the data provided in the Test Data for Startkey.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTOPTIONAL	None	None	None

	Keyword	Object	Display Name	Attribute Value
Example	STARTOPTIONAL	None	None	None

Test Data

None.

Reference Image

None.

A.2.53 STARTRECOVERY

This keyword is used to start the Recovery block. A Recovery block is used to handle an action which might cause an exception.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTRECOVERY	None	None	None
Example	STARTRECOVERY	None	None	None

Test Data

None.

Reference Image

None.

A.2.54 STARTTAB

This keyword is used to start the Tab block. A Tab block is the block where code will be generated for click tab and data elements when a value is provided for any one field in the tab block.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTTAB	None	None	None
Example	STARTTAB	None	None	None

Test Data
None.

Reference Image
None.

A.2.55 STARTXLTBLVERIFY
This keyword is used to verify the EBS web/forms table data with Excel data.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTXLTBLVERIFY	None	None	None
Example	STARTXLTBLVERIFY	None	None	None

Test Data
None.

Reference Image
None.

A.2.56 STARTXLVERIFY
This keyword is used to verify the data in the Web page with Excel data.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTXLVERIFY	None	None	None

	Keyword	Object	Display Name	Attribute Value
Example	STARTXLVERIFY	None	None	None

Test Data

None.

Reference Image

None.

A.2.57 UNCHECK

This keyword is used to perform uncheck operation on a checkbox.

A.2.57.1 CHECKBOX

The object type Checkbox for the Uncheck keyword is used to perform an uncheck operation on a checkbox.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Checkbox
- Attribute value – Attribute value of the Checkbox (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	UNCHECK	CHECKBOX	Name of the Checkbox	Attribute value of the Checkbox
Example	UNCHECK	CHECKBOX	Print	PO_APPROVE_ PRINT_CHECK_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the checkbox	Enter True or False			
Example	Print	True			

Reference Image

None.

A.2.58 VERIFY

This keyword is used to perform verify operation on specified objects.

This keyword is used with these objects:

- Checkbox
- Choicebox
- Edit
- Image
- Link
- List
- Listbox
- Radiobutton
- Spreadcell
- Statusbar
- Text
- Textarea

A.2.58.1 CHECKBOX

The object type Checkbox for the Verfiy keyword is used to verify a checkbox is checked or unchecked.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Checkbox
- Attribute value – Attribute value of the Checkbox (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	CHECKBOX	Name of the Checkbox	Attribute value of the Checkbox
Example	VERIFY	CHECKBOX	Print	PO_APPROVE_ PRINT_CHECK_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the checkbox	Enter True or False			
Example	Print	True			

Reference Image

None.

A.2.58.2 CHOICEBOX

The object type Choicebox for the Verify keyword is used to verify a choice box Alert message in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Choicebox
- Attribute value – Name of the Choicebox

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	CHOICEBOX	Name of the Choicebox	Name of the Choicebox
Example	VERIFY	CHOICEBOX	Note	Note

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the choice box	Note			
Example	Note	Transaction complete.			

Reference Image

None.

A.2.58.3 EDIT

The object type Edit for the Verify keyword is used to verify a value in an edit field.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field
- Attribute value – Attribute value of the Edit field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	EDIT	Name of the Edit field	Attribute value of the edit field
Example	VERIFY	EDIT	Supplier Site	PO_HEADERS_ VENDOR_SITE_ CODE_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the edit field	Enter the text to verify			
Example	Supplier Site	GE			

Reference Image

None.

A.2.58.4 LINK

The object type Link for the Verify keyword is used to verify link text in an EBS web application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Link
- Attribute value – Attribute value of the Link (properties from the xpath like text, name, id etc.)

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	LINK	Name of the Link	Attribute value of the Link
Example	VERIFY	LINK	Line	Item 1 description

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the link	Value of the link to verify			
Example	Line	Item 1 description			

Reference Image

None.

A.2.58.5 LIST

The object type List for the Verify keyword is used to verify a selected item in the List.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List

- Attribute value – Attribute value of the List (properties from the xpath like text, name, id etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	LIST	Name of the List	Attribute value of the List
Example	VERIFY	LIST	Destination Type	PO_ DISTRIBUTIONS_ DESTINATION_ TYPE_0

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the list	Value in the list to verify			
Example	Destination Type	Inventory			

Reference Image

None.

A.2.58.6 LISTBOX

The object type ListBox for the Verify keyword is used to verify a selected item in the Listbox.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Listbox
- Attribute value – Attribute value of the Listbox (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	LISTBOX	Name of the List box	Attribute value of the list box
Example	VERIFY	LISTBOX	Actions	ActionListTop

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the list box	Value in the listbox to verify			
Example	Actions	Duplicate			

Reference Image

None.

A.2.58.7 RADIOBUTTON

The object type Radiobutton for the Verify keyword is used to verify whether a radio button is selected or not.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Radio button
- Attribute value – Attribute value of the Radio button (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	RADIOBUTTON	Name of the Radio button	Attribute value of the Radio button
Example	VERIFY	RADIOBUTTON	Always	always

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the radio button	Enter true or false			
Example	Always	True			

Reference Image

None.

A.2.58.8 SPREADCELL

The object type SpreadCell for the Verify keyword is used to verify a cell value in an EBS Forms spread table.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Spread table column name
- Attribute value – Spread table Column number

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	SPREADCELL	Name of the column in the spread table	Column number in the spread table
Example	VERIFY	SPREADCELL	Resource	2

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the column in the spread table	Value to verify in the spreadcell.			
Example	Resource	Karmer, Ralph			

Reference Image

None.

