# Oracle Application Testing Suite for Oracle Utilities

**Administration and User's Guide**

Release 4.2.0

**E58833-01**

January 2015

Oracle Application Testing Suite for Oracle Utilities Administration and User's Guide, Release 4.2.0

E58833-01

# Contents

# Preface

Welcome to the Oracle Application Testing Suite for Oracle Utilities (OATSOU) Administration and User's Guide. This guide explains how to use Oracle Application Testing Suite for Oracle Utilities to automate the business test flows for testing the Oracle Utilities' applications.

Oracle Application Testing Suite for Oracle Utilities is a licensed product and requires Oracle Functional Tester (OFT).

The Oracle Application Testing Suite for Oracle Utilities media pack comprises the following documentation and installation packages:

- **Documentation Package**
  - Oracle Application Testing Suite for Oracle Utilities v4.2.0 Release Notes
  - Oracle Application Testing Suite for Oracle Utilities v4.2.0 Administration and User's Guide
  - Oracle Application Testing Suite for Oracle Utilities v4.2.0 Reference Guide for Oracle Utilities Mobile Workforce Management v2.2.0.1 and Oracle Real-Time Scheduler v2.2.0.1
- **Installation Package**
  - Oracle Application Testing Suite for Oracle Utilities v4.2.0 Multiplatform

This preface contains the following:

- Audience
- Prerequisite Knowledge
- Related Documents
- Notational Conventions

## Audience

This guide is intended for Administrators, Automation Developers, and Test Engineers who will be automating the business test flows for testing the Oracle Utilities' applications.

## Prerequisite Knowledge

This guide does require an understanding of software testing concepts. The users must be familiar with Oracle Flow Builder.

> **Note**: *Oracle Flow Builder User's Guide* can be downloaded from Oracle Technology Network (http://www.oracle.com/technetwork/oem/downloads/index-084446.html).

The metadata driven automation development paradigm of Oracle Application Testing Suite for Oracle Utilities does not require any programming experience to develop scripts for testing. However, the advanced programming features available in Oracle OpenScript do require an experience with the Java programming language.

## Related Documents

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

- *Oracle Application Testing Suite for Oracle Utilities Release Notes*
- *Oracle Application Testing Suite for Oracle Utilities Reference Guide for Mobile Workforce Management and Oracle Real-Time Scheduler*

See also:

- Oracle Application Testing Suite Documentation Library
- Oracle Functional Testing OpenScript Documentation Library

## Notational Conventions

The following text conventions are used in this document:

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

# Chapter 1
# Getting Started

This chapter introduces the Oracle Application Testing Suite for Oracle Utilities application and provides an overview of the features available.

- Introduction

## Introduction

Oracle Application Testing Suite for Oracle Utilities (OATSOU) comprises test automation accelerators for automated testing of the Oracle Utilities applications. It is a framework based on Oracle Functional Tester (OFT) for creating the Web services automation scripts.

Oracle Application Testing Suite for Oracle Utilities enables users to create the automation scripts using keywords or metadata, and without using any programming language. This saves the test automation development effort and avoid programming the scripts using OpenScript Workbench.

**Note**: Oracle Functional Tester and OpenScript are part of Oracle Application Testing Suite. See the *Oracle Functional Tester User Guide* for more information.

The accelerators contain out-of-the-box delivered test components that can be used to build test flows for the Oracle Utilities applications. Users can extend the delivered components or create a new component to build their customized test flows. Utilities' application specific sample test flows are provided in the respective reference guides.

This introduction includes the following sections:

- Terminology
- Application Architecture
- Understanding the Roles
- Pointers for Getting Started

## Terminology

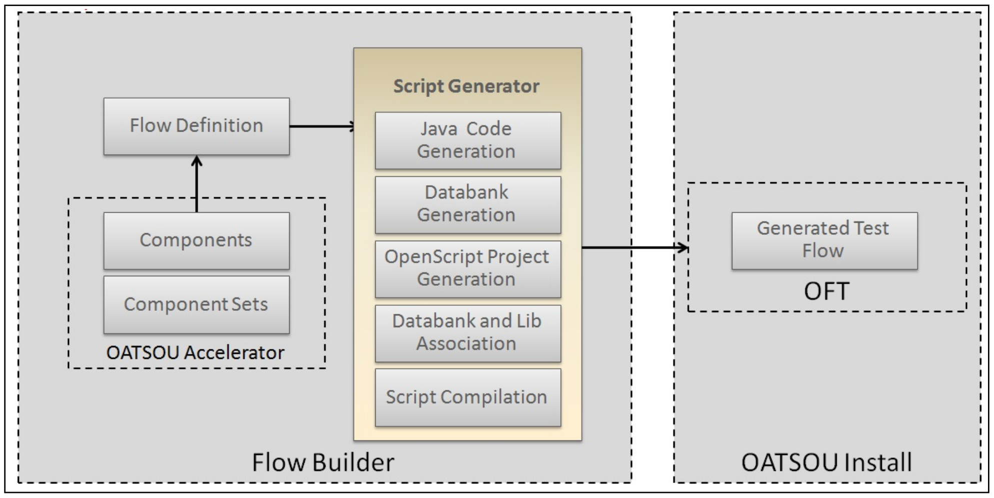
This section lists the different terms used in the document.

| Term | Description |
|---|---|
| Oracle Flow Builder (OFB) | Helps to build and maintain components and flows for automated testing. This tool is part of Oracle Application Testing Suite.<br><br>**Note**: See the *Oracle Flow Builder User's Guide* for more information. |

| Term | Description |
|---|---|
| Keyword | Pre-defined set of words used to define a specific step in a test case. |
| Component Line | Represents a step in a test case defined by a keyword, and associates values and parameters. |
| Component | Reusable automated test or part of a test.<br>A component is the building block of an automated test flow, comprising one or more component lines. |
| Component Set | Set of reusable group of components arranged in a pre-determined order.<br>A component set is used to perform a set of repeatable tasks. Component sets, along with components, are used to define a flow. |
| Flow | Automated test<br>A flow comprises one or more components and/or component sets that are called in a pre-determined sequence. |
| Databank | Container of test data used by an automated test flow.<br>The databank is defined using comma separated values (.csv) in a text file. |

## Application Architecture

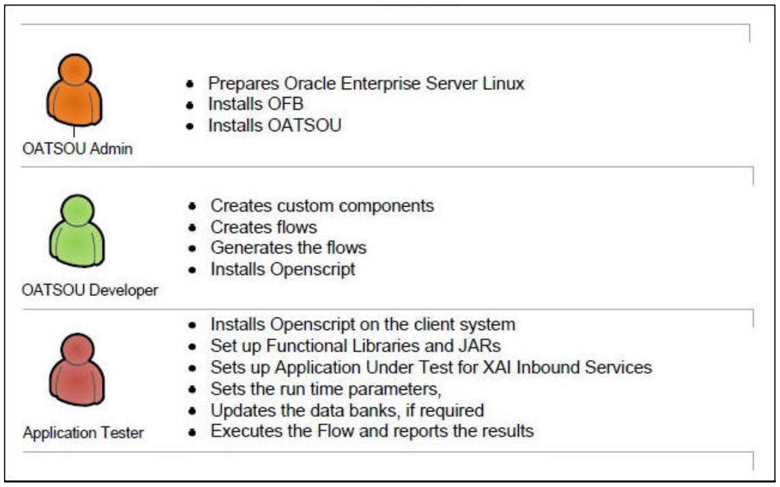
Below is a high-level architecture diagram for Oracle Application Testing Suite for Oracle Utilities.



Components and component sets are defined using metadata in Oracle Flow Builder. Using these components a flow can be assembled and then generated. The generated script, then, can be used and executed using Oracle Functional Tester.

## Understanding the Roles

The following diagram shows various roles and the setup tasks performed by each of the roles.


OATSOU Admin
Prepares Oracle Enterprise Server Linux
Installs OFB
Installs OATSOU
OATSOU Developer
Creates custom components
Creates flows
Generates the flows
Installs Openscript
Application Tester
Installs Openscript on the client system
Set up Functional Libraries and JARs
Sets up Application Under Test for XAI Inbound Services
Sets the run time parameters,
Updates the data banks, if required
Executes the Flow and reports the results

## Pointers for Getting Started

This section provides the references to get started with the Oracle Application Testing Suite for Oracle Utilities application.

### Installing Oracle Application Testing Suite for Oracle Utilities

See Chapter 2: Installing Oracle Application Testing Suite for Oracle Utilities for detailed installation instructions.

### Administrative Setup

See Chapter 2: Installing Oracle Application Testing Suite for Oracle Utilities for detailed administrative setup instructions.

### User Setup

See Chapter 3: Developing Metadata Driven Web Service Based Test Automation for the developer and/or user workstation setup instructions.

### Developing Test Automation

See Chapter 3: Developing Metadata Driven Web Service Based Test Automation for instructions on how to develop metadata driven Web service based automation tests using Oracle Application Testing Suite for Oracle Utilities.

### Product Components Reference

See the Oracle Utilities product specific component reference guide for more information.

Also, see the *Oracle Utilities Testing Suite for Oracle Utilities Release Notes* for the products included in this Oracle Application Testing Suite for Oracle Utilities release.

# Chapter 2

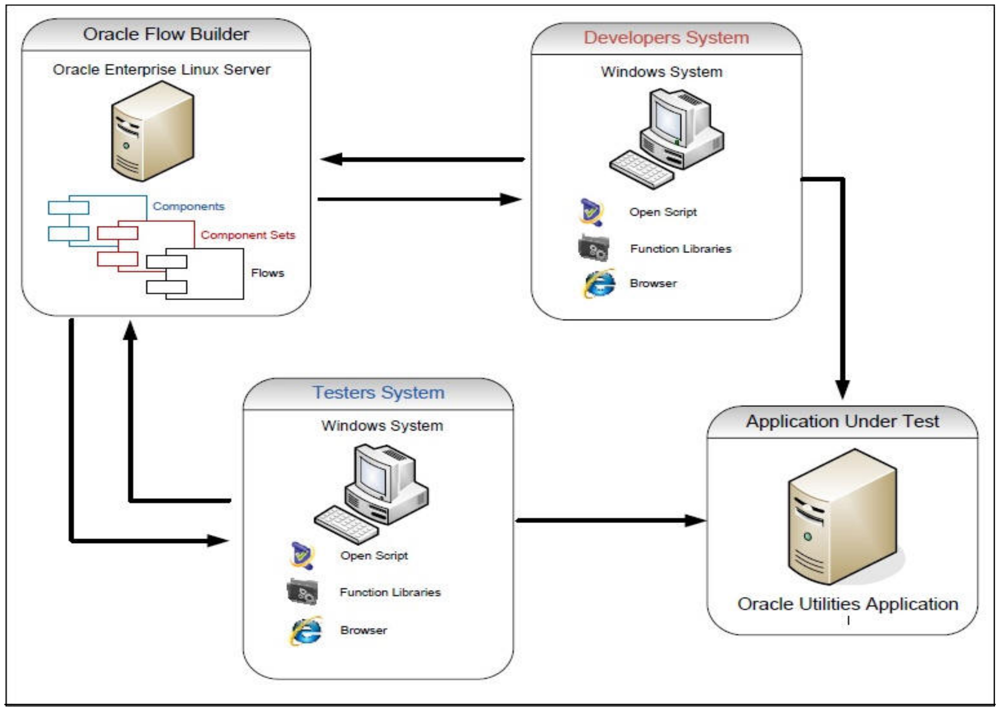
# Installing Oracle Application Testing Suite for Oracle Utilities

This chapter explains how to install and setup Oracle Application Testing Suite for Oracle Utilities. It includes the following sections:

- Understanding the Installation Topology
- Before You Install
- Setup Overview and Roles
- System Requirements
- Prerequisites
- Installing Oracle Application Testing Suite for Oracle Utilities

## Understanding the Installation Topology

The following diagram shows the topology of the Oracle Application Testing Suite for Oracle Utilities automation development environment.



