Oracle Utilities Testing Accelerator

Reference Guide for Core

Release 20A

F32730-01

June 2020

ORACLE

Oracle Utilities Testing Accelerator Reference Guide for Core, Release 20A

F32730-01

Contents

Preface

Welcome to the Oracle Utilities Testing Accelerator Reference Guide for Core.

This guide describes the Core components and the function libraries used to create those components for Oracle Utilities Testing Accelerator. These components are used to build test flows in Oracle Utilities Testing Accelerator Workbench.

The 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 Utilities Testing Accelerator (OUTA) for Core.

Related Documents

For more information, refer to the following Oracle resources.

Release Notes

For Release Notes refer to the Oracle Cloud Readiness portal at https://www.oracle.com/webfolder/technetwork/tutorials/tutorial/readiness/offering.html?offering=customer-20

User and Reference Guides

- *Oracle Utilities Testing Accelerator User's Guide for Cloud*
- *Oracle Utilities Testing Accelerator Reference Guide for Oracle Utilities Customer Cloud Service*
- *Oracle Utilities Testing Accelerator Reference Guide for Oracle Utilities Meter Services Cloud Solution*
- *Oracle Utilities Testing Accelerator Reference Guide for Oracle Utilities Work and Asset Management Cloud Service*

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.

Abbreviations

The following terms are used in this document:

Term	Expanded Form
OUTA	Oracle Utilities Testing Accelerator
OUAF	Oracle Utilities Application Framework

Chapter 1

Component Reference

This chapter lists the Core starter components available to create flows in Oracle Utilities Testing Accelerator, including:

- Overview
- Components

Overview

Oracle Utilities Testing Accelerator for Core is a test starter pack built on top of Oracle Utilities Testing Accelerator that generates test automation scripts using Oracle Utilities Testing Accelerator Workbench.

Oracle Utilities Testing Accelerator for Core contains out-of-the-box product-specific components used to build new test flows in Oracle Utilities Testing Accelerator Workbench 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.

For more information about using these function libraries, refer to Chapter 2: Function Library Reference.

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

For more information about creating components and flows see *Oracle Utilities Testing Accelerator User's Guide for Cloud*.

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.

Component	Functional Area	Description
F1-BatchContol	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.
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.

Component	Functional Area	Description
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.
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.

Component	Functional Area	Description
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.
F1-OutboundMsgType	Admin	Component for Outbound Message type.
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.

Component	Functional Area	Description
F1-FileStorage	Admin	Used for file storage configuration.
F1-GeographicType	Admin	Used to add Geographic Type.
F1-BatchCompletionStatus	Batch	Takes the Batch ID as input and verifies if a given batch is complete or not within the time frame.
F1-BatchRTStatistics	Batch	Used to get the batch run and thread statistics using Batch Job ID.

UI Components

The following table lists out the UI based components. These components only work for the Oracle Utilities' products which are built on Oracle Utilities Application Framework (OUAF) v4.3 SP5.

Component	Functional Area	Description
F1-LoginToCloud	Admin	Used to log into cloud environment through UI
F1-Quit	Admin	Quit Browser
F1-XAISubmission	Admin	Used for XAI submission through UI
F1-VerifyAppViewer	Admin	Used to verify App Viewer
F1-VerifyHelp	Admin	Used to verify Help
F1-CompleteToDoEntry	ToDo	Used to create the complete ToDo entry from UI.
F1-ForwardToUser	ToDo	Used to forward to user from UI.
F1-ReopenToDoEntry	ToDo	Used to reopen a ToDo entry from UI.
F1-SendBackToUser	ToDo	Used to send back a ToDo entry to user.

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.

- OUAFLIB
- OUAFCoreLIB
- UTILITIESLIB
- ToDoLIB
- WebUtilLib
- GeneralLib

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
```

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 Parameter:
   filename - String - Attachment File name

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

OUAFCoreLIB

The OUAFCoreLIB library comprises functions that work on the UI components.

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

navigatePageThroughMenu()

Navigates to the respective page through Admin Menu.

```
Parameters:
@param sTopMenu - String - Top Menu (Admin/Menu)
@param sMenu - String - Menu Item (for example: U/To Do)
@param sSubMenu - String - Sub Menu (for example: UI Map/To Do Entry).
If there is no sub menu, then mention it as null.
@param add_search - String - Add/Search. If not, mention it as null.
```

loginToCloud()

Logs into the cloud environment.

verifyResult()

Matches the result with actual result and expected result. If it matches, then log it 'passed'; else as 'failed'.

```
Parameters:

@param actualResult - String - Actual Result

@param expectedResult - String - Expected Result
```

switchToFrame()

Switches to a frame. This function handles up to 3 frames (topMenu, tabPage, tabMenu) in Oracle Utilities Application Framework. It does not handle the other under laying frame.

```
Parameters:
@param frameId - String - Frame Id
```

switchToWindow()

Switches to a window. This function closes the window if there is more than one window open.

windowClose()

Closes the open window.

ouafDDLOperations()

Performs DDL operations such as save, delete, clear, or refresh.

```
Parameters:
@param operationName - String - Operation name
```

invokeMethodWithReturn()

Dynamically invokes a function and returns the value.

```
Parameters:
@param aClass - String - Class Name
@param aMethod - String - Method name
@param params - String - No of parameter for invoking class.
@param args - String - comma separated parameter list
```

enterSchemaText()

Enters the schema text.

```
Parameters:
@param schemaText - String - Schema Text
@param schemaFrame - String - Frame name for Schema Editor
```

verifyStringContains()

Verifies the contents of the string.