A.2.58.9 STATUSBAR

The object type Statusbar for the Verify keyword is used to verify the status bar message in an EBS forms application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Statusbar
- Attribute value – None

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	STATUSBAR	Statusbar	None
Example	VERIFY	STATUSBAR	Statusbar	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Statusbar				
Example	Statusbar	No changes to save.			

Reference Image

None.

A.2.58.10 TEXT

The object type Text for the Verify keyword is used to verify text in an EBS Web application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the text
- Attribute value – None

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	TEXT	Name of the text	None
Example	VERIFY	TEXT	Confirmation	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the text				
Example	Confirmation	Your changes have been saved.			

Reference Image

None.

A.2.58.11 TEXTAREA

The object type Textarea for the Verify keyword is used to verify text in a Textarea.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the TextArea
- Attribute value – Attribute value of the TextArea (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example

	Keyword	Object	Display Name	Attribute Value
Usage	VERIFY	TEXTAREA	Name of the Text area	Attribute value of the Text area
Example	VERIFY	TEXTAREA	Description	description

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Text area	Value in the Text area to verify			
Example	Description	Item 1 Description			

Reference Image

None.

A.2.59 WAIT

This keyword is used to wait for the specific object to load in the application for a specific amount of time. This keyword is used with these objects:

- Button
- Edit
- Image
- Link
- List
- Listbox
- Normal
- Textarea
- Window

A.2.59.1 BUTTON

The object type Button for the Wait keyword is used to wait for a button to load.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Button
- Attribute value – Attribute value of the Button (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WAIT	BUTTON	Name of the Button	Attribute value of the Button
Example	WAIT	BUTTON	Go	CreateGoButton

Test Data

None.

Reference Image

None.

A.2.59.2 EDIT

The object type Edit for the Wait keyword is used to wait for an Edit field to load.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Edit field
- Attribute value – Attribute value of the Edit field (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WAIT	EDIT	Name of the Edit field	Attribute value of the Edit field
Example	WAIT	EDIT	Search	SearchTextInput

Test Data

None.

Reference Image

None.

A.2.59.3 IMAGE

The object type Image for the Wait keyword is used to wait for an image to load.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Image
- Attribute value – Attribute value of the Image (properties from the xpath like name, alt, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WAIT	IMAGE	Name of the Image	Attribute value of the Image
Example	WAIT	IMAGE	Add Another Row	Add Another Row

Test Data

None.

Reference Image

None.

A.2.59.4 LINK

The object type Link for the Wait keyword is used to wait for a link to load in an EBS Web application.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Link
- Attribute value – Attribute value of the Link (properties from the xpath like name, text, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WAIT	LINK	Name of the Link	Attribute value of the Link
Example	WAIT	LINK	Show Details	Show Details

Test Data

None.

Reference Image

None.

A.2.59.5 LIST

The object type List for the Wait keyword is used to wait for a List to load.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the List
- Attribute value – Attribute value of the List (properties from the xpath like name, text, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WAIT	LIST	Name of the List	Attribute value of the List
Example	WAIT	LIST	Destination Type	PO_ DISTRIBUTIONS_ DESTINATION_ TYPE_0

Test Data
None.

Reference Image
None.

A.2.59.6 LISTBOX
The object type Listbox for the Wait keyword is used to wait for a Listbox to load.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Listbox
- Attribute value – Attribute value of the Listbox (properties from the xpath like name, id, etc.)

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WAIT	LISTBOX	Name of the Listbox	Attribute value of the Listbox
Example	WAIT	LISTBOX	Actions	ActionListTop

Test Data
None.

Reference Image
None.

A.2.59.7 NORMAL
The object type Normal for the Wait keyword is used to wait for specific amount of time.

Technical Details
Provide the following values for this keyword-object combination in the component:

- Display name – None
- Attribute value – Wait time

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WAIT	NORMAL	None	Wait time
Example	WAIT	NORMAL	None	5

Test Data
None.

Reference Image

None.

A.2.59.8 TEXTAREA

The object type Textarea for the Wait keyword is used to wait for a Textarea to load.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the TextArea
- Attribute value – Attribute value of the TextArea (properties from the xpath like name, id, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WAIT	TEXTAREA	Name of the Textarea	Attribute value of the Textarea
Example	WAIT	TEXTAREA	Description	description

Test Data

None.

Reference Image

None.

A.2.59.9 WINDOW

The object type Window for the Wait keyword is used to wait for the Web window to load.

Technical Details

Provide the following values for this keyword-object combination in the component:

- Display name – Name of the Window
- Attribute value – Attribute value of the Window (properties from the xpath like title, index, etc.)

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WAIT	WINDOW	Name of the Window	Attribute value of the Window
Example	WAIT	WINDOW	Orders	Orders

Test Data

None.

Reference Image

None.

A.3 PLSQL/Open Interface Keywords and Objects

The following sections list the Keywords and valid objects available to use to specify component code for PLSQL/Open Interface applications.

A.3.1 SETAPPTYPE

This keyword is used to set the Application type.

This keyword is used with these objects:

- PLSQL_OI
- WS

A.3.1.1 PLSQL_OI

The object type PLSQL_OI for the Setapptype keyword is used to set the application type as PLSQL or Open interface.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETAPPTYPE	PLSQL_OI	None	None
Example	SETAPPTYPE	PLSQL_OI	None	None

Test Data

None.

Reference Image

None.

A.3.1.2 WS

The object type WS for Setapptype keyword is used to set the application type as Webdservice.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETAPPTYPE	WS	None	None
Example	SETAPPTYPE	WS	None	None

Test Data

None.

Reference Image

None.

A.3.2 FUNCTIONCALL

This keyword is used to execute a function specified in the object.