## Before You Install

See the *Oracle Application Testing Suite for Oracle Utilities Release Notes* for important notes about this product release.

## Setup Overview and Roles

The following table provides an overview of the Oracle Application Testing Suite for Oracle Utilities application setup tasks.

| Step | Task | Role |
|---|---|---|
| 1 | Verify the system requirements.<br>See System Requirements | Administrator |
| 2 | Verify the prerequisites have been met.<br>See Prerequisites | Administrator |
| 3 | Install Oracle Flow Builder.<br>See *Oracle Flow Builder User Guide* ((http://www.oracle.com/technetwork/oem/downloads/index-084446.html) | Administrator |
| 4 | Perform the initial setup for Administrators.<br>See Installing Oracle Application Testing Suite for Oracle Utilities | Administrator |

| Step | Task | Role |
|---|---|---|
| 5 | Install Oracle Application Testing Suite for Oracle Utilities. See Installing Oracle Application Testing Suite for Oracle Utilities | Administrator |
| 6 | Import Oracle Application Testing Suite for Oracle Utilities export zip into Oracle Flow Builder. | Administrator |
| 7 | Install OpenScript. See Setting Up Automation Development Environment | User (Developer) |
| 8 | Download third party jars. See Downloading and Setting up Jars | User (Developer) |

## System Requirements

**Note:** See the *Oracle Flow Builder User's Guide* for the system requirements and certified platform details for Oracle Flow Builder.

The following table provides the browser, operating system, and database combination details for which this Oracle Application Testing Suite for Oracle Utilities release has been certified.

| System/Software | Requirement |
|---|---|
| Browser | Firefox 31ESR |
| Operating System (Client Admin) | Windows 7 (64-bit) |
| Operating System (Server) | Oracle Linux 6.5 (64-bit) |
| Chipset | x86_64 |
| Oracle Application Testing Suite (OATS) OpenScript | 12.4.0.2 |
| Database | Oracle Database 11.2.0.3 Enterprise Edition |
| Oracle Flow Builder (OFB) | 12.4.0.2 (Patch 20322564) |

### Supported Oracle Utilities Products

Test accelerators for the following Oracle Utilities products are provided by this Oracle Application Testing Suite for Oracle Utilities release.

| Product | Version |
|---|---|
| Oracle Real-Time Scheduler | 2.2.0.1 |
| Oracle Utilities Mobile Workforce Management | 2.2.0.1 |

## Prerequisites

The following table provides the prerequisite software details to install Oracle Application Testing Suite for Oracle Utilities in the respective machine categories:

| Machine Category | Requirement |
|---|---|
| Prerequisite Software (Server) | Oracle Flow Builder 12.4.0.2 (Patch 20322564)<br><br>Oracle Database 11.2.0.3 Enterprise Edition |
| Prerequisite Software (Client Admin) | Oracle Database Client 11.2.0.3 |
| Prerequisite Software (User) | Oracle Functional Tester 12.4.0.2 |

**Note**: See the *Oracle Flow Builder User's Guide* for steps about how to install Oracle Flow Builder and its prerequisite software.

**Note**: Set the database connection pool size (recommended) to 50 or higher, using the WebLogic console. Below is the pseudo format of the URL to access the WebLogic console:

http://<OFB_HOST>:<OFB_ADMIN_PORT>/console

## Installing Oracle Application Testing Suite for Oracle Utilities

This section provides instructions for installing Oracle Application Testing Suite for Oracle Utilities v4.2.0 and the post-installation checklist. It includes the following:

- Extracting OATSOU Packages
- Installing Components and Flows
- Initial Setup for Administrators

### Extracting OATSOU Packages

**Note**: Ensure that the prerequisite software is successfully installed before proceeding with the steps below.

To install the product successfully, follow these steps:

1. Download the Oracle Application Testing Suite for Oracle Utilities v4.2.0 Multiplatform part from Oracle Software Delivery Cloud (OSDC) (https://edelivery.oracle.com/) onto a Windows desktop.
2. Create the <TEMPDIR> directory.
3. Extract the zip file into the <TEMPDIR> created in the above step.
4. Change the directory to <TEMPDIR>\OATSOU_4.2.0.0.0\install directory.
5. Run **setup.exe**. The **Oracle Universal Installer Welcome** screen is displayed.
6. Click **Next**. The **Specify Home Details** screen is displayed.
7. Enter the **Name** and **Path** details as below:

   **Name**: <Name of the directory where Oracle Application Testing Suite for Oracle Utilities should be installed>

   **Path**: <Path where the installation directory should be created>
8. Click **Next**. The **Summary** screen is displayed.
9. Verify the details (global settings, product languages, space requirements, etc)

10. Click **Install**.
    This completes the setup process.

## Installing Components and Flows

After installing Oracle Application Testing Suite for Oracle Utilities, perform the following steps:

1. Login to any SQL client (sqlplus, for example).
2. Execute the following script for the intended Oracle Utilities product version:

   ```
   <OATSOU_HOME>/scripts/<PRODUCT>/<VERSION>/
   insert_update_ofb_metadata.sql
   ```

   **Note**: The script has to be executed as "OFB" user.

3. Navigate to the extracted installer folder at <OATSOU_HOME>\lib\MWM\<VERSION>, and then copy the Oracle Utilities Mobile Workforce Management function libraries and property files to the Oracle Flow Builder server at the following location:

   <OFB INSTALL DIR>/data/function-libraries/outsp-function-libs

4. Login to Oracle Flow Builder as an Administrator. The **Home** page is displayed.
5. Click the **Administration** tab.
6. Click the **Import** link under **Tools** in the left navigation menu.
7. Click **Browse** and navigate to the intended Oracle Utilities product export zip located in the following path:

   <OATSOU_HOME>/exports/<PRODUCT>/<VERSION>/<PRODUCT_PACK>.zip

   For example:

   <OATSOU_HOME>/exports/MWM/2.2.0.1/
   OATSOU_MWM_FLOWS_COMPONENTS_CNDDT_DUMP.zip

8. Click **Open**. The file name is populated in the **Upload Data** field.
9. Click **Save** to start the import process.

After the import is complete, the application is ready for use. The users can now proceed with the initial setup for Oracle Flow Builder.

## Initial Setup for Administrators

See the **Initial Setup for Administrators** section in the *Oracle Flow Builder User's Guide* for setup details.

To access the Oracle Application Testing Suite for Oracle Utilities application, users should register. See the **Registering User Credentials** section in the *Oracle Flow Builder User's Guide* for more details on how to register.

# Chapter 3

# Developing Metadata Driven Web Service Based Test Automation

The Oracle Application Testing Suite for Oracle Utilities components, component sets, and flows are organized in a tree hierarchy. This hierarchy compartmentalizes these for different Oracle Utilities applications.

This chapter is intended primarily for Automation Developers and Testers. It describes the procedures to create components and component sets, and to create and execute test work flows.

## Metadata Driven Automation Development Methodology

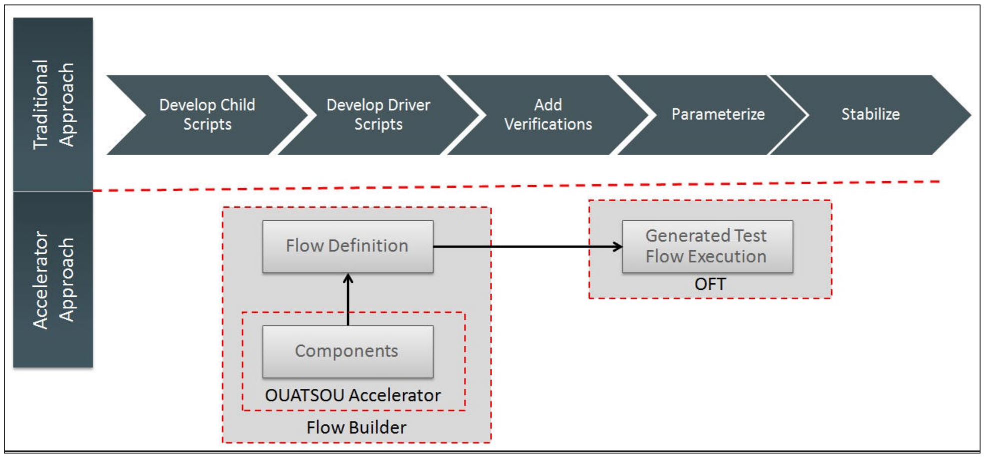
This section describes the metadata-driven automation development methodology that enables a test automation engineer to create automation scripts for an Oracle Utilities application.

An application has to be tested for its base functionality and extensions or customization. For this, the users create granular tests or larger end-to-end business test flows. Irrespective of the test design techniques, these tests can be used for regression testing the application in case of upgrades or customization to ensure that the existing functionality is not broken.

Typically, automation development is a time consuming exercise and teams have challenges in knowing and implementing the industry best practices and automation tools that work best for their product technology stack, helping them be successful in their efforts. Few of such challenges are as follows:

- Selecting an automation tool
- Creating the automation framework
- Identifying the automation development methodology
- Ensuring the automated tests are updated for new releases
- Ensuring the coverage levels are up to date
- Configuration management of automated test programs

The metadata-driven automation development methodology provides solutions to such challenges.



For the Oracle Utilities applications built on Oracle Utilities Application Framework, Web service based automated testing is proven to be more robust, maintainable, and faster to develop and execute. Hence, Oracle Application Testing Suite for Oracle Utilities comprises Web services based components that enable creation of test flows and executing the same. UI based automation components can also be created using Oracle Flow Builder. See the *Oracle Flow Builder User's Guide* for more information.

The following sections provide the test automation development phases in which an automated test flow is created.

## Planning

To plan an automated test flow, identify the business test flow to be automated and the components required for the flow. If necessary, create additional components or extend the delivered components. See the Extending Components section for details on how to extend the components.

## Design and Development

A flow design explains the order in which the components will be used to interact with each other in the flow. It also defines the test data combinations to use.

To design and develop an automated test flow, follow these steps:

1. Create/extend the required components that are identified in planning phase.
2. Create a test flow in Oracle Flow Builder that maps to the identified business test flow in the application.

   See the Creating Test Flows section for details on how to create a test flow. See the **Sample Work Flows** chapter in the *Oracle Application Testing Suite for Oracle Utilities Reference Guide for Mobile Workforce Management and Oracle Real-Time Scheduler* for delivered sample flows to understand the flow creation.
3. Drag and drop the required components into the flow.
4. Add the test data for the flow.

   The test data can be modified at the runtime using the standard OpenScript databanks. See the Test Data Management section for more details.
5. Assemble and generate the script for the test flow and then download the test script.

## Test Execution

To execute the automated test flow, execute the script in OpenScript Workbench.

To use another data set to execute the script, change the databanks in the generated scripts project, and then execute the script. See the Executing Test Flows section for more details.

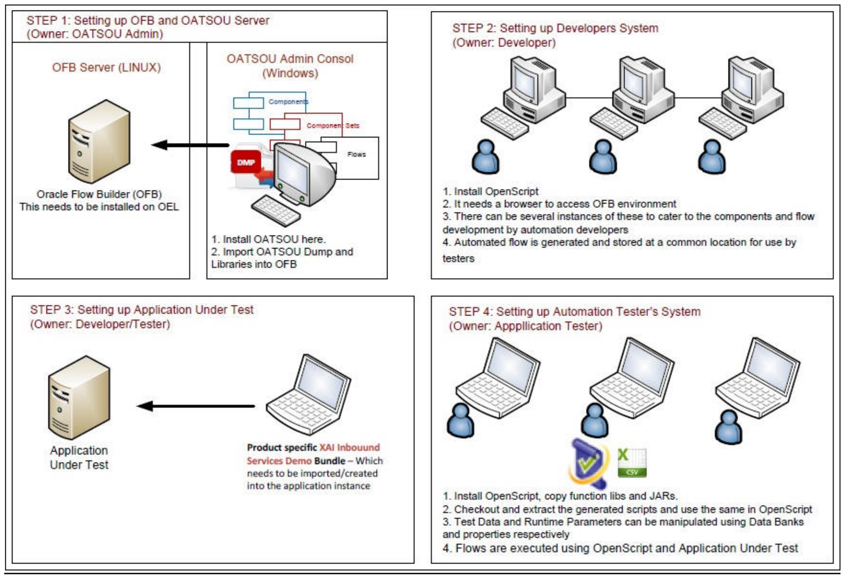
The components and test flows developed using this approach are stored and version controlled in the Oracle Flow Builder database. It takes care of the challenges in configuration management of automated tests. This methodology and framework works only with Oracle Functional Tester.

