Oracle Functional Testing Advanced Pack for Oracle Utilities

Reference Guide for Core

Release 5.0.1

E67844-03

November 2016

ORACLE

Oracle Functional Testing Advanced Pack for Oracle Utilities Reference Guide for Core, Release 5.0.1

E67844-03

Contents

Preface

This guide describes the Core components and the function libraries used to create those components for Oracle Functional Testing Advanced Pack for Oracle Utilities (OFTAPOU) v5.0.1. These components are used to build test flows in Oracle Flow Builder (OFB).

This preface includes the following sections:

Audience

This guide is intended for QA/Test Engineers and Automation Developers to understand the various components and libraries available for them to automate the business test flows for Core using Oracle Functional Testing Advanced Pack for Oracle Utilities (OFTAPOU) for Core.

Related Documents

For more information, see the following documents:

- *Oracle Functional Testing Advanced Pack for Oracle Utilities Release Notes*
- *Oracle Functional Testing Advanced Pack for Oracle Utilities Installation and Administration Guide*
- *Oracle Functional Testing Advanced Pack for Oracle Utilities User's Guide*

See also:

- Core Documentation Library

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

Component Reference

This chapter lists the Core starter components available to create flows in Oracle Flow Builder.

The chapter includes the following sections:

Overview

Oracle Functional Testing Advanced Pack for Oracle Utilities for Core is a test starter pack built on top of Oracle Functional Testing Advanced Pack for Oracle Utilities that generates test automation scripts using Oracle Flow Builder.

Oracle Functional Testing Advanced Pack for Oracle Utilities for Core contains out-of-the-box product-specific components used to build new test flows in Oracle Flow Builder to test the Core based applications. These out-of-the-box components correspond to specific business entities, such as business objects, service scripts, or business services used for interfacing with the application. Users can use these components as available or can extend them. Users can also create new components to be used to create flows. This starter pack also contains a set of function libraries that can be used for creating custom components.

> **Note:** See Chapter 2: Function Library Reference for detailed information about using these function libraries.

Consider this pack to be a starter kit which can be expanded and built upon. A few sample flows are included as an example.

> **Note:** See the *Oracle Functional Testing Advanced Pack for Oracle Utilities User's Guide* for information about creating components and flows.

The components are categorized under the following functional areas:

- Admin
- Batch
- ToDo
- AdminUI
- ToDo UI

Components

The following table lists the starter components available in Core.

Pre-requisites: The Inbound Web Service using the respective business object should be available in the application.

Additional Notes: Failure while creating, reading, or updating the component is logged in the test execution report, thus facilitating debugging/analysis of the problems.

Inbound Web Service Components

The following table lists out the Inbound Web Service based components.

Component	Functional Area	Description
F1-Algorithm	Admin	Used to create, read, update, and delete Algorithm via a Web service. After creation, the CRUD operations can be performed through these components against the F1-AlgorithmPhysicalBO business object.
F1-BatchControl	Admin	Used to create, read, or update a 'Batch Control' via a Web service. After creation, the CRUD operations can be performed through these components against the F1-BatchContolMO business object.
F1-Country	Admin	Used to create, read, or update a 'Country' via a Web service. After creation, the CRUD operations can be performed through these components against the F1-COUNTRY business object.
F1-DisplayProfile	Admin	Used to create, read, or update a 'DisplayProfile via a Web service. After creation, the CRUD operations can be performed through these components against the F1-DisplayProfilePhysicalBO business object.
F1-ExternalSystem	Admin	Used to create, read, or update an 'ExternalSystem' via a Web service. After creation, the CRUD operations can be performed through these components against the F1-ExternalSystemPhysicalBO business object.

