Application Testing Suite

OpenScript Functional Testing Introduction

Yutaka Takatsu
Group Product Manager
Oracle Enterprise Manager - ATS
Agenda

• Application Testing Suite (ATS) & OpenScript Overview (10 min)
• Starting with OpenScript Functional Testing (30 min)
  ✓ Recording your Web Transaction with OpenScript
  ✓ Reading the Script Commands
  ✓ Playing back the recorded Script
  ✓ Customizing the Script

• OpenScript Product Demo (15 min)
• OpenScript Features – Current and Future Releases (5 min)
  ✓ OpenScript FT 12.1 - New Features and Updates
  ✓ Coming Features and Current Limitation in OpenScript 12.1
• *ADF Functional Testing Introduction*

• Bug Submission and Problem Reporting
Oracle Application Testing Suite
A powerful, integrated testing solution for ensuring application quality, performance and reliability

Oracle Functional Testing
(OpenScript)
Automated functional & regression testing
Also Generates Load Testing Scripts

Oracle Test Manager
Test process management, requirements and defect tracking

Oracle Load Testing
Automated load & performance testing
Oracle Functional Testing - OpenScript

- Automates functional & regression testing for Web, SOA and packaged applications
- Also a load testing script generator
- Custom Accelerators for Oracle Packaged Applications & Web technologies
- GUI-based visual scripting interface
- Flexible scripting extensibility using Java
- Integrated Eclipse IDE for script development & debugging
  - Most of the case, Advanced knowledge in Java/Eclipse is not necessary
- Integrated with Oracle Load Testing & Oracle Test Manager

GUI View: Standard - Easy
Java Code View: Technical - Extensible
Eclipse View: Advanced - Optional
Optimized testing for Oracle Applications

Application Testing Suite Testing Accelerators

Oracle Fusion Applications
Oracle Ebusiness Suite
Siebel

JD Edwards Enterprise ONE
PeopleSoft
Hyperion

Accelerators
Web
Siebel
EBS/FORMS
Fusion/ADF
JDEdwards Enterprise One
Web Services
Database
Adobe® Flex®

OpenScript Framework (API)

Eclipse Platform
Agenda

• Application Testing Suite (ATS) & OpenScript Overview

• **Starting with OpenScript Functional Testing - Basic Operations**
  ✓ Recording your Web Transaction with OpenScript
  ✓ Reading the Script Commands
  ✓ Playing back the recorded Script
  ✓ Customizing the Script

• OpenScript Product Demo

• OpenScript Features – Current and Future Releases
  ✓ OpenScript FT 12.1 - New Features and Updates
  ✓ Coming Features and Current Limitation in OpenScript 12.1

• ADF Functional Testing Introduction

• Bug Submission and Problem Reporting
Recording a script with OpenScript

- OpenScript records Web application that opens from the browser.
- Super Basic Steps to Record a Script
  - Create a Script
  - Hit the Record button to Start the recording
  - Navigate Through the Browser
  - Press the Stop button to Stop the Recording
- Script is recorded.
What are Recorded in the Script

- GUI Commands
- Contents Returned from the Server

Why is this important? You can later use the information to debug your script, by comparing it with the playback results.
OpenScript Browser Helper Service

- OpenScript requires Oracle ATS Helper service up and running in your system, in order to record your web transaction from the browser.

Points at: <ATS install>\openScript\HelperService\bin\wrapper.exe

- Services installed by ATS Setup
  - Oracle ATS Agent: Remote Agent Service used for Load testing
  - **Oracle ATS Helper: OpenScript Helper Service.**
  - Oracle ATS Server: Oracle Load Testing & Oracle Test Manager Console. (Weblogic server)
  - OracleServiceXE: Oracle XE Database used by OLT and OTM
  - OracleXETNSListener: Listener Service used for the above OLT/OTM Database
OpenScript adds Browser Add-ons to capture your Web Transactions. Internet Explorer: OpenScript Toolbar and BHO

- OpenScript adds 2 Add-ons to IE.
  - **OpenScript Tool Bar**
  - **OpenScript BHO.**

- OpenScript Toolbar is enabled in the IE Browser
- If NOT enabled, OpenScript will fail to start the recording.

If you see “Cannot get a connection from the BHO” error at the recording, check whether the toolbar is enabled.
OpenScript adds Browser Add-ons to capture your Web Transactions. FireFox: OpenScript WebDOM extension

- OpenScript WebDom Extension is added to the FireFox browser

- OpenScript WebDOM is enabled in the FF Browser
- If NOT enabled, OpenScript will fail to start the recording.