## Setting Up Automation Development Environment

The steps involved to set up the development environment for Oracle Application Testing Suite for Oracle Utilities are as follows:

- Step 1: Setting Up the OATSOU Server
- Step 2: Setting Up Workstations for Development
- Step 3: Setting Up the Source Application
- Step 4: Setting Up Workstations for Testing

The following figure explains the same.


STEP 1: Setting up OFB and OATSOU Server
(Owner: OATSOU Admin)
OFB Server (LINUX)
Oracle Flow Builder (OFB)
This needs to be installed on OEL
OATSOU Admin Consol
(Windows)
Components
Component Sets
Flows
DMP
1. Install OATSOU here.
2. Import OATSOU Dump and Libraries into OFB
STEP 2: Setting up Developers System
(Owner: Developer)
1. Install OpenScript
2. It needs a browser to access OFB environment
3. There can be several instances of these to cater to the components and flow development by automation developers
4. Automated flow is generated and stored at a common location for use by testers
STEP 3: Setting up Application Under Test
(Owner: Developer/Tester)
Application Under Test
Product specific XAI Inbouund Services Demo Bundle – Which needs to be imported/created into the application instance
STEP 4: Setting up Automation Tester's System
(Owner: Appplication Tester)
CSV
1. Install OpenScript, copy function libs and JARs.
2. Checkout and extract the generated scripts and use the same in OpenScript
3. Test Data and Runtime Parameters can be manipulated using Data Banks and properties respectively
4. Flows are executed using OpenScript and Application Under Test

### Step 1: Setting Up the OATSOU Server

This section explains the steps to be performed to setup the server.

- Installing Oracle Flow Builder
- Installing Oracle Application Testing Suite for Oracle Utilities
- Administrative Tasks

### Installing Oracle Flow Builder

Ensure that Oracle Flow Builder is installed before installing Oracle Application Testing Suite for Oracle Utilities. See *Oracle Flow Builder User's Guide* for instructions to install Oracle Flow Builder.

> **Note**: The database connection pool size has to be (recommended) set to 50 or higher, using the WebLogic console.
>
> Below is the pseudo format of the URL to access the WebLogic console:
>
> http://<OFB_HOST>:<OFB_ADMIN_PORT>/console

### Installing Oracle Application Testing Suite for Oracle Utilities

Oracle Application Testing Suite for Oracle Utilities has to be installed on a client workstation. See the Installing on Client Admin Workstation section for installation instructions.

> **Note:** Oracle Application Testing Suite for Oracle Utilities need not be installed on the user workstations. Users only need browser access to Oracle Flow Builder for the component and flow development.

### Administrative Tasks

See the *Oracle Flow Builder User's Guide* for the Oracle Flow Builder administrative tasks (such as stopping and starting the application instance).

## Step 2: Setting Up Workstations for Development

This section provides the steps to set up the Oracle Application Testing Suite for Oracle Utilities developer workstations. The tasks include:

- Installing OpenScript
- Creating Test Execution Folder Structure
- Downloading and Setting up Jars

### Installing OpenScript

Ensure OpenScript is installed on each user workstation where automation execution is performed or where component and flow development is intended to be performed. See the System Requirements section for the certified OpenScript version details.

### Creating Test Execution Folder Structure

See the Creating Folder Structure for Generated Scripts section to create the folder structure for placing the necessary test artifacts.

### Downloading and Setting up Jars

See the Creating Folder Structure for Generated Scripts section to download and set up the necessary .jar files.

## Step 3: Setting Up the Source Application

See the respective Oracle Utilities' application specific installation guide for the setup details.

## Step 4: Setting Up Workstations for Testing

This section provides the steps to set up the Oracle Application Testing Suite for Oracle Utilities workstations for test environment. The tasks include:

- Importing XAI Inbound Web Services Demo Bundle
- Installing OpenScript
- Creating a Test Execution Folder Structure
- Using the configuration.properties file
- Downloading and Setting up Jars

### Importing XAI Inbound Web Services Demo Bundle

Import the XAI Web services demo bundle on the Oracle Utilities product application instance before executing the Web services automation code. This bundle is found in the following location:

<OATSOU_HOME>/exports/<PRODUCT>/<VERSION>/XAI Inbound Services Demo Export for <PRODUCT>.xml

To import the bundle, follow these steps:

1. Open the Oracle Utilities application.
2. Navigate to **Admin** > **X** > **XAI Import Service**.
3. Click **Browse**.
4. Select the XAI export bundle.
5. Click **Save**.

   **Note**: To override the existing services, select the **Override Existing Services** check box.

   **Note**: See the application-specific *Oracle Application Testing Suite for Oracle Utilities Reference Guide* for details about creating XAI Inbound Services in the respective application.

### Installing OpenScript

Install OpenScript on each test workstation where automation execution has to be performed. See the System Requirements section for the certified OpenScript version details.

### Creating a Test Execution Folder Structure

See the Creating Folder Structure for Generated Scripts section to create the folder structure for placing the necessary test artifacts.

### Using the configuration.properties file

Update the configuration.properties file to suit the test execution product setup requirements. See the Configuring the Runtime Properties section to understand the properties to be configured.

### Downloading and Setting up Jars

See the Creating Folder Structure for Generated Scripts section to download and set up the necessary .jar files.

## Creating Web Service Based Components

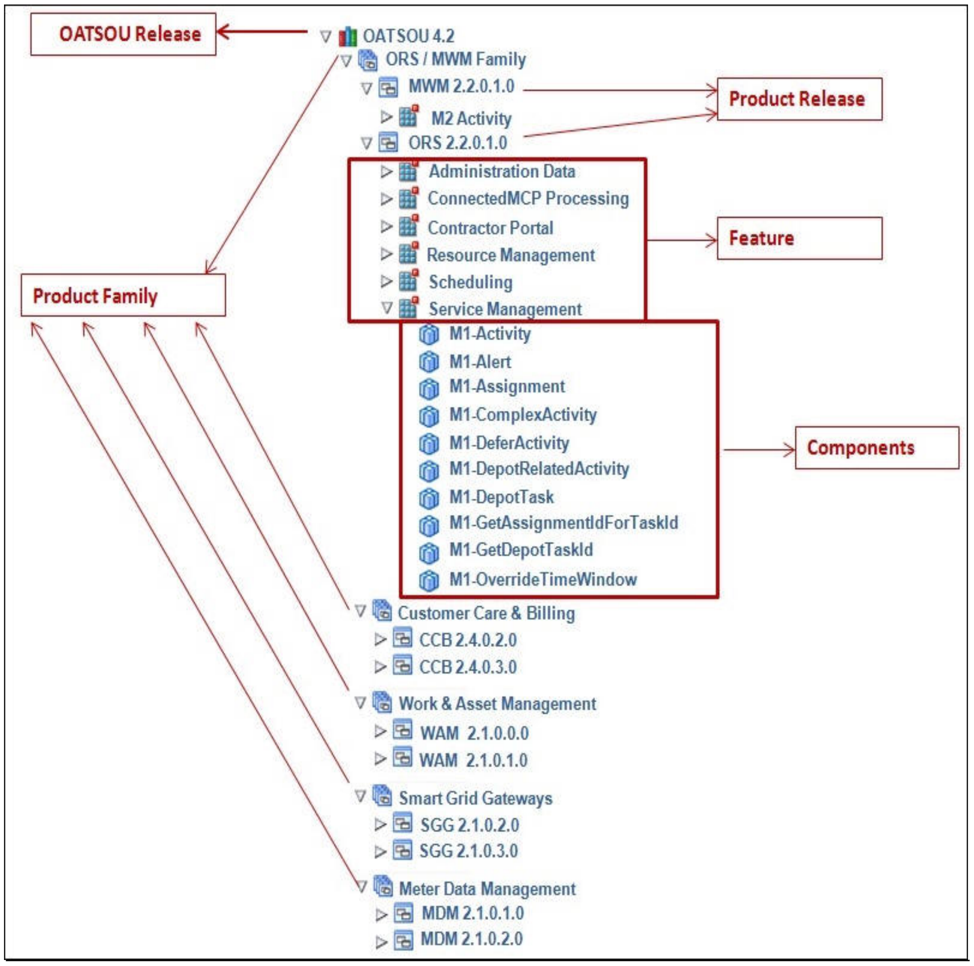
This section describes the component hierarchy in Oracle Application Testing Suite for Oracle Utilities. The hierarchy is organized as follows:

> Oracle Application Testing Suite for Oracle Utilities Release > Product Family > Product Release > Features > Components.

The section also explains the steps to create a Web service based component. It includes:

- Creating Components
- Keywords, Definitions, and the Usage
- Handling the List Elements
- Setting Up Transaction Types
- Using Runtime Variables in Components
- Resolving the Repeating Elements in Response XML
- Adding Validations
- Logging and Reporting
- Extending Components
- Using Function Libraries

The figure below shows the high-level component structure.



## Creating Components

Ensure the component is created under the accurate hierarchy level.

To create a component, follow these steps:

1. Navigate to the component tree where the component has to be created.
2. Right-click the feature in the component tree.

   **Note**: Create a new feature folder if it is not found in the delivered tree structure.

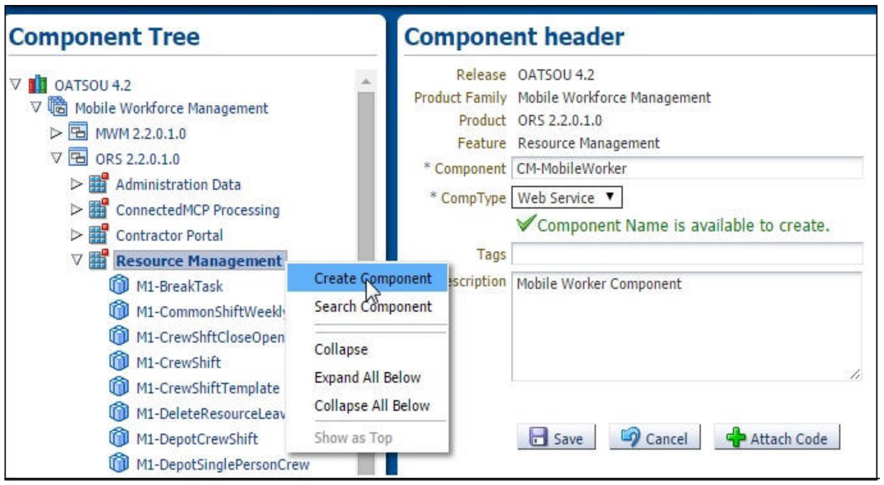
   For example: To create a "CM-MobileWorker" component under the "Resource Management" feature for the product release ORS 2.2.0.1.0:

   a. Navigate to **OATSOU 4.2** > **ORS / MWM Product Family** > **ORS 2.2.0.1.0** > **Resource Management**.

   b. Right-click the **Resource Management** feature.

   c. Select **Create Component**.

   **Note**: The component name must be prefixed with 'CM' and the **Tags** field should have a CM tag for every component. The tagging enables porting the custom components to latest Oracle Application Testing Suite for Oracle Utilities release.



Release OATSOU 4.2
Product Family Mobile Workforce Management
Product ORS 2.2.0.1.0
Feature Resource Management
* Component CM-MobileWorker
* CompType Web Service
✓Component Name is available to create.
Tags
Description Mobile Worker Component

3. Enter **CM-MobileWorker** in the **Component** field to name the component.
4. Select **Web Service** in the **CompType** drop-down list.
5. Enter a description in the **Description** field.
6. Click **Attach Code** to add the metadata. The **Component** window is displayed.
7. Create component lines. See the Creating a Component Line section for more information about creating component lines.
8. Click **Save & Unlock** to save and create the component.