Component	Functional Area	Description
F1-FeatureConfig	Admin	Used to create, read, or update a 'FeatureConfig' via a Web service. After creation, the CRUD operations can be performed through these components against the F1-FeatureConfigPhysicalBO business object.
F1-TimeZone	Admin	Used to create, read, or update a 'TimeZone' via a Web service. After creation, the CRUD operations can be performed through these components against the F1-TimeZonePhysicalBO business object.
F1-User	Admin	Used to create, read, or update a 'User' via a Web service. After creation, the CRUD operations can be performed through these components against the F1-UserPhysicalBO business object.
F1-UserGroup	Admin	Used to create, read, or update a 'UserGroup' via a Web service. After creation, the CRUD operations can be performed through these components against the F1-UserGroupPhysicalBO business object.
F1-WorkCalendar	Admin	Used to create, read, or update a 'WorkCalendar' via a Web service. After the creation, the CRUD operations can be performed through these components against the F1-WorkCalendarPhysicalBO business object.
F1-ToDoRole	ToDo	Used to create, read, or update a 'ToDoRole' via a Web service. After creation, the CRUD operations can be performed through these components against the F1-ToDoRolePhysical business object.

Component	Functional Area	Description
F1-ToDoType	ToDo	Used to create, read, or update a 'ToDoType' via a Web service. After creation, the CRUD operations can be performed through these components against the F1-ToDoTypePhysicalBO business object.
F1-BatchSubmission	Batch	Used to run or retrieve the Batch submission status via a Web service. After creation, execute the batch job and retrieve the batch job running status against the F1-BatchJob business object.
F1-ToDoEntryAdd	ToDo	Used to create a To Do Entry via a Web service. After creation, add To Do entries through these component against the F1-AddToDoEntry business service.
F1-ToDoEntryRead	ToDo	Used to retrieve a To Do Entry via a Web service. After creation, retrieve the To Do Entry through this component against the F1-MaintainToDoEntry business service.
F1-ToDoEntryUpdate	ToDo	Used to update a To Do Entry via a Web service. After creation, retrieve the To Do Entry through this component against the F1-UpdateToDoEntry business service.
F1-ToDoEntryComplete	ToDo	Used to complete a To Do Entry via a Web service. After creation, complete the To Do Entry through this component against the F1-CompleteToDoEntry business service.
F1-SendResultsMail	Admin	Parses the results and generates the email. This component sends the test executions results to the configured email ID.

Component	Functional Area	Description
F1-WaitTime	Admin	Used to wait for some time to complete certain task. The wait time is in specified in minutes.
F1-ImportAndDeployBundle	Admin	Reads the bundle content from the attachment and applies the bundle. This component returns the status and ID of the bundle objects in OUAF.
F1-InstallationOption	Admin	Enables edge applications to set the installation options values in OUAF.
F1-OutboundMessageType	Admin	Adds/reads the outbound message type.
F1-XAISender	Admin	Adds/reads the XAI sender.
F1-BatchCompletionStatus	Batch	Takes the Batch ID as input and verifies if a given batch is complete or not within the time frame.

UI Components

The following table lists out the UI based components. These components only work for the Oracle Utilities' products which are based on Oracle Utilities Application Framework (OUAF) v4.3 SP1.

Component	Functional Area	Description
F1-Login	Admin	Used to login into Core through UI.
F1-LoginSsl	Admin	Used to login into Core , ignoring the SSL certificate error.
F1-Logout	Admin	Used to logout from Core on UI.
F1-ToDoRoleAdd	ToDo	Used to create the ToDo Role from UI. After creating, this component can be used to create To Do types and used for CRUD operations.
F1-ToDoRoleRead	ToDo	Used to read the To Do Role from UI.
F1-ToDoRoleUpdate	ToDo	Used to update the To Do Role from UI.
F1-ToDoRoleDelete	ToDo	Used to delete the To Do Role object from Core application.