```
Parameters:
@param firstString - String - Actual String
@param sunString - String - Verify String
```

getTableData()

Gets data from the table.

```
Parameters:
@param idTable - String - Table ID
@param row - String - Row number
@param col- String - Column number
```

actionOnURL()

Helps in navigating to a new URL.

```
Parameters:
@param actionToPerform - String - Action to be perform on the URL (eg:
flushAll, debug etc)
```

enterText()

Enter text in the text area through javascriptExecutor.

```
Parameters:
@param locatorType - String - Locator Type
@param locator - String - Locator (ex: id,xpath)
@param text - String - Text which need to enter in text area
```

UTILITIESLIB

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

generateRandomString()

Used to generate random strings.

readFromAttachmentFile()

Reads files from the Attachment folder. Pass the parameter as "null" to ignore this function.

```
Parameters:
fileName - String - File name (Ex: UIMapSchema.xml)
```

ToDoLIB

forwardToDoEntry()

Used to give values in the window opened after clicking the Forward button in ToDo entry.

WebUtilLib

getAttribute()

Returns the value of a particular attribute.

Example: To get the value of the "value" attribute for batch name field: getAttribute("id","BATCH_CD","value")

```
Parameters:
@param locatorType - String - Locator Type (Example: id,xpath etc)
@param webElement - Web element locator path (Example: Batch_CD)
@param attributeName - String - Name of the attribute (Example: value)
@return - String - return the value of the attribute
```

getText()

Returns the text value for an element.

```
Parameters:
@param locatorType - String - Locator Type (Example: id,xpath)
```

```
@param locatorPath - String - Locator path
@return - textValue - String - Text value of element
```

acceptPopUp()

Accepts any pop-up windows coming on the page.

closeBrowser()

Closes the browser.

GeneralLib

verifyAppViewerDialog()

Verifies the Application Viewer page UI.

verifyHelpDialog()

Verifies help dialog.

```
Parameters:
@param helpKeyword - String - Help Keyword
```

verifyXAIResponse ()

Verify if the expected string exists in the XAI.

```
Parameters:
@param xaiResponse  - String - XAI Response
@param verificationStrings - String - Comma separated verification
string
```

Chapter 3

Sample Work Flows

This chapter describes the Core sample flows that illustrate common use cases for Oracle Utilities Testing Accelerator. It also explains the procedure to execute these sample flows. It includes the following sections:

Sample Flows

The sample flows delivered as part of Oracle Utilities Testing Accelerator 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 Utilities Testing Accelerator 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-BatchExecution
- F1-BundleImport
- F1-InitialInstallOption
- F1-OutboundMessageType
- F1-XAISender

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.

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.

Make sure to attach the file before generating the script.

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
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.
- Download UTA.exe to the local machine.

  See *Oracle Utilities Testing Accelerator Installation and Administration Guide* for the version details.
- Oracle Utilities Testing Accelerator is installed and repository/directory is setup in the local machine appropriately. See *Oracle Utilities Testing Accelerator 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 Utilities Testing Accelerator 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. Configure the **configuration.properties** file as follows:

   a. Provide the application URL for the parameter:

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

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

   Example: C:\\CORE_DEMO\\OUTSP\\Logs\\

   gStrApplicationUIURL = UI URL of the cloud
   ```

   Note: Use OUAF501_IWS_4305x.xml for web services import.

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 Utilities Testing Accelerator User's Guide for Cloud*.

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

The appendix includes the following:

Inbound Web Services

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

- ATF1Algorithm
- ATF1BatchContol
- ATF1BatchSubmission
- ATF1BundleImport
- ATF1BundleImportApply
- ATF1CheckUserAuthorization
- ATF1Country
- ATF1DateMath
- ATF1DisplayProfile
- ATF1EmailService
- ATF1ExternalSystem
- ATF1FeatureConfig
- ATF1GeoService
- ATF1GetAppSrvAccMds
- ATF1InitialInstallOption
- ATF1InsertGTTRecords
- ATF1OutboundMessageType
- ATF1PhoneNumberValidation
- ATF1PhoneType
- ATF1PhoneTypePhysicalBO
- ATF1ShiftDateTime
- ATF1TimeZone
- ATF1ToDoEntryADD
- ATF1ToDoEntryComplete
- ATF1ToDoEntryREAD
- ATF1ToDoEntryUPDATE
- ATF1ToDoRole
- ATF1ToDoType
- ATF1User
- ATF1UserGroup
- ATF1WorkCalendar
- ATF1XAISenderPhysicalBO
- ATZZBATCHRTS
- ATF1OutboundMsgType
- ATF1GeographicType

Modified Components

The components that are modified in this release are:

- F1-Country
- F1-ExternalSystem

Deprecated Components

The components deprecated in this Oracle Utilities Testing Accelerator release are:

- F1-ToDoRoleAdd
- F1-ToDoRoleRead
- F1-ToDoRoleUpdate
- F1-ToDoRoleDelete
- F1-ToDoTypeAdd
- F1-ToDoTypeRead
- F1-ToDoTypeUpdate
- F1-ToDoTypeDelete
- F1-ToDoEntryAdd
- F1-ToDoEntryRead
- F1-ToDoEntryAssign
- F1-ToDoEntryComplete

Deprecated Libraries

The OUAFUILIB library is deprecated in this Oracle Utilities Testing Accelerator release.

Deprecated Flows

The flows that are deprecated in this Oracle Utilities Testing Accelerator release are:

- ToDoRoleFlow
- ToDoFlow