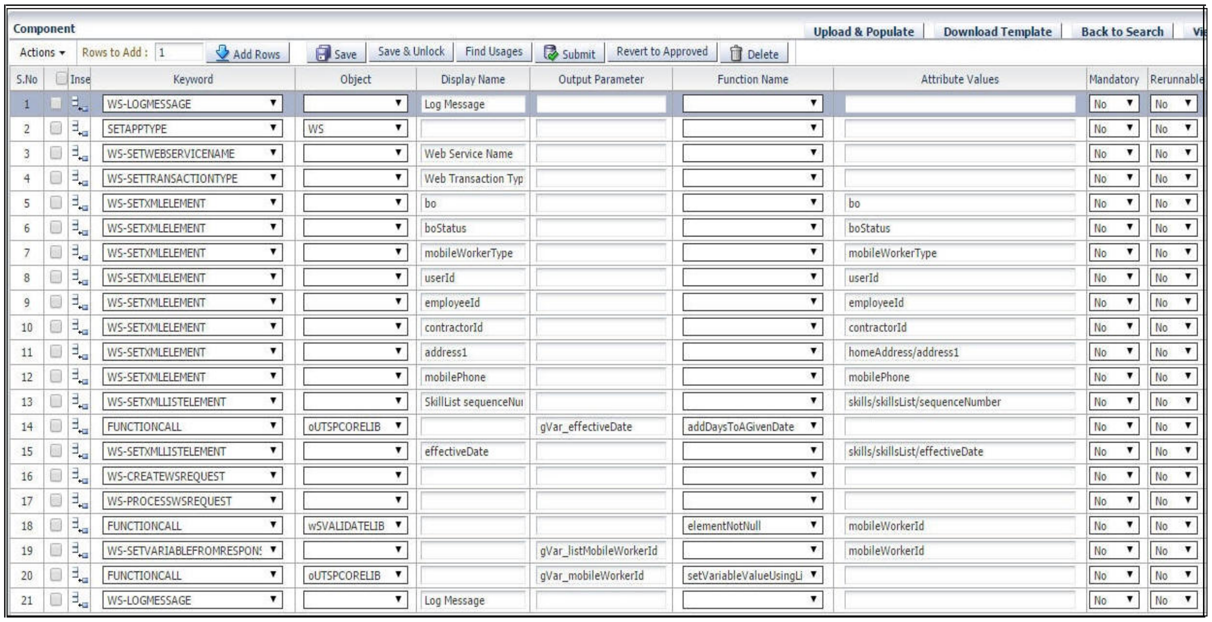
## Creating a Component Line

A component consists of several component lines. Each component line comprises a keyword, display name, attribute value, library, function, output parameters, rerunnable flag, mandatory flag.

The list below describes each entity in a component line. See the *Oracle Flow Builder User's Guide* for more details.

- **Keyword**: Specifies the step to be performed. Example: WS-SETVARIABLEFROM RESPONSE, WS-VALIDATE, etc
- **Object**: Specifies the Oracle Application Testing Suite for Oracle Utilities function library name from where the function is called.
- **Display Name**: Indicates the component line description.
- **Attribute Values**: Specifies the Web service XML tag name used as variable to store its value.
- **Default Data**: Stores the default data used in the component line.
- **Function Name**: Stores the function name called from the library.
- **Output Parameters**: Stores the output in the form of a variable.
- **Rerunnable**: Specifies "Yes" to create unique data for rerunnable. Rerunnable appends a random variable to the test data.
- **Mandatory**: Specifies "Yes" for all the mandatory fields.
- **Tooltip**: Presents the data as a tool tip during the flow creation.

The figure below shows the **Component** page with the available component lines.



| S.No | | Inse | Keyword | Object | Display Name | Output Parameter | Function Name | Attribute Values | Mandatory | Rerunnable |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | WS-LOGMESSAGE | | Log Message | | | | No | No |
| 2 | | | SETAPPTYPE | WS | | | | | No | No |
| 3 | | | WS-SETWEBSERVICENAME | | Web Service Name | | | | No | No |
| 4 | | | WS-SETTRANSACTIONTYPE | | Web Transaction Typ | | | | No | No |
| 5 | | | WS-SETXMLELEMENT | | bo | | | bo | No | No |
| 6 | | | WS-SETXMLELEMENT | | boStatus | | | boStatus | No | No |
| 7 | | | WS-SETXMLELEMENT | | mobileWorkerType | | | mobileWorkerType | No | No |
| 8 | | | WS-SETXMLELEMENT | | userId | | | userId | No | No |
| 9 | | | WS-SETXMLELEMENT | | employeeId | | | employeeId | No | No |
| 10 | | | WS-SETXMLELEMENT | | contractorId | | | contractorId | No | No |
| 11 | | | WS-SETXMLELEMENT | | address1 | | | homeAddress/address1 | No | No |
| 12 | | | WS-SETXMLELEMENT | | mobilePhone | | | mobilePhone | No | No |
| 13 | | | WS-SETXMLLISTELEMENT | | SkillList sequenceNui | | | skills/skillsList/sequenceNumber | No | No |
| 14 | | | FUNCTIONCALL | oUTSPCORELIB | | gVar_effectiveDate | addDaysToAGivenDate | | No | No |
| 15 | | | WS-SETXMLLISTELEMENT | | effectiveDate | | | skills/skillsList/effectiveDate | No | No |
| 16 | | | WS-CREATEWSREQUEST | | | | | | No | No |
| 17 | | | WS-PROCESSWSREQUEST | | | | | | No | No |
| 18 | | | FUNCTIONCALL | wSVALIDATELIB | | | elementNotNull | mobileWorkerId | No | No |
| 19 | | | WS-SETVARIABLEFROMRESPON! | | | gVar_listMobileWorkerId | | mobileWorkerId | No | No |
| 20 | | | FUNCTIONCALL | oUTSPCORELIB | | gVar_mobileWorkerId | setVariableValueUsingLi | | No | No |
| 21 | | | WS-LOGMESSAGE | | Log Message | | | | No | No |

Add the required component lines using the **Keyword** drop-down list to define the Web services based component.

See the Keywords, Definitions, and the Usage section for a list of keywords used to define the Web service based components. See the *Oracle Flow Builder User's Guide* for a complete list of generic keywords.

This example shows different component lines created for the CM-MobileWorker component.

1. Select SETAPPTYPE in the **Keyword** drop-down list to define the application type. Then, select "WS" in the **Object** drop-down list to denote that it is a Web services based component.

2. Select the WS-SETWEBSERVICENAME keyword to define the Web service name.
3. Select the WS-SETTRANSACTIONTYPE keyword to define the transaction type of the Web service call.

   **Note**: The final script of a component is Web service call to create, update, and delete.
4. Select the WS-LOGMESSAGE keyword to log comments in component definition. This helps in debugging the script code for that component.
5. Select the WS-SETXMLELEMENT keyword to set the value into a specific element of request xml.

   Consider a component "CM-MobileWorker" in Oracle Utilities Mobile Workforce Management. This component maps to the MobileWorker business object. It includes elements, such as:

   ```
   <mobileWorkerType />
   <contractorId />
   ```
6. Select the WS-SETXMLLISTELEMENT keyword to set a value into the list element tags. The list element is 'skills'.
7. Click **Save**.

## Keywords, Definitions, and the Usage

This section provides the description, definition (use case), and the usage details (object, display name, attribute values, default data, function name, and output parameters) for each of the keywords used to define Web service based components.

### WS-SETWEBSERVICENAME

Sets the name of the application Web service.

**Use Case**: Defines the Web service to which the component's Web service request is sent. The Web service name is provided in the attribute values column during the component development. This service name is appended with the WebContainerURL to form a complete WSDL URL for processing the request. The WebContainerURL has to be specified in the flow runtime configuration property file.

| Usage Details | Value |
|---|---|
| Keyword | WS-SETWEBSERVICENAME |
| Display Name | User Defined Display Name |
| Attribute Values | Web Service Name |

### WS-SETXMLELEMENT

Sets the element (Xpath) value in the Web service request using either a variable or a value.

**Use Case**: Enables the Web service creation request (XML) with the element values populated by setting each value for the defined element.

| Usage Details | Value |
|---|---|
| Keyword | WS-SETXMLELEMENT |
| Display Name | User Defined Display Name |
| Attribute Values | Xpath of the element |

### WS-SETXMLLISTELEMENT

Sets the repeating list element (Xpath) value in the Web service request using either a variable or a value.

**Use Case**: Enables the Web service creation request (XML) with repeating list element values populated by setting each value set for the defined element list. The values are provided from the test data.

| Usage Details | Value |
|---|---|
| Keyword | WS-SETXMLLISTELEMENT |
| Display Name | User Defined Display Name |
| Attribute Values | Xpath of the element |

### WS-SETVARIABLE

Sets a value to a global variable.

**Use Case**: Used for setting a value to a global variable used across the flow for validations or for setting XML elements. The values are provided from the test data.

| Usage Details | Value |
|---|---|
| Keyword | WS-SETVARIABLE |
| Display Name | User Defined Display Name |
| Output Parameters | Variable Name |

### WS-SETVARIABLEFROMRESPONSE

Used to retrieve the XML element value from the response and stores it in a global variable for further processing.

**Use Case**: Enables use of a response value, such as ID from a component, as an input to a request in another component.

| Usage Details | Value |
|---|---|
| Keyword | WS-SETVARIABLEFROMRESPONSE |
| Display Name | User Defined Display Name |
| Attribute Values | Xpath of the element in response |
| Output Parameters | Variable Name |

### WS-SETTRANSACTIONTYPE

Sets a value for the transaction type.

**Use Case**: Used to set a value to a transaction type variable used in the request XML to pass a request for specific operations, such as ADD, UPDATE, READ, DELETE, etc. The transaction type is provided from the test data.

| Usage Details | Value |
|---|---|
| Keyword | WS-SETTRANSACTIONTYPE |
| Display Name | User Defined Display Name |

### WS-LOGMESSAGE

Used to set custom log messages in the execution results report.

**Use Case**: Provides the necessary extensibility to provide custom log messages for the generated results report, such as to identify the start and completion of a transaction, etc.

| Usage Details | Value |
|---|---|
| Keyword | WS-SETLOGMESSAGE |
| Display Name | User Defined Value |
| Attribute Values | Message |

### WS-CREATEWSREQUEST

Creates a Web service request XML and stores it in the "WSDLXML" global variable.

**Use Case**: Enables the manipulation of the Web service XML request generated before submitting it to the application for processing, giving greater flexibility in development.

| Usage Details | Value |
|---|---|
| Keyword | WS-CREATEWSREQUEST |
| Display Name | User Defined Display Name |
| Attribute Values | Web Service Name |

### WS-PROCESSWSREQUEST

Sends the Web services request and receives the response from the application for the specified WSDL URL.

**Use Case**: Posts the generated XML request from WS-CREATEWSREQUEST to the application, and then processes the response. This keyword performs the core process of the Web services based request-response model.

| Usage Details | Value |
|---|---|
| Keyword | WS-PROCESSWSREQUEST |
| Display Name | User Defined Display Name |
| Attribute Values | Web Service Name |

## Handling the List Elements

Use the WS-SETXMLLISTELEMENT keyword to define a list element of request XML. The XML schema for the CM-MobileWorker component has the 'Skills' list element. The element "skills/skillsList" has multiple occurrences in the 'skills' element.