Component	Functional Area	Description
F1-ToDoTypeAdd	ToDo	Used to create the ToDo Type through UI in Core application.
F1-ToDoTypeRead	ToDo	Used to read the To Do Type existing in the application.
F1-ToDoTypeUpdate	ToDo	Used to update the To Do Type existing in the application.
F1-ToDoTypeDelete	ToDo	Used to delete the To Do Type existing in the application.
F1-ToDoEntryAdd	ToDo	Used to create an To Do Entry for the specific To Do Type.
F1-ToDoEntryRead	ToDo	Used to retrieve the To Do entry existing in Core, using the ToDo ID.
F1-ToDoEntryAssign	ToDo	Used to read the ToDo entry and assign the To Do Entry to user.
F1-ToDoEntryComplete	ToDo	Used to read the To Do Entry and complete it.

Chapter 2

Function Library Reference

This chapter lists the Core libraries and functions available to create components and flows in Oracle Flow Builder to test Core.

OUAFUILIB

The OUAFUILIB library comprises functions that work on the UI components. These functions are typically used to click the text area before setting values, and also **Alert** and **Confirm** windows.

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

confirmButton

Clicks **OK** on the **Confirm** window displayed while deleting an object.

Example:

```
confirmButton()

Input Parameters: NULL
Return Type: Boolean
```

alertButton

Identifies the **Alert** window and clicks **OK**. Use this function in negative test scenarios where an Alert window is displayed.

Example:

```
alertButton()

Input Parameters: NULL
Return Type: Boolean
```

toDoEntryAdd_textAreaClick

Clicks the text area called before setting the value in that text area. This function is used while creating a To Do Entry.

Example:

```
toDoEntryAdd_textAreaClick()

Input Parameters: NULL
Return Type: Boolean
```

toDoEntryAssign_textAreaClick()

Clicks the text area to be called before setting the value in that text Area. This function is used while assigning a To Do Entry.

Example:

```
toDoEntryAssign_textAreaClick()

Input Parameters: NULL
Return Type: Boolean
```

getToDoId()

Gets the To Do ID from the To Do Entry created. This To Do ID is used further to assign or complete the ToDo Entry.

Example:

```
getToDoId()

Input Parameters: NULL
Return Type: String
```

OUAFLIB

The OUAFLIB library comprises functions that work on the IWS components. These functions are generally used to get status from the database and read files from the attachment folder. Also, the library includes generic functions used for performing basic tasks.

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

checkBatchStatus()

Executes SQL query in database to retrieve the batch status for a batch ID. This function checks the batch status for a given max time.

Example:

```
checkBatchStatus()

Input Parameters:
   batchJobId - String - Batch job ID
   strMaXTimeToCheck - String - Max time to check the batch status

Return Parameter:
   result_stat - String - Return the batch status
```

waitConditionState()

Returns 'True' until the start time reaches the max time.

Example:

```
waitConditionState()

Input Parameters:
   StartTime - long - Start Time
   TimeInMinutes - float - Max time to check in minutes

Return parameters:
   Return Type: Boolean - Return true if start time is not reach to
    max time
```

checkFileExt()

Checks the file extension.

Example:

```
checkFileExt()

Input Parameters:
   filename - String - Input File name
   fileFormat - String - Desired file format

Return Parameters:
   Return Type: Boolean - Return true if input file name is in desired
   file format
```

readBundleFile()

Reads file from the Attachment folder and modifies the content as required to send the bundle import request. (Return string can be use as input for bundle Content.) This function supports only .xml or .txt file as the input bundle file.

Example:

```
readBundleFile()

Input Parameter:
   filename - String - Bundle file name

Return Parameter:
   strIWSBundleContents - String - Modified bundle content
```

checkBundleStatus()

Checks the bundle status from database.

Example:

```
checkBundleStatus()

Input Parameter:
   bundleId - String - Bundle ID

Return Parameter:
   result_stat - String - Returns the bundle status
```

compare2Strings()

Compares two strings. If the input strings are not equal, then component fails.