..And you will see an error message like this
Supported Browsers for OpenScript Functional Testing in 12.1

- OpenScript 12.1 FT Supports:
  - Internet Explorer 9 *
  - FireFox 6 *

- OpenScript 12.1 FT Does NOT Support:
  - FireFox 7 and above
  - Any other browsers (Chrome, Safari, Opera etc)

- Also the followings are not supported:
  - Stand-alone, Windows Application
  - Mobile Simulators (iOS, Android emulator etc)
  - Any app that does not start from the browser

FireFox FT Browser Support
- FIREFOX 10 ESR (Fix By 12.2)
- FIREFOX 9, (To Review)
- FIREFOX 8, (To Review)
- FIREFOX 7, (To Review)
- FIREFOX 6, (Implemented in 12.1)
- FIREFOX 5, (Implemented in 12.1)

*IE 9 & FF6 browser support is new feature in 12.1
Agenda

• Application Testing Suite (ATS) & OpenScript Overview

• **Starting with OpenScript Functional Testing**
  ✓ Recording your Web Transaction with OpenScript
  ✓ **Reading the Script Commands**
  ✓ Playing back the recorded Script
  ✓ Customizing the Script

• OpenScript Product Demo

• OpenScript Features – Current and Future Releases
  ✓ OpenScript FT 12.1 - New Features and Updates
  ✓ Coming Features and Current Limitation in OpenScript 12.1

• ADF Functional Testing Introduction

• Bug Submission and Problem Reporting
The user actions are captured at the recording, and they are written down into the script in the form of: `<XPATH>. Action` commands.

Xpath is the logic how OpenScript identifies the object.

At the Playback, OpenScript executes them **ONE BY ONE** until it gets to the end of the Script.
How do I read Xpath in a command?

```
web.textBox(9,
  "//web:window[@title='Single Sign On - Login']" +
  "//web:document[@index='0']" +
  "//web:form[@name='LoginForm']" +
  "//web:input_text[@name='ssousername' or @index='0']")
  .setText("yutaka.takatsu@oracle.com")
```

When executing this command, OpenScript will:

- Find a **window** with an attribute `title="Single Sign On - Login"`  
- Within the window, find a **document** `index='0'`,  
- Within the document, find a **form** that has an attribute `name='LoginForm'`  
- Within the form, find a **object** "Input" type "text",  
- That has an attribute either `name = 'ssousername'` or `index='0'`  
- Enter text "yutaka.takatsu@oracle.com" into the text box object
Define your Object Identification rule in the Preference

- Object identification refers to how objects are located within your application.
- Objects are identified by one or more attributes in the Xpath syntax
  - (e.g. Link object can be identified by attributes: text, href and Index)
- Object attributes are recorded based on Object Identification preferences (customizable!)
- Separate rules for each technology module (Web, Forms, Siebel, ADF)

Link object will be recorded with attributes: text, href or index in its Xpath.

ADFCommandButton object will be recorded with attributes: text and absoluteLocator in its Xpath.

Defines What Attribute(s) you want to use to identify Which Object
Agenda

• Application Testing Suite (ATS) & OpenScript Overview

• Starting with OpenScript Functional Testing
  ✓ Recording your Web Transaction with OpenScript
  ✓ Reading the Script Commands
  ✓ Playing back the recorded Script
  ✓ Customizing the Script

• OpenScript Product Demo

• OpenScript Features – Current and Future Releases
  ✓ OpenScript FT 12.1 - New Features and Updates
  ✓ Coming Features and Current Limitation in OpenScript 12.1

• ADF Functional Testing Introduction

• Bug Submission and Problem Reporting
Playback OpenScript Script

- Basic Steps to Playback a Script
  - Hit the Playback button to Start the Playback
  - Browser opens and plays back the script
  - Once the Playback is finished, You see:
    - Playback results (duration, status, what data is used, error information)
    - Session Report in HTML
Observe the Playback Results

- Results View tells you the details of the script playback.
- Displays Duration, Status and Data used of each command.
- Shows error details in case of script failure.
Compare the Recorded and Playback Data

- Select WaifForPage node in the Result view
- Compare the results between the Recorded and Playback (Browser, HTML, Screenshot)
- Allows you to narrow down to the cause of the problem, in case of the script failure
Agenda

• Application Testing Suite (ATS) & OpenScript Overview

• **Starting with OpenScript Functional Testing**
  ✓ Recording your Web Transaction with OpenScript
  ✓ Reading the Script Commands
  ✓ Playing back the recorded Script
  ✓ **Customizing the Script – Databanks, Object Libraries, Scripts, and Functions**

• OpenScript Product Demo

• OpenScript Features – Current and Future Releases
  ✓ OpenScript FT 12.1 - New Features and Updates
  ✓ Coming Features and Current Limitation in OpenScript 12.1

• ADF Functional Testing Introduction

• Bug Submission and Problem Reporting
Working with Script Assets: Databanks

- Parameterization allows users to take a single script and run it back with multiple data inputs.
- OpenScript provides Databank Wizard to Add Databank to the script, as a script Asset.
- Use Databank to substitute script inputs with variables which are associated to external data files.
  - Example: Login script that runs with different username/password combinations.

1. Select a node you want to parameterize.
2. Click the Substitute Variable Icon.
3. Select an external file you have a source of data.
4. Select the column of the data you want to use in the script.
5. Data is Parameterized!
Working with Script Assets: Object Libraries

- Object paths are specified in the script code with XPath notation that can make the script lengthy, and difficult to read.
- Optionally, object paths can be stored in an Object Library File that associates object paths to simple names.
- Allows users to store object paths used in multiple scripts in a single location (and make updates to multiple scripts at once).