The WS-SETXMLLISTELEMENT keyword allows you to enter data for such elements while entering the test data.

```
<skills type="group">
<skillsList type="list" mapChild="M1_RESRC_CAP">
<mobileWorkerId suppress="true" mapField="RESRC_ID"
dataType="string"/>   <sequenceNumber suppress="true"
dataType="number" mapField="SEQNO"/>
```

```
<skill mdField="M1_SKILL" fkRef="M1-SKILL" mapField="CAP_TYPE_CD"
dataType="string"/>  <effectiveDate dataType="date"
mapField="EFF_DT"/>
<expirationDate dataType="date" mapField="M1_EXP_DT"/>
</skillsList>
</skills>
```

## Setting Up Transaction Types

A transaction type determines the action to be taken while executing a Web service request. It can be a CREATE, UPDATE, READ, or DELETE operation. The value for the keyword "WS-SETTRANSACTIONTYPE" is specified while adding the test data for the flow. If designed so, the same component can be used to add, update, or delete operations. For example: To create a new mobile worker, or to update or delete an existing mobile worker, set up the transaction type for appropriate the instance of the component in the flow.

## Using Runtime Variables in Components

In some cases, few elements from the response component execution have to be passed as inputs to another component's request XML. To achieve this, store the output of first component in the global variable by using the WS-SETVARIABLEFROMRESPONSE keyword. This keyword requires Xpath of the response element whose value are to be stored. It should be specified in the **Attribute Values** column. The global variable which holds this value in the script is defined in the **Output Parameter** column.

The WS-SETVARIABLEFROMRESPONSE keyword stores the mobileWorkerId obtained after a mobile worker component execution to the global variable gVar_mobileWorkerId1 declared in the **Output Parameter** column.

See the Creating Test Flows section for procedure about how a dependent component reads such global variables

## Resolving the Repeating Elements in Response XML

If the response XML has repeating elements, the value embedded within the repeating elements is retrieved as below.

Consider the response as follows:

```
<ContactDetails>
<Phone> 123-456-7890 </Phone>
<Phone>234-567-8901 </Phone>
<email> joe@oracle.com </email>
</ContactDetails>
```

1. Use the WS-SETVARIABLEFROMRESPONSE keyword to retrieve the response of the Web service invocation into the global variable. gVar1 is defined in the **Output Parameter** column as below:

   The keyword resolves all occurrences of the Phone element and stores all values in the gVar1 variable separated by comma. gVar1 will be set to "123-456-7890,234-567-8901".

2. Use the FUNCTIONCALL keyword to call the setVariableValueUsingListIndex function available in the OUTSPCORE library.

   The keyword retrieves the value(s) based on the parameters passed. Parameters passed are global variables storing the values (gVar1 and index).

See Chapter 4: Function Library Reference for more details.

## Adding Validations

To validate the response, use the FUNCTIONCALL keyword to validate the content, in particular, the Xpath of response XML. Select the function library wSVALIDATELIB from the **Object** drop-down list and the function to be called from the **Function Name** drop-down list.

See Chapter 4: Function Library Reference for a complete reference of the Validation function library.

## Logging and Reporting

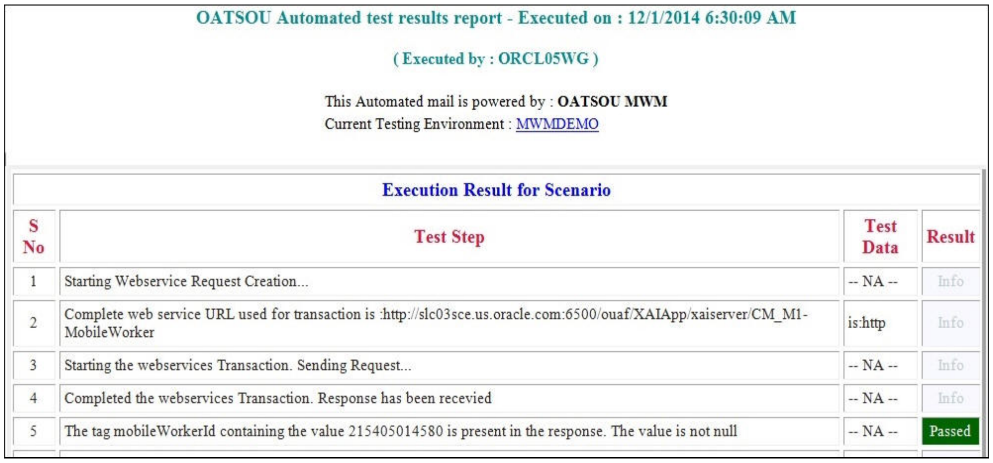
Apart from OpenScript, Oracle Application Testing Suite for Oracle Utilities provides the following types of logging and reporting:

1. **Test execution log file**: The test execution logs are created in the **Logs** folder and separate logs are generated for each flow.

2. **Email report in HTML format**: The test execution email provides brief information about the overall test execution. It comprises of the following:

   - Test step
   - Test data
   - Result (Pass/Fail)

The figure below is a snippet of a generated email.



OATSOU Automated test results report - Executed on : 12/1/2014 6:30:09 AM

( Executed by : ORCL05WG )

This Automated mail is powered by : **OATSOU MWM**
Current Testing Environment : MWMDEMO

Execution Result for Scenario

| S No | Test Step | Test Data | Result |
|---|---|---|---|
| 1 | Starting Webservice Request Creation... | -- NA -- | Info |
| 2 | Complete web service URL used for transaction is :http://slc03sce.us.oracle.com:6500/ouaf/XAIApp/xaiserver/CM_M1-MobileWorker | is:http | Info |
| 3 | Starting the webservices Transaction. Sending Request... | -- NA -- | Info |
| 4 | Completed the webservices Transaction. Response has been recevied | -- NA -- | Info |
| 5 | The tag mobileWorkerId containing the value 215405014580 is present in the response. The value is not null | -- NA -- | Passed |

## Extending Components

The components that are delivered can be customized. However, modifying the existing components is not allowed. Any attempt to do so will not allow other components to work with the modified component.

A component can be extended by making its copy or clone and saving it with a different name, and then adding or modifying the metadata or key words as below.

1. Right-click a component and select **Copy Component**.
2. Select and right-click **Feature** to create a clone, and then select **Paste Component**.
3. Enter a new name (that is different from the component being cloned). The name should always be prefixed with 'CM'.
4. Add the '**CM**' tag in the **Tag** field, and then click **Save**.

Right-click this cloned component and select **Attach Code** to modify it. Follow the steps in the Creating Components section to proceed with modifications.

## Using Function Libraries

This section explains how to use the function libraries shipped with this Oracle Application Testing Suite for Oracle Utilities release and create new help libraries.

Function libraries shipped with Oracle Flow Builder and Oracle Application Testing Suite for Oracle Utilities can be accessed in the **Component** window using the FUNCTIONCALL key word and specifying the library name in the **Object** column and the function name in the **Function Name** column. Define the variable name in the **Output Parameters** field to store the return value of the function.

Function parameters can be provided while entering test data for the component in a flow. See the Test Data Management section for more details.

See Chapter 4: Function Library Reference for a list of libraries and functions available in Oracle Application Testing Suite for Oracle Utilities.

## Creating Component Sets

A component set is a group of components arranged in a pre-determined order. It can be used to perform a set of tasks that are repeatable.

For example: The M1-CrewShift component in Oracle Utilities Mobile Workforce Management is actually a component set. To create an M1-CrewShift component in a flow, ensure M1-MobileWorker, M1-Vehicle, and M1-MultiPersonCrew are available. Then, create the dependant M1-CrewShift component. To avoid dragging and dropping these four components to a flow, they can be grouped in a sequence to form a component set. It can later be used to create flows.

To create a component set, follow these steps:

1. Navigate to the component set tree to locate the feature under which the component set should be created.
2. Right-click the feature and select **Create Component Set**. The **Component Set** page is displayed.
3. Enter a unique component name **CM-CrewShift** and add a **Description**.
4. Click **Create Structure**.
5. Drag and drop the components M1-MobileWorker, M1-Vehicle, M1-MultiPersonCrew, M1-CrewShift to the component set structure root.
6. Click **Unlock** to remove the lock on the component set.

## Creating Test Flows

Test flows are actual business tests executed on the application under test. The flows are assembled in Oracle Flow Builder by using predetermined components and are updated with data to guide the flow execution.

This section explains the steps to create a flow. It also includes:

- Creating Scenarios
- Using Global Variables
- Test Data Management
- Adding the Email Capabilities to Flows

To create a flow, identify the components required to create the flow.

> **Note**: The components delivered with Oracle Application Testing Suite for Oracle Utilities have to be extended or new components have to be created.

To create a flow, follow these steps:

1. Navigate to the feature in the flow tree to create the flow.

   For example: The "Assign and Complete a CrewShift" flow includes the following components:

   - M1-MobileWorker
   - M1-Vehicle
   - M1-MultiPersonCrew
   - M1-CrewShift (to create a crew shift)
   - M1-Activity (to create an activity)
   - M1-CrewShift (to plan a crew shift)
   - M1-CheckIfActivityIsScheduled (to check if the activity has been scheduled)
   - M1-CrewShift (to start the crew shift)
   - M1-GetAssignmentIdForTaskId (to get the assignment ID for the activity)
   - M1-Assignment (to enroute to the assignment)
   - M1-Assignment (to start the assignment)
   - M1-Assignment (to complete the assignment)
   - M1-CrewShift (to complete the crew shift)
2. Right-click the product "ORS 2.2.0.1.0" under "ORS / MWM Product Family" and select **Create Flow**.
3. In the **Create Flow** pane, enter the **Flow Name**, **Flow Type**, and **Description**.
4. Click **Create Structure** in the **Create Flow** pane. This creates a new flow and a new scenario is added to the flow tree.
5. Expand the flow tree and select the scenario to add the components.
6. Drag and drop the components from the **Select Component** pane to the **Flow Creation** pane. See the Creating Scenarios section for information about adding scenarios to a flow.
7. The test data needs to be entered at the component level while defining a flow and before the flow is assembled. To add data for M1-MobileWorker component, right click it and select **Enter Test Data**. Similarly, data can be added for the remaining components.
8. Enter the test data in the **Enter Component Test Data** page.
9. Click **Save & Close** to return to the **Flow Creation** page.
10. Click **Assemble**.
    This completes the procedure to define a test flow in Oracle Flow Builder.

## Creating Scenarios

See the **Adding Scenarios to Flow** section in the *Oracle Flow Builder User's Guide* to understand how to create and use scenarios in the flow.

## Using Global Variables

This section explains the usage of global variables to pass data across components.

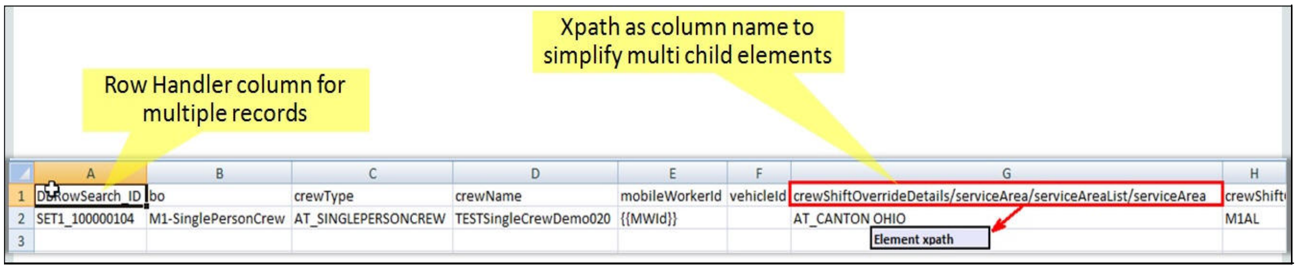
The component M1-CrewShift is a dependant component and during runtime needs The IDs of the M1-MobileWorker, M1-Vehicle, and M1 MultiPersonCrew components in addition to the other data.

To add component references to a dependant component (M1-CrewShift), follow these steps:

1. To add the MobileWorker ID, select **gVar_mobileWorkerId1** from the **Value1** drop-down list against the **resourceAllocationList/resourceId** display name.
2. To add VehicleId, select **gVar_vehicleId1** from the **Value2** drop-down list against the **resourceAllocationList/resourceId** display name.
3. To add MultiPersonCrewId, select **gVar_CrewId** in the **Value 1** drop-down list against the **crewId** display name.