This keyword is used with these objects:

- gENAPILIB
- gENWSLIB

A.3.2.1 gENAPILIB

The object type gENAPILIB for the Functioncall keyword is used to execute a function in the gENAPILIB library.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Purpose of the function
- Attribute value – Attribute value for the function call to be made. Providing attribute value is depends on the function called in the 'Function Name' field.
- Function Name – Name of the Function

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Function Name
Usage	FUNCTIONCALL	gENAPILIB	Purpose of the function	Attribute value for function call	Name of the GENAPILIB function
Example	FUNCTIONCALL	gENAPILIB	Conditional Commit	Conditional Commit	conditionalCommit

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Purpose of the function	expectedValue	actualValue		
Example	Conditional Commit	S	L_X_ RETURN_ STATUS1		

Reference Image

None.

A.3.2.2 gENWSLIB

The object type gENWSLIB for the Functioncall keyword is used to execute a function in the gENWSLIB llibrary.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Purpose of the function
- Attribute value – Attribute value for the function call to be made. Providing attribute value is depends on the function called in the 'Function Name' field.
- Function Name – Name of the Function

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Function Name
Usage	FUNCTIONCALL	gENWSLIB	Purpose of the function	Attribute value for function callof the field	Name of the GENWSLIB function
Example	FUNCTIONCALL	gENWSLIB	GetNodeValue	a/b/c	getNodeElem emtData

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Purpose of the functionName of the field	DataElement Index			
Example	GetNodeValue	1			

Reference Image

None.

A.3.3 SELECT

This keyword is used to specify the Database Table Name and Column Name in the testdata.

This keyword is used with these objects:

- COLUMN
- TABLE

A.3.3.1 COLUMN

The object type Column for the Select keyword is used specify the table column.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Specify appropriate information like Column Details.
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SELECT	COLUMN	Column Details	None
Example	SELECT	COLUMN	Column Details	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Column Details	Database Column Name			
Example	Column Details	ITEM_ID			

Reference Image

None.

A.3.3.2 TABLE

The object type Table for the Select keyword is used specify the table name.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Specify appropriate information like Name of the Table.
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SELECT	TABLE	Name of the Table	None
Example	SELECT	TABLE	Name of the Table	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Name of the Table	Database Table Name			
Example	Name of the Table	EMPLOYEES			

Reference Image

None.

A.3.4 BEGINCALL

This keyword is used to start the Call block.

- Begincall is used in to call a PLSQL procedure

Technical Details

Provide the following values for this keyword–object combination in the component:

- Begincall is used to call a PLSQL procedure
  - Provide attribute value as "Name of the package. Name of the Procedure for procedure.
  - Provide attribute value as "Name of the package. Name of the Procedure for procedure,@retruntype = function returntype" for functions.

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	BEGINCALL	None	Optional	Name of the Procedure	
Example 1	BEGINCALL	None	Insert Activity	GMD_ ACTIVITIES _ PUB.INSERT _ACTIVITY	
Example 2	BEGINCALL	None	Capital Budget Interface	FND_ REQUEST.S UBMIT_ REQUEST,@ returntype= NUMBER	L_FASCB_ REQ_ID

Test Data

None.

Reference Image

None.

A.3.5 ENDCALL

This keyword is used to end the call block.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDCALL	None	None	None
Example	ENDCALL	None	None	None

Test Data

None.

Reference Image

None.

A.3.6 STARTRECORDTYPE

This keyword is used to start the Record type block. A Record type block is used to define Database Record type.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Record Name
- Attribute value — Database Package Name.Database Record Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	STARTRECORDTYPE	None	Database Record Name	Database Package Name.Database Record Name	>Output variable

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Example	STARTRECORDTYPE	None	Insertion Record	@udt=GMD_ACTIVITIES_PUB.GMD_ACTIVITIES_REC_TYPE	L_ITEM_REC

Test Data

The following table provides details for entering Test Data.

Specify whether record data is going as input data to the procedure or record is returning from the procedure by specifying either as INPUT or OUPUT.

- INPUT – Record data going as input to the procedure
- OUTPUT – Record returned as output from the procedure

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Record Name	Specify INPUT or OUTPUT			
Example	Insertion Record	INPUT			

Reference Image

None.

A.3.7 ENDRECORDTYPE

This keyword is used to end the record type block.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDRECORDTYPE	None	None	None
Example	ENDRECORDTYPE	None	None	None

Test Data

None.

Reference Image

None.

A.3.8 STARTTABLETYPE

This keyword is used to start the database Table type block. A Table type block is used to define Database Table type.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – database Table Name
- Attribute value – database Package Name.Database Table Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	STARTTABLETYPE	None	Database Table Name	Database Package Name.Database Table Name	Output variable
Example	STARTTABLETYPE	None	GMD Activities Table	@udt=GMD_ACTIVITIES_PUB.GMD_ACTIVITIES_TBL_TYPE	L_ACTIVITY_TBL

Test Data

The following table provides details for entering Test Data.

Specify whether database table data is going as input data to the procedure or database table is returning from the procedure by specifying either as INPUT or OUPUT.

- INPUT – database table data going as input to the procedure
- OUTPUT – database table returned as output from the procedure

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Table Name	Specify INPUT or OUTPUT			
Example	GMD Activities Table	INPUT			

Reference Image

None.

A.3.9 ENDTABLETYPE

This keyword is used to end the Table type block.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDTABLETYPE	None	None	None
Example	ENDTABLETYPE	None	None	None

Test Data

None.

Reference Image

None.

A.3.10 STARTVARRAYTYPE

This keyword is used to start the VArray type block. A VArray type block is used to perform actions on VArray type data.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database VArrayName
- Attribute value – Database Package Name.Database VArray Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	STARTVARRAYTYPE	None	Database VArrayName	Procedure.Name of the VARRAY	Output variable
Example	STARTVARRAYTYPE	None	GMD Activities	@udt=GMD_ACTIVITIES_PUB.GMD_ACTIVITIES_VARRAY_TYPE	L_Activitties_VArray

Test Data

The following table provides details for entering Test Data.