Example:

```
compare2Strings()

Input Parameter:
   String_A - String - Input String
   String_B - String - Input String

Return Parameter: NULL
```

readFromFile()

Reads content from file in the Attachment folder.

Example:

```
readFromFile()
```

```
Input Parameters:
   filename - String - Attachment File name

Return Parameter:
   Sb - String - Return file content
```

Chapter 3

Sample Work Flows

This chapter describes the Core sample flows that illustrate common use cases for Oracle Functional Testing Advanced Pack for Oracle Utilities. It also explains the procedure to execute these sample flows.

The chapter includes the following sections:

Sample Flows

The sample flows delivered as part of Oracle Functional Testing Advanced Pack for Oracle Utilities for Core demonstrate how flows can be created for Web services based testing and for a combination of Web services and UI based testing using the same framework.

These flows are designed to run using the demo data, giving the user the ability to deploy Oracle Functional Testing Advanced Pack for Oracle Utilities for Core and execute the sanity flows immediately. The flows perform a part of the basic sanity testing required to certify that the Core environment has been setup appropriately.

This section includes the following sample work flows for both UI and Web Services:

F1-ToDoFlow

F1-ToDoFlow is an Web service flow comprising the creation for To Do Role, To Do Type, To Do Entry, Assign To Do Entry, and Complete To Do Entry, and also the completion life cycle of an To Do.

The following table lists the tasks that are created and their respective components in Core.

Task	Core Component
To Do Role	F1-ToDoRole
To Do Type	F1-ToDoType
To Do Entry Add	F1-ToDoEntryAdd
To Do Entry Read	F1-ToDoEntryRead
To Do Entry Assign	F1-ToDoEntryUpdate
To Do Entry Complete	F1-ToDoEntryComplete
Send results mail	F1-SendResultsMail

The F1-ToDoFlow flow includes the complete ToDo flow:

1. Creates ToDoRole.
2. Creates ToDoType using the ToDoRole already created.
3. Adds ToDo Entry.
4. Reads the created ToDoEntry.
5. Updates the ToDoEntry entry status and verifies it.
6. Completes the ToDo Entry.
7. Sends the result email.

> **Note**: This flow works for Oracle Utilities Customer Care and Billing v2.5.0.1 and Oracle Utilities Work and Asset Management vv2.1.1.0, and is not re-runnable. To make it re-runnable, change the test data for F1-ToDoRole and F1-ToDoType before generating the OFT scripts.

ToDoRoleFlow

ToDoRoleFlow is a UI flow that includes the CRUD operations of ToDo Role.

The following table lists the tasks that are created and their respective components in Core.

Task	Core Component
Login	F1-LoginSsl
ToDo Role Add	F1-ToDoRoleAdd
ToDo Role Read	F1-ToDoRoleRead
ToDo Role Update	F1-ToDoRoleUpdate
ToDo Role Delete	F1-ToDoRoleDelete
Logout	F1-Logout

The ToDoRoleFlow flow includes the complete ToDo flow:

1. Logs in to the application.
2. Creates a ToDoRole.
3. Retrieves the existing ToDoRole.
4. Updates ToDoRole.
5. Deletes ToDoRole.
6. Logs out from the application.

ToDoFlow

ToDoFlow includes the creation of ToDo Role, ToDo Type, ToDo Entry, and also to complete the ToDo Entry.

The following table lists the tasks that are created and their respective components in Core.

Task	Core Component
Login	F1-LoginSsl
To Do Role Add	F1-ToDoRoleAdd
To Do Type Read	F1-ToDoTypeAdd
To Do Entry Add	F1-ToDoEntryAdd
To Do Entry Complete	F1-ToDoEntryComplete
F1-LoginSsl	F1-Logout

The ToDoRoleFlow flow includes the complete ToDo flow:

1. Logs in to the application.
2. Creates a ToDoRole.
3. Creates ToDoType.
4. Creates a ToDoEntry for the ToDoType already created.
5. Completes the ToDoEntry.
6. Logs out of the application.