## Test Data Management

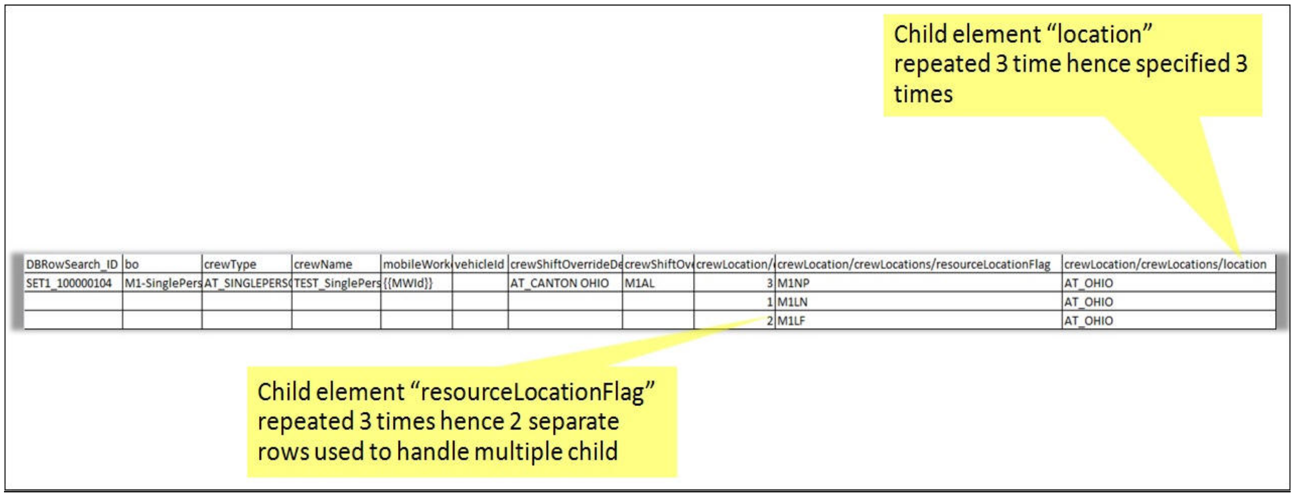
The test data can be mentioned in the flow meta data. The Oracle Flow Builder code generator generates OpenScript databanks (CSV files) for each component. Number and names of the columns in the generated databanks are based on the test data provided. The databanks can be updated with new data before test execution.


Xpath as column name to simplify multi child elements
Row Handler column for multiple records
Element xpath

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | DBRowSearch_ID | bo | crewType | crewName | mobileWorkerId | vehicleId | crewShiftOverrideDetails/serviceArea/serviceAreaList/serviceArea | crewShift |
| 2 | SET1_100000104 | M1-SinglePersonCrew | AT_SINGLEPERSONCREW | TESTSingleCrewDemo020 | {{MWId}} | | AT_CANTON OHIO | M1AL |
| 3 | | | | | | | | |

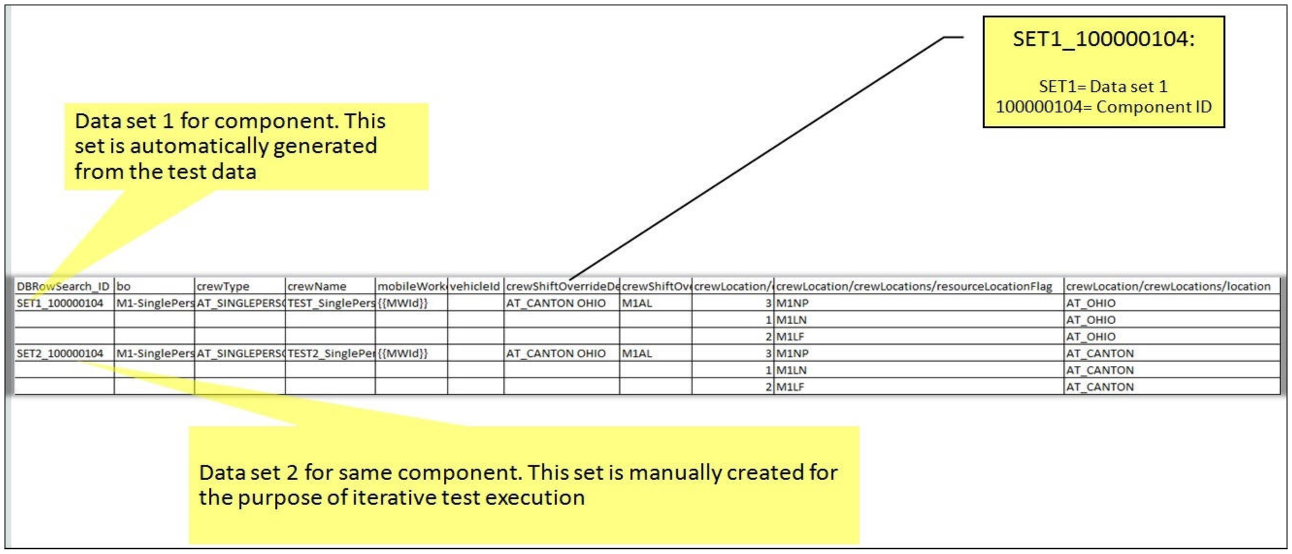
The generated script for the test flow can be executed for multiple sets of data. The data sets have to be provided in the component databank CSV for each of the component. The first data set for a flow will be generated by the script generator using the Oracle Flow Builder data.

**Case 1**: The component is called just once in the flow and it has repeating list elements (such as location). The figure below shows the CSV generated for the component.


Child element "location" repeated 3 time hence specified 3 times
Child element "resourceLocationFlag" repeated 3 times hence 2 separate rows used to handle multiple child

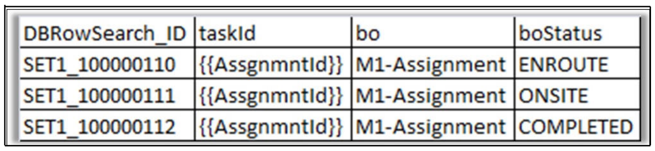
| DBRowSearch_ID | bo | crewType | crewName | mobileWork | vehicleId | crewShiftOverrideDe | crewShiftOv | crewLocation/ | crewLocation/crewLocations/resourceLocationFlag | crewLocation/crewLocations/location |
|---|---|---|---|---|---|---|---|---|---|---|
| SET1_100000104 | M1-SinglePers | AT_SINGLEPERS | TEST_SinglePers | {{MWId}} | | AT_CANTON OHIO | M1AL | 3 | M1NP | AT_OHIO |
| | | | | | | | | 1 | M1LN | AT_OHIO |
| | | | | | | | | 2 | M1LF | AT_OHIO |

The figure below shows the CSV after adding a second set of data. (Since the data has repeating list elements, the second data set starts two rows after the first with the prefix SET2_).


SET1_100000104:
SET1= Data set 1
100000104= Component ID
Data set 1 for component. This set is automatically generated from the test data
Data set 2 for same component. This set is manually created for the purpose of iterative test execution

| DBRowSearch_ID | bo | crewType | crewName | mobileWork | vehicleId | crewShiftOverrideDe | crewShiftOve | crewLocation/ | crewLocation/crewLocations/resourceLocationFlag | crewLocation/crewLocations/location |
|---|---|---|---|---|---|---|---|---|---|---|
| SET1_100000104 | M1-SinglePers | AT_SINGLEPERS | TEST_SinglePers | {{MWId}} |  | AT_CANTON OHIO | M1AL | 3 | M1NP | AT_OHIO |
|  |  |  |  |  |  |  |  | 1 | M1LN | AT_OHIO |
|  |  |  |  |  |  |  |  | 2 | M1LF | AT_OHIO |
| SET2_100000104 | M1-SinglePers | AT_SINGLEPERS | TEST2_SinglePer | {{MWId}} |  | AT_CANTON OHIO | M1AL | 3 | M1NP | AT_CANTON |
|  |  |  |  |  |  |  |  | 1 | M1LN | AT_CANTON |
|  |  |  |  |  |  |  |  | 2 | M1LF | AT_CANTON |

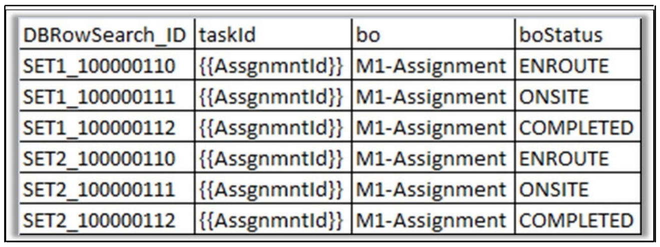
**Case 2**: The component is called more than once in the flow and it does not have repeating list elements. The figure below shows the CSV generated for the component. In the figure, the text enclosed in the curly brackets is the variable.



| DBRowSearch_ID | taskId | bo | boStatus |
|---|---|---|---|
| SET1_100000110 | {{AssgnmntId}} | M1-Assignment | ENROUTE |
| SET1_100000111 | {{AssgnmntId}} | M1-Assignment | ONSITE |
| SET1_100000112 | {{AssgnmntId}} | M1-Assignment | COMPLETED |

**Note:** The **taskId** column points to the **AssgnmntId** global variable declared within the curly brackets. The value stored in **AssgnmntId** will be set from the execution or the previous component and stored in **AssgnmntId**.

The figure below shows the CSV values after adding the second data set. (since the data has repeating list elements, the second data set starts two rows after the first with the prefix SET2_).



| DBRowSearch_ID | taskId | bo | boStatus |
|---|---|---|---|
| SET1_100000110 | {{AssgnmntId}} | M1-Assignment | ENROUTE |
| SET1_100000111 | {{AssgnmntId}} | M1-Assignment | ONSITE |
| SET1_100000112 | {{AssgnmntId}} | M1-Assignment | COMPLETED |
| SET2_100000110 | {{AssgnmntId}} | M1-Assignment | ENROUTE |
| SET2_100000111 | {{AssgnmntId}} | M1-Assignment | ONSITE |
| SET2_100000112 | {{AssgnmntId}} | M1-Assignment | COMPLETED |

## Adding the Email Capabilities to Flows

The test execution report can be sent to users as an email. To add email capabilities in a flow, add the component line mentioned in the table below towards the end in the flow.

| Usage Details | Value |
|---|---|
| Keyword | FUNCTIONCALL |
| Object | wSCOMMONLIB |
| Function Name | generateAndSendReport |

The email configuration properties file is in the server at:

```
install directory] /data/function-libraries/function-libs/OUTSP/
configuration.properties
```

Update the values mentioned below to configure the email:

```
#Email Details
gStrSMTP_HOST_NAME=internal-mail-router.oracle.com
gStrSMTP_PORT=25
gStrTO_EMAIL_RECIPIENTS=bhagwat.kolde@oracle.com
```

## Executing Test Flows

This section explains the steps to execute a test flow.

- Generating OpenScript Projects
- Creating Folder Structure for the Generated Scripts
- Configuring the Runtime Properties

### Generating OpenScript Projects

To generate an OpenScript project, follow these steps:

1. On the flow tree page, right-click a flow and select **Generate OFT Scripts**.
2. Enter the local folder details where the OpenScript project has to be generated.

The OpenScript project generated from the test flow has the following structure:

- **Master Driver** folder - This is an OpenScript project containing the script.java file which is the master driver script invoking all the scenario scripts in the scenario folder.
- **Scenario** folder - This is an OpenScript project. There will be one OpenScript project for each scenario defined in the flow. Each scenario's script will be invoked from the Master Driver script.
- **Components** - Components will be the OpenScript instructions called from the scenario scripts. Each component is enclosed within a step group in OpenScript.

### Creating Folder Structure for the Generated Scripts

After generating and downloading the OpenScript project, create the following folder structure to organize the generated test scripts. This folder structure needs to be created on every workstation where the automation scripts are executed.

- **ebs-function-libs**

  Stores the generic function library. Oracle Application Testing Suite for Oracle Utilities needs three generic function libraries, namely GENLIB, WEBLABELLIB, and WEBTABLEOBJ.