Specify whether database VArray data is going as input data to the procedure / database VArray type is returning from the procedure by specifying either as INPUT or OUPUT.

- INPUT – database table data going as input to the procedure
- OUTPUT – database table returned as output from the procedure

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database VArray Name	Specify INPUT or OUTPUT			
Example	GMD Activities	INPUT			

Reference Image

None.

A.3.11 ENDVARRAYTYPE

This keyword is used to end the VArray block.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDAVRRAYTYPE	None	None	None
Example	ENDVARRAYTYPE	None	None	None

Test Data

None.

Reference Image

None.

A.3.12 STARTOBJECTTYPE

This keyword is used to start the Object type block.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Object Name
- Attribute value – Database Object Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	STARTOBJECTTYPE	None	Database Object Name	Database Object Name	Output variable

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Example	STARTOBJECTTYPE	None	Wf Event	@udt=WF_ EVENT_T	L_EVENT

Test Data

The following table provides details for entering Test Data.

Specify whether object data is going as input data to the procedure or object is returning from the procedure by specifying either as INPUT or OUPUT.

- INPUT – Object data going as input to the procedure
- OUTPUT – Object returned as output from the procedure

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Object Name	Specify INPUT or OUTPUT			
Example	Wf Event	INPUT			

Reference Image

None.

A.3.13 ENDOBJECTTYPE

This keyword is used to end the Object type block.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDOBJECTTYPE	None	None	None
Example	ENDOBJECTTYPE	None	None	None

Test Data

None.

Reference Image

None.

A.3.14 SETVARIN

This keyword is used to specify the parameter as a input parameter to the database procedure or function.

This keyword is used with these objects:

- BOOLEAN
- CHAR
- DATE
- INT
- INTEGER
- LONG
- NUMBER
- OBJECT_TYPE
- RECORD_TYPE
- TABLE_TYPE
- VARCHAR
- VARCHAR2
- VARRAY_TYPE

A.3.14.1 BOOLEAN

The object type Boolean for the Setvarin keyword is used to provide True/False.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	BOOLEAN	Procedure Parameter Name	Procedure Parameter Name
Example	SETVARIN	BOOLEAN	Validate	P_VALIDATE

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name	Specify True/False			
Example	Validate	TRUE			

Reference Image

None

A.3.14.2 CHAR

The object type Char for the Setvarin keyword is used to provide character data.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	CHAR	Procedure Parameter Name	Procedure Parameter Name
Example	SETVARIN	CHAR	Enable Flag	P_RI_BOM_ ENABLED_FLAG

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name	Specify character value			
Example	Enable Flag	T			

Reference Image

None

A.3.14.3 DATE

The object type Date for the Setvarin keyword is used to provide a Date value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	DATE	Procedure Parameter Name	Procedure Parameter Name
Example	SETVARIN	DATE	Start Date	START_DATE

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name	Specify Date	Specify Date Format		
Example	Start Date	#SYSDATE(1)	DD–MON–YY YY		

Reference Image

None

A.3.14.4 INT

The object type Int for the Setvarin keyword is used to provide an Int value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	INT	Procedure Parameter Name	Procedure Parameter Name
Example	SETVARIN	INT	Line Id	LINE_ID

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Line Id	12			

Reference Image

None

A.3.14.5 INTEGER

The object type Integer for the Setvarin keyword is used to provide an Integer value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	INTEGER	Procedure Parameter Name	Procedure Parameter Name
Example	SETVARIN	INTEGER	Applicationidin	APPLICATIONIDIN

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Applicationidin	12345			

Reference Image

None

A.3.14.6 LONG

The object type Long for the Setvarin keyword is used to provide a Long Integer value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	LONG	Procedure Parameter Name	Procedure Parameter Name
Example	SETVARIN	LONG	Comments	P_COMMENTS

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Comments	1234567			

Reference Image

None

A.3.14.7 NUMBER

The object type Number for the Setvarin keyword is used to provide a Numeric value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	NUMBER	Procedure Parameter Name	Procedure Parameter Name
Example	SETVARIN	NUMBER	Line Number	LINE_NUMBER

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Line Number	245			

Reference Image

None.

A.3.14.8 OBJECT_TYPE

The object type "OBJECT_TYPE" for the Setvarin keyword is used to provide an Object Type defined in the database.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Object Name
- Attribute value – Database Object Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	OBJECT_TYPE	Database Object Name	Database Object Name
Example	SETVARIN	OBJECT_TYPE	Event	P_EVENT,@udt='WF_EVENT_T'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Object Name	Specify Object type associated to it			
Example	Event	WF_EVENT_T			

Reference Image

None

A.3.14.9 RECORD_TYPE

The object type RECORD_TYPE for the Setvarin keyword is used to provide a Record Type defined in the database.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Record Name
- Attribute value – Database Record Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	RECORD_TYPE	Database Record Name	Database Record Name
Example	SETVARIN	RECORD_TYPE	Approverin	APPROVERIN,@udt ='AME_ UTIL.APPROVERR ECORD2'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Record Name	Specify Record type associated to it			
Example	Approverin	APPROVERR ECORD2			

Reference Image

None

A.3.14.10 TABLE_TYPE

The object type Table Type for the Setvarin keyword is used to provide a Table Type defined in database.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Table Name
- Attribute value – Database Table Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	TABLE_TYPE	Database Table Name	Database Table Name
Example	SETVARIN	TABLE_TYPE	Insertion Table	P_INSERTION @udt='AME_ UTIL.INSERTIONST ABLE2'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Table Name	Specify Table type associated to it			
Example	Insertion Table	INSERTIONST ABLE2			