Note: This flow is not re-runnable.

F1-BatchExecution

The F1-BatchExecution flow includes submitting the batch and getting back the status.

Note: Before generating the script, provide the test data (shown below) for the F1-BatchCompletionStatus component.

For WS-STARTPOLLWS keyword:

- Value1 is for total time to check the batch status.
- Value2 is for interval to check the batch status.

For WS-STOPPOLLWSIF keyword:

- Value1 for the condition to break the loop. (For example: batch status ED or PD)

The following table lists the tasks that are created and their respective components in Core.

Task	Core Component
Submitting the batch job	F1-BatchSubmission
Get the batch completion status	F1-BatchCompletionStatus
Send results e-mail	F1-SendResultsMail

F1-BundleImport

The F1-BundleImport flow includes reading the bundle content from the Attachment folder and applying the bundle. It returns the status and ID of the bundle objects in OUAF.

Note: Before generating the script, attach the file.

The following table lists the tasks that are created and their respective components in Core.

Task	Core Component
Read and apply the bundle content	F1-ImportAndDeployBundle
Send results e-mail	F1-SendResultsMail

F1-InitialInstallOption

The F1-InitialInstallOption flow enables the source applications to set installation option(s) values in OUAF.

The following table lists the tasks that are created and their respective components in Core.

Task	Core Component
Set installation option(s) values	F1-InstallationOption
Send results e-mail	F1-SendResultsMail

F1-OutboundMessageType

The F1-OutboundMessageType flow creates outbound message types.

The following table lists the tasks that are created and their respective components in Core.

Task	Core Component
Set values for outbound message type	F1-OutboundMessageType
Send results e-mail	F1-SendResultsMail

F1-XAISender

The F1-XAISender flow creates an XAI sender.

The following table lists the tasks that are created and their respective components in Core.

Task	Core Component
Set values for creating XAI sender	F1-XAISender

Task	Core Component
Send results e-mail	F1-SendResultsMail

Executing Sample Flows

This section describes the procedure to setup sample flows and execute them.

Pre-requisites

To execute a sample flow, ensure the following pre-requisites are met:

- Core is up and running.
- OpenScript is installed in the local machine. See the *Oracle Functional Testing Advanced Pack for Oracle Utilities Installation and Administration Guide* for the version details.
- Oracle Functional Testing Advanced Pack for Oracle Utilities is installed and repository/ directory is setup in the local machine appropriately. See the *Oracle Functional Testing Advanced Pack for Oracle Utilities Installation and Administration Guide* for more details.

Setting Up Inbound Web Service Sample Flows

To setup an Inbound Web Service based sample flow, follow these steps:

1. Login to Core.
2. Import the Inbound Web services into the Core where the scenarios need to be executed.

   See the **Importing Inbound Web Services** section in the *Oracle Functional Testing Advanced Pack for Oracle Utilities User's Guide* for steps to import the Inbound Web services.
3. Navigate to **Admin > B > Bundle Import > Add**.
4. Enter the **External Reference**, **Detailed Description**, and **Bundle Details** from the IWS Bundle Export Dump.
5. Click **Save,** and then click **Apply bundle**.
6. Launch OpenScript in the local machine and perform the following steps:
   a. Navigate to **View > OpenScript Preferences**.
   b. In the left tree, select **OpenScript**. In the sub tree, select **Playback**, and then select **Error Recovery**.
   c. Click **SetAll** and select **Report Error and Continue**.
   d. Click **Apply**, and then click **Close**.
7. Configure the **configuration.properties** file as follows:
   a. Provide the application URL for the parameter:

```
gStrApplicationURL = http\://<%serverName%>\:<%portNumber%>/
ouaf
```

b. Provide the additional path required for Inbound Web service URL:

```
gStrApplicationXAIServerPath=/<%webservices/
gStrApplicationURL%>/<%AppendThisToAbove gStrApplicationURL%>/
```