- **outsp-function-libs**

  Stores the Oracle Application Testing Suite for Oracle Utilities function libraries and product-specific function libraries. The default Oracle Flow Builder installer includes OUTSPCORELIB, WSCOMMONLIB, and WSVALIDATELIB libraries. These function libraries have to be stored in this folder, along with product-specific libraries and custom libraries.

- **genericJars**

  For executing the generated Web service automation code, the following third party jars are required. These jars need to be copied to the genericJars folder.

| Jar Name | Jar Version | Download URL |
|---|---|---|
| soa-model-core | 1.4.1.4 | http://www.membrane-soa.org/downloads/ |
| groovy | 2.0.4 | http://groovy.codehaus.org/Download |
| groovy-xml | | http://groovy.codehaus.org/Download |
| sjsxp | 1.0.2 | https://java.net/projects/sjsxp/downloads/ |
| asm | 4.0 | http://asm.ow2.org/download/index.html |
| httpclient | 4.2.2 | http://archive.apache.org/dist/httpcomponents/httpclient/ |
| httpcore | | http://archive.apache.org/dist/httpcomponents/httpcore/ |
| Jdom | 1.1.1 | http://www.jdom.org/dist/binary/archive/ |

- **XSD** - Stores the run time generated XSDs required for test execution summary email.
- **Etc** - Consists of the property file for providing runtime and email configuration details, such as test environment name, application user and password, email ID, etc.
- **Logs** - Stores the runtime generated test execution logs that can be later used for debugging.
- **Downloaded test suite zip** - Download and extract the test suite file from the location where all the above listed folders are placed.

## Configuring the Runtime Properties

The **configuration.properties** file is located in the **etc\** folder. It is used to store the runtime test execution parameters, such as application URL, application access information, email configuration, etc.

To use the email functionality for receiving the test execution report, provide the values mentioned below in the configuration.properties file (etc\configuration.properties file).

```
#Email Details
gStrSMTP_HOST_NAME=
gStrSMTP_PORT=
gStrTO_EMAIL_RECIPIENTS=
```

To provide the test environment details, provide the below values:

```
# Application URL pointing to test execution
gStrApplicationURL =
gStrApplicationXAIServerPath=
gStrEnvironmentName=
```

To provide the application user information for login, provide the values for below keys:

```
gStrApplicationUserName =
gStrApplicationUserPassword =
```

If the test suite has any database-side validations, provide the database details as below:

```
gStrApplicationDBConnectionString =
gStrApplicationDBUsername =
gStrApplicationDBPassword =
```

The path for the output files generated for reporting is as below:

```
# Output file details
gStrOutputFilePath =
gStrXSDFiles=
```

# Chapter 4

# Function Library Reference

This chapter lists the Oracle Application Testing Suite for Oracle Utilities function libraries and functions available to create components and flows in Oracle Flow Builder for testing Oracle Application Testing Suite for Oracle Utilities.

The chapter explains the following libraries:

- OUTSPCORELIB
- WSVALIDATELIB
- WSCOMMONLIB

## OUTSPCORELIB

Use the OUTSPCORELIB function library to develop the component code and flows for Web services and general applications. The library includes functions with date and time processing and string processing capabilities, as well as database and file operations.

This section provides a list of the functions included in the library, along with their usage details.

### runBatchFile

Executes a existing batch file.

Example:

```
runBatchFile("C://Test//Test.bat")

Input Parameters: String
Return Type: void
```

### killBatchFile

Kills the batch process in execution.

Example:

```
killBatchFile ("cmd.exe")

Input Parameters: String
Return Type: void
```

## getCurrentTimeInMilliSeconds

Gets the time in milliseconds.

Example:

```
getCurrentTimeInMilliSeconds ()

Input Parameters: <none>
Return Type: Sting
```

## rand

Gets the random number for the given range.

Example:

```
rand(int lo, int hi) ()

Input Parameters: lo, hi
Return Type: int
```

## randomStringWithGivenRange

Gets the random string for the given range.

Example:

```
randomStringWithGivenRange(int lo, int hi)

Input Parameters: lo, hi
Return Type: String
```

## Randomstring

Generates the random string based on the parameters passed. 'lo' and 'hi' are the lowest and highest numbers to be used to generate the random string.

Example:

```
randomstring (lowerLim, higherLim)

Input Parameters: int lowerLim , int higherLim
Return Type: String
```

## compare2Strings

Compares two strings and returns a boolean result based on the result of comparison.

> **Note**: This function returns "True" if strings provided are same. Else, it returns 'False'.

Example:

```
compare2Strings (String_A, String_B)

Input Parameters: String_A, String_B
Return Type: String
```

## randomNumberUsingDateTime

Gets the random string with date and time in it.

Example:

```
randomNumberUsingDateTime()

Input Parameters: <none>
Return Type: String
```

## getCurrentDateTimeWithGivenDateFormat

Gets the current date and time in the specified format.

Example:

```
getCurrentDateTimeWithGivenDateFormat(String dFormat)
getCurrentDateTimeWithGivenDateFormat("mm-dd-yyyy:hh.mm.ss")

Input Parameters: dFormat
Return Type: String
```

## getDateDiffInSecsWithGivenDateFormat()

Gets the difference in the date.

Example:

```
getDateDiffInSecsWithGivenDateFormat(String dateStart, String
dateStop, String dFormat)
getDateDiffInSecsWithGivenDateFormat("12-13-2014", "12-29-2014",
"mm-dd-yyy")

Input Parameters: String dateStart, String dateStop, String dFormat
Return Type: String
```

## getAdjustedTimeWithGivenDateTime

Gets the adjusted time with the given date and time.

Example:

```
getAdjustedTimeWithGivenDateTime(String dateTime, String offset,
String dFormat)
getAdjustedTimeWithGivenDateTime("12-13-2014", "-02:30","mm-dd-
yyyy")

Input Parameters: String dateTime, String offset, String dFormat
Return Type: String
```

## getAdjustedTimeWithCurrentDateTime

Returns the date and time after adding the specified offset to the current date and time in the specified date/time format. Date and time are the inputs to this function.

Example:

```
getAdjustedTimeWithCurrentDateTime(String offset, String dFormat)
getAdjustedTimeWithCurrentDateTime("-2.30", "mm-dd-yyyy")

Input Parameters: String dateTime, String offset, String dFormat
Return Type: String
```

## getAdjustedTimeWithGivenDateAndTime

Returns the date and time after adding the specified offset to specified date and time in the specified date/time format.

Example:

```
getAdjustedTimeWithGivenDateAndTime(String cuDate,String
cuTime,String offset, String dFormat)
getAdjustedTimeWithGivenDateAndTime("12-13-2014","12:15:00","-
2.30", "mm-dd-yyyy")

Input Parameters: String cuDate, String cuTime, String offset,
String dFormat
Return Type: String
```

### addDaysToCurrentDateWithGivenFormat

Adds the number of days to the current date and returns the result in the specified format.

Example:

```
addDaysToCurrentDateWithGivenFormat(String noOfDays, String
dFormat)
addDaysToCurrentDateWithGivenFormat("45", "mm-dd-yyyy")

Input Parameters: String noOfDays, String dFormat
Return Type: String
```

### serverDate

Gets the server date.

Example:

```
serverDate()

Input Parameters: <none>
Return Type: String
```

### executeSQLQry

Executes SQL and returns the record set.

Example:

```
executeSQLQry(String Query)
executeSQLQry("SELECT * FROM EMP")

Input Parameters: String Query
Return Type: Result Set
```

### executeSQLQryWithGivenDBDetails

Executes SQL and returns the record set.

Example:

```
executeSQLQryWithGivenDBDetails(String Query, String
ConnectionString ,String DBUsername,String DBPassword)
executeSQLQryWithGivenDBDetails("SELECT * FROM EMP", CONN_STR,
"system", "system00")

Input Parameters: String Query, String ConnectionString,String
DBUsername,String DBPassword
Return Type: Result Set
Exceptions: Database exception
```

### serverTime

Gets the server time.

Example:

```
serverTime()

Input Parameters: <none>
Return Type: String
```

## waitForTime

Waits for the specified time.

Example:

```
waitForTime(String strWaitTimeInMinutes)
waitForTime(“15”)

Input Parameters: String strWaitTimeInMinutes
Return Type: void
```

## verifyLastBatchRun

Verifies if the batch is in execution in the last x minutes.

Example:

```
verifyLastBatchRun(String Batch_CD, String strMaXTimeToCheck)
verifyLastBatchRun("1234567890", "90")

Input Parameters: String Batch_CD, String strMaXTimeToCheck
Return Type: String
```

## getCurrentOffsetTime

Gets the current offset time.

Example:

```
getCurrentOffsetTime(String cuDate, String cuTime, String
offset,String timeFormat)
getCurrentOffsetTime("12-13-2014", "12:30:00", "+2:30","mm-dd-
yyyy")

Input Parameters: String cuDate, String cuTime, String offset,
String timeFormat
Return Type: String
```

## addDaysToAGivenDate

Adds days to the provided date.

Example:

```
addDaysToAGivenDate(String date, String noOfDays)
addDaysToAGivenDate("12-13-2014", "19")

Input Parameters: String date, String noOfDays
Return Type: String
```

## randomNumber

Gets the random number.

Example:

```
randomNumber()

Input Parameters: <none>
Return Type: String
```

### createFile

Creates file in the specified path.

Example:

```
createFile(String FilePath)
createFile("C:\Logs.txt")

Input Parameters: String FilePath
Return Type: void
```

### getWaitConditionState

Waits for the specified time.

Example:

```
getWaitConditionState(long StartTime, float TimeInMinutes)
getWaitConditionState("12345L", "12.00")

Input Parameters: long StartTime, float TimeInMinutes
Return Type: boolean
```

### compare2Files

Compares two files.

Example:

```
compare2Files(String strFileName_A, String strFileName_B)
compare2Files("C:\Logs01", "C:\Logs04")

Input Parameters: strFileName_A, String strFileName_B
Return Type: String
```

### copyFile

Copies files from source to destination.

Example:

```
copyFile(String srcFilePath, String destFilePath)
copyFile("C:\temp.txt", "D:\temp.txt")

Input Parameters: strFileName_A, String strFileName_B
Return Type: void
```

### deleteFile

Deletes a file.

Example:

```
deleteFile(String filePath)
deleteFile("C:\temp.txt")

Input Parameters: String filePath
Return Type: void
```

## executeSQLQryUpdate

Executes SQL for the update query.

Example:

```
executeSQLQryUpdate(String Query, String ConnectionString,String
DBUsername,String DBPassword)
executeSQLQryUpdate("UPDATE EMP SET NAME="Oracle" where
EMPID='123'" CONN_STR, "system", "system00")

Input Parameters: String Query, String ConnectionString,String
DBUsername, String DBPassword
Return Type: String
```

## getDistinctObjects

Returns the count of distinct objects in a specific column in a table.

Example:

```
getDistinctObjects(String tableName, String columnName, String
condition, String ConnectionString ,String DBUsername,String
DBPassword)
getDistinctObjects ("EMP", "EMPID" CONN_STR, "system", "system00")

Input Parameters: String tableName, String columnName, String
condition, String ConnectionString,String DBUsername, String
DBPassword
Return Type: String
```

## setVariableValueUsingListIndex

Handles the resolving repeating elements in the response XML and retrieves the value(s) based on the parameters passed. The parameters passed are global variable (gVar1) and index value.

Example:

```
setVariableValueUsingListIndex(String listVariableName,String
index)
setVariableValueUsingListIndex(“data1,data2,data3”, 2)

Input Parameters: String listVariableName: List values separated by
comma
String index: the index number to retrieve value
Return Type: String: Value
```

## closeConnections

Closes the database connection opened for database verification.

Example:

```
closeConnection()

Input Parameters: <none>
Return Type: <none>
```

# WSVALIDATELIB

Use the WSVALIDATELIB function library to validate the test components (referred to as verification points) in the components. The library covers validation routines for string and XML elements in the returned response XML.

This section provides a list of functions in the library, along with the usage details.