Reference Image

None

A.3.14.11 VARCHAR

The object type Varchar for the Setvarin keyword is used to provide character data of indeterminate length.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	VARCHAR	Procedure Parameter Name	Procedure Parameter Name
Example	SETVARIN	VARCHAR	Transactiontypein	TRANSACTIONTY PEIN

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Transactiontypein	CREATE			

Reference Image

None

A.3.14.12 VARCHAR2

The object type Varchar2 for the Setvarin keyword is used to provide character data of specific length.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	VARCHAR2	Procedure Parameter Name	Procedure Parameter Name
Example	SETVARIN	VARCHAR2	Transactionidin	TRANSACTIONIDIN,@size='1000'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Transactionidin	AS55234			

Reference Image

None

A.3.14.13 VARRAY_TYPE

The object type VARRAY_TYPE for the Setvarin keyword is used to provide a VArray Type defined in the database.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Varray Name
- Attribute value – Database Varray Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVARIN	VARRAY_TYPE	Database Varray Name	Database Varray Name
Example	SETVARIN	VARRAY_TYPE	Tagid	P_ TAGID,@udt='WMS _EPC_TAGID_ TYPE'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Varray Name	Specify Varray type associated to it			
Example	Tagid	WMS_EPC_ TAGID_TYPE			

Reference Image

None

A.3.15 SETVAROUT

This keyword is used to set the variable type.specify the parameter is a output parameter of the database procedure or function.

This keyword is used with these objects:

- BOOLEAN
- CHAR
- DATE
- INT
- INTEGER
- LONG
- NUMBER
- OBJECT_TYPE
- RECORD_TYPE
- TABLE_TYPE
- VARCHAR

- VARCHAR2
- VARRAY_TYPE

A.3.15.1 BOOLEAN

The object type Boolean for the Setvarout keyword is used to get a Boolean output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	BOOLEAN	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVAROUT	BOOLEAN	Validate	P_VALIDATE	X_VALIDATE

Test Data

None

Reference Image

None

A.3.15.2 CHAR

The object type Char for the Setvarout keyword is used to get a character value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	CHAR	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVAROUT	CHAR	Enable Flag	P_RI_BOM_ ENABLED_ FLAG	X_ENABLE_ FLAG

Test Data
None

Reference Image
None

A.3.15.3 DATE
The object type Date for the Setvarout keyword is used to get an output Date value.

Technical Details
Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	DATE	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVAROUT	DATE	Start Date	START_ DATE	X_ STARTDATE

Test Data
None

Reference Image
None

A.3.15.4 INT
The object type Int for the Setvarout keyword is used to get an Int value.

Technical Details
Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	INT	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVAROUT	INT	Line Id	LINE_ID	X_LINEID

Test Data
None

Reference Image
None

A.3.15.5 INTEGER
The object type Integer for the Setvarout keyword is used to get an Integer value.

Technical Details
Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	INTEGER	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVAROUT	INTEGER	Applicationidin	APPLICATIONIDIN	X_APPLICATIONIDIN

Test Data
None

Reference Image
None

A.3.15.6 LONG
The object type Long for the Setvarout keyword is used to get a Long Integer value.

Technical Details
Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	LONG	Procedure Parameter Name	Procedure Parameter Name	Output Variable

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Example	SETVAROUT	LONG	Comments	P_ COMMENTS	X_ COMMENTS

Test Data

None.

Reference Image

None

A.3.15.7 NUMBER

The object type Number for the Setvarout keyword is used to get a Numeric value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	NUMBER	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVAROUT	NUMBER	Line Number	LINE_ NUMBER	L_LINE_ NUM

Test Data

None

Reference Image

None

A.3.15.8 OBJECT_TYPE

The object type "Object_Type" for the Setvarout keyword is used to get an Object Type as output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Object Name
- Attribute value – Database Object Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	OBJECT_TYPE	Database Object Name	Database Object Name	Output Variable
Example	SETVAROUT	OBJECT_TYPE	Event	P_ EVENT,@ud t='WF_ EVENT_T'	X_EVENT

Test Data

None

Reference Image

None

A.3.15.9 RECORD_TYPE

The object type Record Type for the Setvarout keyword is used to get a Record Type as output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Record Name
- Attribute value – Database Record Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	RECORD_ TYPE	Database Record Name	Database Record Name	Output Variable
Example	SETVAROUT	RECORD_ TYPE	Approverout	APPROVER OUT,@udt=' AME_ UTIL.APPR OVERRECO RD2'	X_ APPROVERO UT

Test Data

None

Reference Image

None

A.3.15.10 TABLE_TYPE

The object type Table Type for the Setvarout keyword is used to get a Table Type as output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Table Name
- Attribute value – Database Table Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	TABLE_TYPE	DatabaseTable Name	Database Table Name	Output Variable
Example	SETVAROUT	TABLE_TYPE	Insertion Table	P_ INSERTION @udt='AME _ UTIL.INSER TIONSTABL E2'	X_ INSERTION

Test Data

None

Reference Image

None

A.3.15.11 VARCHAR

The object type Varchar forthe Setvarout keyword is used to get character data of indeterminate length as output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	VARCHAR	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVAROUT	VARCHAR	Transactionty peout	TRANSACT IONTYPEO UT	X_ TRANSACTI ONTYPEOUT

Test Data

None

Reference Image

None

A.3.15.12 VARCHAR2

The object type Varchar2 for the Setvarout keyword is used to get character data of specific length as output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	VARCHAR2	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVAROUT	VARCHAR2	Transactionid out	TRANSACTIONIDOUT, @size='1000'	X_ TRANSACTIONIDOUT

Test Data

None

Reference Image

None

A.3.15.13 VARRAY_TYPE

The object type VArray Type for the Setvarout keyword is used to get VArray Type as output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Varray Name
- Attribute value – Database Varray Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	VARRAY_ TYPE	Database Varray Name	Database Varray Name	Output Variable

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Example	SETVAROUT	VARRAY_ TYPE	Tagid	P_ TAGID,@udt ='WMS_ EPC_ TAGID_ TYPE'	L_TAGID

Test Data

None

Reference Image

None

A.3.16 SETVARINOUT

This keyword is used to specify the parameter is used as both an input and output parameter to the database procedure or function.