c. Provide an environment name for display in the results email:

```
gStrEnvironmentName= <%testEnvironmentName%>
```

d. Provide the application login user ID:

```
gStrApplicationUserName= <%UserNAme%>
```

e. Provide the application login password:

```
gStrApplicationUserPassword= <%password%>
```

f. Provide the SMTP email server and e-mail ID:

```
gStrSMTP_HOST_NAME=<%SMTP ServerName%>
gStrSMTP_PORT=<%PortNumber%>
gStrTO_EMAIL_RECIPIENTS=<%e-mail Id%>
```

g. Provide the application database details as below:

```
gStrApplicationDBConnectionString =<%jdbc Connectionstring%>
eg: jdbc\:oracle\:thin\:@<%DBserverName%>\:<%port%>\:<%DBSID%>
gStrApplicationDBUsername=<%DBUserID%>
gStrApplicationDBPassword=<%DBPassword%>
```

h. Provide the full directory path of Oracle Application Testing Suite repository directories in the local machine.

```
gStrOutputFilePath=<%LogFilePath%>
Example: C:\\CORE_DEMO\\OUTSP\\Logs\\
gStrXSDFiles=<%XSD Folder path%>
Example: C:\\CORE_DEMO\\OUTSP\\Logs\\
```

9. Create a folder (Core) in the outsp-function-libs folder in the Oracle Functional Testing Advanced Pack for Oracle Utilities repository directory. Copy the function libraries to the respective folders.
   - Core
     - OUTSPCORELIB
     - WSCOMMONLIB
     - WSVALIDATELIB
     - OUAFLIB
10. Copy all the .jar files provided in the installer into the **genericJars** folder in the Oracle Functional Testing Advanced Pack for Oracle Utilities repository directory.

Setting Up UI Sample Flows

To setup a UI based sample flow, follow these steps:

1. Create a folder 'Core' in the outsp-function-libs folder in the Oracle Functional Testing Advanced Pack for Oracle Utilities repository directory. Copy the function libraries to the respective folders.
   - Core
     - OUAFUILIB

2. Copy all the .jar files provided in the installer into the **genericJars** folder in the Oracle Functional Testing Advanced Pack for Oracle Utilities repository directory.
3. Ensure the **ebs-function-libs** folder exists in the Oracle Functional Testing Advanced Pack for Oracle Utilities repository directory, and then copy all the latest function libraries.

   Note: To execute the UI based sample flows, provide the Application URL, User Name, and Password in the F1-LoginSsl component test data before downloading/generating the Oracle Functional Tester script.

Appendix A

Inbound Web Services

The Core components are developed using Web services method, and these components require Inbound Web Services to be defined in the application.

For instructions to create, import, or search an Inbound Web Service, see the **Setting Up Inbound Web Services** appendix in *Oracle Functional Testing Advanced Pack for Oracle Utilities User's Guide*.

> **Note**: Starting Oracle Utilities Application Framework, Release 4.3 SP2, the username token policy is not applicable any more.

List of Inbound Web Services

The list of Inbound Web Services provided to use with the delivered components and flows is as follows:

- ATF1DisplayProfile
- ATF1Algorithm
- ATF1ExternalSystem
- ATF1FeatureConfig
- ATF1BatchContol
- ATF1User
- ATF1WorkCalendar
- ATF1UserGroup
- ATF1TimeZone
- ATF1ToDoRole
- ATF1ToDoType
- ATF1ToDoEntryADD
- ATF1ToDoEntryREAD
- ATF1ToDoEntryUPDATE
- ATF1ToDoEntryComplete
- ATF1Country
- ATF1BatchSubmission
- ATF1InitialInstallOption

- ATF1BundleImportApply
- ATF1OutboundMessageType
- ATF1XAISenderPhysicalBO