### elementNotNull

Verifies if the specified element in response is null.

Example:

```
elementNotNull(String responseTag)
elementNotNull(mobileNumber)

Input Parameters: String responseTag
Return Type: void
```

### elementIsNull

Verifies if the specified element in response is not null.

Example:

```
elementIsNull (String responseTag)
elementIsNull (mobileNumber)

Input Parameters: String responseTag
Return Type: void
```

### elementValueEquals

Verifies if the specified element value in response is equal to the provided value.

```
elementValueEquals(String responseTag, String expectedValue)
elementValueEquals(mobileNumber, "1234567890")

Input Parameters: String responseTag, String expectedValue
Return Type: void
```

### elementValueNotEquals

Verifies if the specified element value in response is not equal to the provided value.

Example:

```
elementValueNotEquals(String responseTag, String expectedValue)
elementValueNotEquals (mobileNumber, "1234567890")

Input Parameters: String responseTag, String expectedValue
Return Type: void
```

### elementValueGreaterThan

Verifies if the specified element value in response is greater than the provided value.

Example:

```
elementValueGreaterThan(String responseTag, String valueToCompare)
elementValueGreaterThan("count","5")

Input Parameters: String responseTag, String valueToCompare
Return Type: void
```

## elementValueGreaterThanEqualTo

Verifies if the specified element value in response is greater than or equal to the provided value.

Example:

```
elementValueGreaterThanEqualTo(String responseTag,String
valueToCompare)
elementValueGreaterThanEqualTo("totalRecords", "50")

Input Parameters: String responseTag, String valueToCompare
Return Type: void
```

## elementValueLesserThan

Verifies if the specified element value in response is less than the provided value.

```
elementValueLesserThan(String responseTag,String valueToCompare)
elementValueLesserThan ("counter", "50")

Input Parameters: String responseTag, String valueToCompare
Return Type: void
```

## elementValueLesserThanEqualTo

Verifies if the specified element value in response is less than or equal to the provided value.

Example:

```
elementValueLesserThanEqualTo(String responseTag,String
valueToCompare)
elementValueLesserThanEqualTo ("attempts", "10")

Input Parameters: String responseTag, String valueToCompare
Return Type: void
```

## elementContains

Verifies if the specified element is available in the response.

Example:

```
elementContains(String responseTag,String valueToBeChecked)
elementContains("batchName", "F1-BILLING)

Input Parameters: String responseTag, String valueToCompare
Return Type: void
```

## elementNotContains

Verifies if the specified element is not available in the response.

Example:

```
elementNotContains(String responseTag, String valueToBeChecked)
elementNotContains ("description", "billing")

Input Parameters: String responseTag, String valueToCompare
Return Type: void
```

### reponseNotContains

Verifies if the specified value or element is not available in the response.

Example:

```
reponseNotContains(String value)
reponseNotContains("Failed")

Input Parameters: String responseTag, String valueToCompare
Return Type: void
```

### responseContains

Verifies if the specified value or element is available in the response.

```
responseContains(String value)
responseContains("Exception")

Input Parameters: String responseTag, String valueToCompare
Return Type: void
```

## WSCOMMONLIB

Use the WSCOMMONLIB function library to perform common operations in the Oracle Application Testing Suite for Oracle Utilities Web services testing, such as composing request, sending request, composing email summary, converting it to HTML format, sending an email, and parsing WSDL.

> **Note**: This library does not have any component development functions other than the generateAndSendReport function that provides result reporting and email capabilities to the user. See the Logging and Reporting section in Chapter 3 for more details.

This section provides the functions included in the library, along with their usage details.

### generateAndSendReport

Generates the HTML test execution report and sends the execution summary via email. The email settings can be specified in the configuration.properties file available in the /etc directory of the execution folder structure.

```
generateAndSendReport ()

Input Parameters: NA
Return Type: NA
```

# Appendix A

# Setting Up XAI Inbound Services

The Oracle Utilities application-specific components are developed using the Web services method, and these components need the XAI Inbound Services to be defined in the application.

This chapter includes the following sections:

- Creating XAI Inbound Services
- Importing XAI Inbound Services

## Creating XAI Inbound Services

To create an XAI Inbound Service, follow these steps:

1. Login to the Oracle Utilities application.
2. Click **Admin** > **X** > **+XAI Inbound Service**.
3. On the **XAI Inbound Service** page, do the following:
    a. Enter the **XAI Inbound Service Name**.
    b. Select **BusinessAdaptor** as the **Adapter**.
    c. Select **Business Object** as the **Schema Type**.
    d. Select the appropriate **Schema Name**.
    e. Enter the **Description** and **Detailed Description**.
    f. Select **Add** as the **Transaction Type**, and then click **Save**.

## Importing XAI Inbound Services

To import an XAI Inbound Service into the Oracle Utilities application, follow these steps:

**Note**: Ensure the exported XAI services are available in the local machine.

1. Login to the Oracle Utilities application.
2. Click **Admin** > **X** > **XAI Service Import**.
3. On the **XAI Service Import** page, click **Browse**.
4. Select the location of the file, and then click **Open**.
5. On the **XAI Service Import** page, click **Read**.
6. Select the **Import?** check box for the respective XAI Inbound Service Name.
7. Click **Save**. The "Imported Successfully" message appears in the **Message** text column.

# Appendix B

# License and Copyright Notices

This section provides license and copyright information for the associated products. It includes the following:

# Third-Party Products

The following sections provide notices and information about the third party products indicated.

## Notice Concerning Usage of Apache Software

The following files are covered under the Apache 2.0 license:

- groovy-2.0.4.jar
- groovy-xml-2.0.4.jar
- httpclient-4.2.2.jar
- httpclient-4.2.2.jar
- jdom-1.1.1.jar

### Apache License

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

## Notice Concerning Usage of sjsxp

sjsxp is covered under the COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) and the GNU General Public License (GPL) licenses.

**COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL) Version 1.0**

1. **Definitions**

   a. “Contributor” means each individual or entity that creates or contributes to the creation of Modifications.

   b. “Contributor Version” means the combination of the Original Software, prior Modifications used by a Contributor (if any), and the Modifications made by that particular Contributor.

   c. “Covered Software” means (a) the Original Software, or (b) Modifications, or (c) the combination of files containing Original Software with files containing Modifications, in each case including portions thereof.

   d. “Executable” means the Covered Software in any form other than Source Code.

   e. “Initial Developer” means the individual or entity that first makes Original Software available under this License.

   f. “Larger Work” means a work which combines Covered Software or portions thereof with code not governed by the terms of this License.

   g. “License” means this document.

   h. “Licensable” means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

   i. “Modifications” means the Source Code and Executable form of any of the following:

      a. Any file that results from an addition to, deletion from or modification of the contents of a file containing Original Software or previous Modifications;

b. Any new file that contains any part of the Original Software or previous Modification; or

c. Any new file that is contributed or otherwise made available under the terms of this License.

j. "Original Software" means the Source Code and Executable form of computer software code that is originally released under this License.

k. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

l. "Source Code" means (a) the common form of computer software code in which modifications are made and (b) associated documentation included in or with such code.

m. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

**2. License Grants**

**a. The Initial Developer Grant**

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, the Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license:

a. under intellectual property rights (other than patent or trademark) Licensable by Initial Developer, to use, reproduce, modify, display, perform, sublicense and distribute the Original Software (or portions thereof), with or without Modifications, and/or as part of a Larger Work; and

b. under Patent Claims infringed by the making, using or selling of Original Software, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Software (or portions thereof).

c. The licenses granted in Sections 2.1(a) and (b) are effective on the date Initial Developer first distributes or otherwise makes the Original Software available to a third party under the terms of this License.

d. Notwithstanding Section 2.1(b) above, no patent license is granted: (1) for code that You delete from the Original Software, or (2) for infringements caused by: (i) the modification of the Original Software, or (ii) the combination of the Original Software with other software or devices.

**b. Contributor Grant**

Conditioned upon Your compliance with Section 3.1 below and subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

a. under intellectual property rights (other than patent or trademark) Licensable by Contributor to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof), either on an unmodified basis, with other Modifications, as Covered Software and/or as part of a Larger Work; and

b. under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: (1) Modifications made by that Contributor (or portions thereof); and (2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

c. The licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first distributes or otherwise makes the Modifications available to a third party.

d. Notwithstanding Section 2.2(b) above, no patent license is granted: (1) for any code that Contributor has deleted from the Contributor Version; (2) for infringements caused by: (i) third party modifications of Contributor Version, or (ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or (3) under Patent Claims infringed by Covered Software in the absence of Modifications made by that Contributor.

**3. Distribution Obligations**

**a. Availability of Source Code**

Any Covered Software that You distribute or otherwise make available in Executable form must also be made available in Source Code form and that Source Code form must be distributed only under the terms of this License. You must include a copy of this License with every copy of the Source Code form of the Covered Software You distribute or otherwise make available. You must inform recipients of any such Covered Software in Executable form as to how they can obtain such Covered Software in Source Code form in a reasonable manner on or through a medium customarily used for software exchange.

**b. Modifications**

The Modifications that You create or to which You contribute are governed by the terms of this License. You represent that You believe Your Modifications are Your original creation(s) and/or You have sufficient rights to grant the rights conveyed by this License.

**c. Required Notices**

You must include a notice in each of Your Modifications that identifies You as the Contributor of the Modification. You may not remove or alter any copyright, patent or trademark notices contained within the Covered Software, or any notices of licensing or any descriptive text giving attribution to any Contributor or the Initial Developer.

**d. Application of Additional Terms**

You may not offer or impose any terms on any Covered Software in Source Code form that alters or restricts the applicable version of this License or the recipients' rights hereunder. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, you may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear that any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

**e. Distribution of Executable Versions**

You may distribute the Executable form of the Covered Software under the terms of this License or under the terms of a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable form does not attempt to limit or alter the recipient's rights in the Source Code form from the rights set forth in this License. If You distribute the Covered Software in Executable form under a different license, You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

f. **Larger Works**

You may create a Larger Work by combining Covered Software with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Software.

4. **Versions of the License**

a. **New Versions**

Sun Microsystems, Inc. is the initial license steward and may publish revised and/or new versions of this License from time to time. Each version will be given a distinguishing version number. Except as provided in Section 4.3, no one other than the license steward has the right to modify this License.

b. **Effect of New Versions**

You may always continue to use, distribute or otherwise make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. If the Initial Developer includes a notice in the Original Software prohibiting it from being distributed or otherwise made available under any subsequent version of the License, You must distribute and make the Covered Software available under the terms of the version of the License under which You originally received the Covered Software. Otherwise, You may also choose to use, distribute or otherwise make the Covered Software available under the terms of any subsequent version of the License published by the license steward.

c. **Modified Versions**

When You are an Initial Developer and You want to create a new license for Your Original Software, You may create and use a modified version of this License if You: (a) rename the license and remove any references to the name of the license steward (except to note that the license differs from this License); and (b) otherwise make it clear that the license contains terms which differ from this License.

5. **DISCLAIMER OF WARRANTY**

COVERED SOFTWARE IS PROVIDED UNDER THIS LICENSE ON AN “AS IS” BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED SOFTWARE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED SOFTWARE IS WITH YOU. SHOULD ANY COVERED SOFTWARE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED SOFTWARE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

6. **TERMINATION**

a. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License. You agree that You alone are responsible for compliance with the United States export administration regulations (and the export control laws and regulation of any other countries) when You use, distribute or otherwise make available any Covered Software.

**10. RESPONSIBILITY FOR CLAIMS**

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

NOTICE PURSUANT TO SECTION 9 OF THE COMMON DEVELOPMENT AND DISTRIBUTION LICENSE (CDDL)

The code released under the CDDL shall be governed by the laws of the State of California (excluding conflict-of-law provisions). Any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California and the state courts of the State of California, with venue lying in Santa Clara County, California.

**The GNU General Public License (GPL) Version 2, June 1991**

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

**Preamble**

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and

## Notice Concerning Usage of soa-model-core

## Notice Concerning Usage of ASM