This keyword is used with these objects:

- BOOLEAN
- CHAR
- DATE
- INT
- INTEGER
- LONG
- NUMBER
- OBJECT_TYPE
- RECORD_TYPE
- TABLE_TYPE
- VARCHAR
- VARCHAR2
- VARRAY_TYPE

A.3.16.1 BOOLEAN

The object type Boolean for the Setvarinout keyword is used as both providing True/False and get output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	BOOLEAN	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVARINOUT	BOOLEAN	Validate	P_ VALIDATE	X_VALIDATE

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name	Specify True/False			
Example	Validate	TRUE			

Reference Image

None

A.3.16.2 CHAR

The object type Char for the Setvarinout keyword is used to provide character and get character output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	CHAR	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVARINOUT	CHAR	Enable Flag	P_RI_BOM_ ENABLED_ FLAG	X_ENAB LED_FLAG

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name	Specify True/False			

	Display Name	Value 1	Value 2	Value 3	Value 4
Example	Enable Flag	T			

Reference Image

None

A.3.16.3 DATE

The object type Date for the Setvarinout keyword is used as both providing the Date value and get output Date value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	DATE	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVARINOUT	DATE	Start Date	START_ DATE	X_ STARTDATE

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Example	Start Date	#SYSDATE(1)	DD–MON–YY YY		

Reference Image

None

A.3.16.4 INT

The object type Int for the Setvarinout keyword is used to provide and get an Int value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	INT	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVARINOUT	INT	Line Id	LINE_ID	X_LINEID

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Line Id	12			

Reference Image

None

A.3.16.5 INTEGER

The object type Integer for the Setvarinout keyword is used as both to provide and get an Integer value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	INTEGER	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVARINOUT	INTEGER	Applicationidin	APPLICATIONIDIN	X_APPLICATIONIDIN

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Applicationidin	12345			

Reference Image

None

A.3.16.6 LONG

The object type Long for the Setvarinout keyword is used to provide Long Integer value and get output value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	LONG	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVARINOUT	LONG	Comments	P_ COMMENTS	X_ COMMENTS

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Comments	1234567			

Reference Image

None

A.3.16.7 NUMBER

The object type Number for the Setvarinout keyword is used as both to provide data and get Numeric data.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	NUMBER	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVARINOUT	NUMBER	Line Number	LINE_ NUMBER	X_LINE_ NUM

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Line Number	245			

Reference Image

None

A.3.16.8 OBJECT_TYPE

The object type "Object Type" for the Setvarinout keyword is used as both to provide and get Object Type.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Object Name
- Attribute value – Database Object Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	OBJECT_TYPE	Database Object Name	Database Object Name	Output Variable
Example	SETVARINOUT	OBJECT_TYPE	Event	P_ EVENT,@ud t='WF_ EVENT_T'	X_EVENT

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Object Name	Specify Object type associated to it			
Example	Event	WF_EVENT_T			

Reference Image

None

A.3.16.9 RECORD_TYPE

The object type "Record Type" for the Setvarinout keyword is used as both to provide and get Record Type.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Record Name
- Attribute value – Database Record Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	RECORD_ TYPE	Database Record Name	Database Record Name	Output Variable
Example	SETVARINOUT	RECORD_ TYPE	Approverout	APPROVER OUT,@udt=' AME_ UTIL.APPR OVERRECO RD2'	X_ APPROVERO UT

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Record Name	Specify Record type associated to it			
Example	Approverin	APPROVERR ECORD2			

Reference Image

None

A.3.16.10 TABLE_TYPE

The object type "Table Type" forthe Setvarinout keyword is used as both to provide and get Table Type.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Table Name
- Attribute value – Database Table Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	TABLE_TYPE	DatabaseTable Name	Database Table Name	Output Variable
Example	SETVAROUT	TABLE_TYPE	Insertion Table	P_ INSERTION @udt='AME _ UTIL.INSER TIONSTABL E2'	X_ INSERTION

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Table Name	Specify Table type associated to it			
Example	Insertion Table	INSERTIONST ABLE2			

Reference Image

None

A.3.16.11 VARCHAR

The object type Varchar for the Setvarinout keyword is used as both to provide and get character data of an indeterminate length.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVAROUT	VARCHAR	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVAROUT	VARCHAR	Transactionty peinout	TRANSACT IONTYPEIN OUT	X_ TRANSACTI ONTYPEINO UT

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Transactiontypei nout	CREATE			

Reference Image

None

A.3.16.12 VARCHAR2

The object type Varchar2 for the Setvarinout keyword is used as both to provide and get character data of a specific length.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	VARCHAR2	Procedure Parameter Name	Procedure Parameter Name	Output Variable
Example	SETVARINOUT	VARCHAR2	Transactionidi nout	TRANSACT IONIDINOU T,@size='100 0'	X_ TRANSACTI ONIDINOUT

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Transactionidin	AS55234			

Reference Image

None

A.3.16.13 VARRAY_TYPE

The object type VArray Type for the Setvarinout keyword is used as both to provide and get VArray Type.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Varray Name
- Attribute value – Database Varray Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value	Output Parameter
Usage	SETVARINOUT	VARRAY_ TYPE	Database Varray Name	Database Varray Name	Output Variable
Example	SETVARINOUT	VARRAY_ TYPE	Tagid	P_ TAGID,@udt ='WMS_ EPC_ TAGID_ TYPE'	L_TAGID

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Varray Name	Specify Varray type associated to it			
Example	Tagid	WMS_EPC_ TAGID_TYPE			

Reference Image

None

A.3.17 SETVAR

This keyword is used to specify the parameters of the User Defined Data type like Record, Table etc.

This keyword is used with these objects:

- BOOLEAN
- CHAR
- DATE
- INT
- INTEGER
- LONG
- NUMBER
- OBJECT_TYPE
- RECORD_TYPE
- VARCHAR
- VARCHAR2

A.3.17.1 BOOLEAN

The object type Boolean for the Setvar keyword is used to provide True/False.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	BOOLEAN	Procedure Parameter Name	Procedure Parameter Name
Example	SETVAR	BOOLEAN	Validate	P_VALIDATE

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name	Specify True/False			
Example	Validate	TRUE			

Reference Image

None

A.3.17.2 CHAR

The object type Char for the Setvar keyword is used to provide a Character value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	CHAR	Procedure Parameter Name	Procedure Parameter Name
Example	SETVAR	CHAR	Enable Flag	P_RC_BOM_ ENABLED_FLAG

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name	Specify Character value			
Example	Enable Flag	T			

Reference Image

None

A.3.17.3 DATE

The object type Date for the Setvar keyword is used to provide a Date value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	DATE	Procedure Parameter Name	Procedure Parameter Name
Example	SETVAR	DATE	Start Date	START_DATE

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Example	Start Date	#SYSDATE(1)	DD–MON–YY YY		

Reference Image

None

A.3.17.4 INT

The object type Int for the Setvar keyword is used to provide an Int value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	INT	Procedure Parameter Name	Procedure Parameter Name
Example	SETVAR	INT	Line Id	LINE_ID

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Line Id	12			

Reference Image

None

A.3.17.5 INTEGER

The object type Integer for the Setvar keyword is used to provide an Integer value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	INTEGER	Procedure Parameter Name	Procedure Parameter Name
Example	SETVAR	INTEGER	Applicationidin	APPLICATIONIDIN

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Applicationidin	12345			

Reference Image

None

A.3.17.6 LONG

The object type Long for the Setvar keyword is used to provide a Long Integer value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	LONG	Procedure Parameter Name	Procedure Parameter Name
Example	SETVAR	LONG	Comments	P_COMMENTS

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Comments	1234567			

Reference Image

None

A.3.17.7 NUMBER

The object type Number for the Setvar keyword is used to provide a Numeric value.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	NUMBER	Procedure Parameter Name	Procedure Parameter Name
Example	SETVAR	NUMBER	Line Number	LINE_NUMBER

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Line Number	245			

Reference Image

None

A.3.17.8 OBJECT_TYPE

The object type "OBJECT_TYPE" for the Setvar keyword is used to provide an Object Type defined in a database.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Object Name
- Attribute value – Database Object Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	OBJECT_TYPE	Database Object Name	Database Object Name
Example	SETVAR	OBJECT_TYPE	Event	P_EVENT,@udt='WF_EVENT_T'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Object Name	Specify Object type associated to it			
Example	Event	WF_EVENT_T			

Reference Image

None

A.3.17.9 RECORD_TYPE

The object type Record_Type for the Setvar keyword is used to provide a Record Type defined in a database.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Record Name
- Attribute value – Database Record Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	RECORD_TYPE	Database Record Name	Database Record Name
Example	SETVAR	RECORD_TYPE	Approverin	APPROVERIN,@udt ='AME_ UTIL.APPROVERR ECORD2'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Record Name	Specify Record type associated to it			
Example	Approverin	APPROVERR ECORD2			

Reference Image

None

A.3.17.10 VARCHAR

The object type Varchar for the Setvar keyword is used to provide Character data of indeterminate length.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	VARCHAR	Procedure Parameter Name	Procedure Parameter Name
Example	SETVAR	VARCHAR	Transactiontypein	TRANSACTIONTYPEIN

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Transactiontypein	CREATE			

Reference Image

None

A.3.17.11 VARCHAR2

The object type Varchar2 for the Setvar keyword is used to provide Character data of a specific length.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Procedure Parameter Name
- Attribute value – Procedure Parameter Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SETVAR	VARCHAR2	Procedure Parameter Name	Procedure Parameter Name
Example	SETVAR	VARCHAR2	Transactionidin	TRANSACTIONIDIN,@size='1000'

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Procedure Parameter Name				
Example	Transactionidin	AS55234			

Reference Image

None

A.3.18 STARTROWITERATOR

This keyword is used to start the Row Iterate block. An Row Iterate block is used to repeat rows multiple times during flow Test Data.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Number of Rows
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTROWITERATOR	None	Number of Rows	None
Example	STARTROWITERATOR	None	Number of Rows	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Number of Rows				
Example	Number of Rows	2			

Reference Image

None

A.3.19 ENDROWITERATOR

This keyword is used to end the Row Iterator block.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDROWITERATOR	None	None	None
Example	ENDROWITERATOR	None	None	None

Test Data

None

Reference Image

None

A.3.20 STARTBLOCK

This keyword is used to define the actual procedure call. This is used along with the BEGINCALL keyword.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTBLOCK	None	None	None
Example	STARTBLOCK	None	None	None

Test Data

None

Reference Image

None

A.3.21 ENDBLOCK

This keyword is used to end the Start block.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDBLOCK	None	None	None
Example	ENDBLOCK	None	None	None

Test Data

None

Reference Image

None

A.3.22 SET

This keyword is used to define the database table column. This key word is used along with INSERT or SELECT keywords.

This keyword is used with these objects:

- COLUMN
- CONDITION

A.3.22.1 COLUMN

The Object Type Column for the Set keyword is used to set the value for a Database Table Column.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Column Name
- Attribute value – Database Column Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SET	COLUMN	Database Column Name	Database Column Name
Example	SET	COLUMN	Organization Id	ORGANIZATION_ ID

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Database Column Name				
Example	Organization Id	204			

Reference Image

None

A.3.22.2 CONDITION

The Object Type Condition for the Set keyword is used to specify the condition for a SQL query while retrieving the value from a Database.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Condition Details
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	SET	CONDITION	Condition Details	None
Example	SET	CONDITION	Condition Details	None

Test Data

The following table provides details for entering Test Data.

	Display Name	Value 1	Value 2	Value 3	Value 4
Usage	Condition Details	Database Column Name	Condition	Column value	Conditional Operator
Example	Condition Details	EMP_ID	EQUAL	205	AND

Reference Image

None

A.3.23 INSERT

This Keyword is used to insert Database Row data into a Database Table.

This keyword is used with these objects:

- TABLE

A.3.23.1 TABLE

The object type Table for Insert keyword is used to insert the row data into a Database table.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Database Table Name
- Attribute value – Database Table Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	INSERT	TABLE	Database Table Name	Database Table Name
Example	INSERT	TABLE	PO Lines Interface Table	PO_LINES_ INTERFACE

Test Data

None

Reference Image

None

A.3.24 STARTQUERY

This keyword is used to start the Query block. A Query block is used to perform actions on database tables such as retrieving a value from a database table or inserting a record into a database table.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	STARTQUERY	None	None	None
Example	STARTQUERY	None	None	None

Test Data

None

Reference Image

None

A.3.25 ENDQUERY

This keyword is used to end the Query block.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	ENDQUERY	None	None	None
Example	ENDQUERY	None	None	None

Test Data

None

Reference Image

None

A.3.26 WS-STARTSTRUCTURE

This keyword is used to start Webservice structure. All the web service details should be defined within this keyword.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WS-STARTSTRUCTURE	None	None	None
>**Example**	WS-STARTSTRUCTURE	None	None	None

Test Data

None

Reference Image

None

A.3.27 WS-ENDSTRUCTURE

This keyword is used to end the Webservice structure.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WS-ENDSTRUCTURE	None	None	None
Example	WS-ENDSTRUCTURE	None	None	None

Test Data
None

Reference Image
None

A.3.28 WS-SETWEBSERVICENAME
This keyword is used to specify the WSDL details for the specific payload.

Technical Details
Provide the following values for this keyword–object combination in the component:

- Display name – WSDL Name
- Attribute value – WSDL URL

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WS-SETWEBSERVICENAME	None	WSDL Name	WSDL URL
Example	sWS-SETWEBSERVICENAME	None	WSDL URL Static Part	https://{{HOST}}:{{PORT}}/webservices/AppsWSProvider/oracle/apps/rrs/site/service/SiteService?wsdl

Test Data
None

Reference Image
None

A.3.29 WS-NODENAME
This keyword is used to specify the Node Name and its path from the payload.

Technical Details
Provide the following values for this keyword–object combination in the component:

- Display name – Name
- Attribute value – Node Attribute Value

Usage in Defining Components
The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WS-NODENAME	None	NoneNode Name	Node Attribute Value
Example	WS-NODENAME	None	NoneWIP Mass Load	soapenv:Header/ser:ServiceBean_Header

Test Data

None

Reference Image

None

A.3.30 WS-NODENAMEWITHATTRIBUTE

This keyword is used to specify the Node Name with attributes of the node from the payload.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Node display name
- Attribute value – Node name and it's Attributes with Values

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WS-NODENAMEWITHATTRIBUTE	None	Node Name	Node Name and it's Attribute Values
Example	WS-NODENAMEWITHATTRIBUTE	None	Envelop	soapenv:Envelope,@xmlns:ser="http://xmlns.oracle.com/apps/fnd/ServiceBean",@xmlns:ser1="http://xmlns.oracle.com/apps/rrs/site/service",@xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"

Test Data

None

Reference Image

None

A.3.31 WS-PROCESSWSREQUEST

This keyword is used to start the payload structure.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – None

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WS-PROCESSWS REQUEST	None	None	None
Example	WS-PROCESSWS REQUEST	None	None	None

Test Data

None

Reference Image

None

A.3.32 WS-PARENT

This keyword is used to specify the Parent Node for a specific node from the payload.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – None
- Attribute value – Parent Node name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WS-PARENT	NODE	None	Node Attribute ValueParent Node name
Example	WS-PARENT	NODE	None	soapenv:Envelope

Test Data

None

Reference Image

None

A.3.33 WS-SETXMLELEMENT

This keyword is used to define the Node Element.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Element Name
- Attribute value – Element Name

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WS-SETXMLELEMENT	None	Element Name	Element Name
Example	WS-SETXMLELEMENT	None	responsibility_name	ser:RESPONSIBILITY_NAME

Test Data

None

Reference Image

None

A.3.34 WS-OPERATIONNAME

This keyword is used to specify the operation defined in the WSDL.

Technical Details

Provide the following values for this keyword–object combination in the component:

- Display name – Operation Name
- Attribute value – Operation Value

Usage in Defining Components

The following table lists usage with example.

	Keyword	Object	Display Name	Attribute Value
Usage	WS-OPERATIONNAME	None	Element Display NameOperation Name	Element Attribute ValueOperation Value
Example	WS-OPERATIONNAME	None	Operation Name	SiteService_CreateSite

Test Data

None

Reference Image

None

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

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W