**WITHOUT Object Library: XPath in the script**

```
web.button(9, 
"/web:window[@index='0' or @title='Stocks']
/web:document[@index='0']
/web:window[@index='0']
/web:form[@id='loginform' or @name='loginform' or @index='0']
/web:input_submit[@name='LoginButton' or @value='Login' or @index='0']")
.click()
```

**WITH Object Library: Simple description**

```
web.button(9, 
"{{obj.myOBJlib.web_input_submit_LoginButton}}")
.click()
```

**XPath Saved in the external file**
Working with Script Assets: Object Libraries

• Steps to Configure an Object Library
  1. Create a script. BEFORE recording, select “Apply Object Libraries” from the Script menu.
  2. Add a new Object Library or select an existing object library file (.properties).
  3. Record your script. Object IDs are stored in the Object Library.

• You can also apply an object library to existing scripts with hard-coded object paths and replace them with the simple names.

Alternatively, directly add Object Library in the Script Properties dialog.
Working with Script Assets: Child Scripts

- Production Test Flows can have many steps.
- Implementing a long flow into a single script can make the script large in size, and complex.
- Lengthy scripts are hard to maintain, difficult to debug.
- Users can modularize the test flow into multiple mid-size flows, implement them into multiple child scripts and run from a Master Driver script.
Working with Script Assets: Child Scripts

- Steps to Add Child Scripts into a Master driver script

1. Create a Master Script. Right Mouse Click to select “Add”
2. Select “RunScript” and Click OK
3. Select Child Scripts. By doing this, Child scripts are added as Script assets to the Master Script
4. Child Scripts are added to the Master Script as assets.
5. Optionally, you can dynamically load & run a child script at runtime, WITHOUT adding it as an asset.

*dynamic loading of the assets is a new feature in 12.1
Working with Script Assets: User Defined Functions

- Common transactions or repeated code can be defined as a function.
- Functions can be stored **locally in the same script** or in a **dedicated Function Library** script, and share among the multiple scripts.

Steps to Create a Local Function

1. Right Mouse Click to select “Add”
2. Select “New Function” and Click OK
3. Define function name, and arguments
4. Function is created

```
MyLocalFunction(String MyVar_Str)
```
Working with Script Assets: User Defined Functions

- Steps to Call a Local Function

1. Right Mouse Click to select “Add”
2. Select a local function to call and Click OK
3. Command to call the local Function is created
How to Create Dedicated Function Library in 12.1

- Functions can be placed in a dedicated function library script, and share among the multiple scripts
- Below are the Steps to Create a Dedicated Function Library script in 12.1

1. Check “Create Script as a Function Library” in Script Wizard
2. Specify a Unique Package name and a Class name
3. Function Library files are created in the file systems
How to Convert 9.x Function Library to 12.1 Format

- Steps to convert a Function Library in 9.x format to a new 12.1 format.

In order to enable Code Assist*, function Libraries created in 9.x need to be converted to the 12.1 format.

Open a Function Library Script Created in 9.x. Select “Convert to Function Library” from the Menu.

Specify a Unique Package name and a Class name.

Function Library files are created in the file systems.

This is also the procedure to convert a script created in 12.1, if you did not check “Create Script as a Function Library” check box at the script creation.

*Code assist for calling a function from a dedicated Function library is a new feature in 12.1
How to Call a Function from a Function Library

• In the calling script, add Function Library as a script asset

Once Function Library script is created, write functions and save the Library Script.

In the calling script, open “Script properties”, add Function Library script as a script asset

Function names show up in both Tree View, and Code view*

// Syntax before 12.1 (it is still supported in 12.1):
getScript("myLibrary").callFunction("login", "username", "pw");

// New 12.1 Syntax:
myLibrary.login("username", "pw");

Note the syntax change, for calling function Library (although both are supported in 12.1)

* Code assist for calling a function from a dedicated Function library is a new feature in 12.1
Agenda

• Application Testing Suite (ATS) & OpenScript Overview
• Starting with OpenScript Functional Testing
  ✓ Recording your Web Transaction with OpenScript
  ✓ Reading the Script Commands
  ✓ Playing back the recorded Script
  ✓ Customizing the Script

• OpenScript Product Demo
• OpenScript Features – Current and Future Releases
  ✓ OpenScript FT 12.1 - New Features and Updates
  ✓ Coming Features and Current Limitation in OpenScript 12.1
• ADF Functional Testing Introduction

• Bug Submission and Problem Reporting
OpenScript Demo: Covered Topics

• OpenScript UI
  • Three Windows (Script, Result, and Details)
  • Tree View (standard), Java Code view (extensible), Eclipse UI (advanced)
• Script Recording
  • What are recorded?
    • GUI Commands (Script Window)
    • Contents returned from the server (Details Window)
  • Recording Mechanism: Helper Service, Browser Add-ons BHO (IE and FF), Toolbar, and other ATS services
• Object Identification Preferences
• Script Playback
  • Result pane, Session Report, Details pane (comparison view),
• Working with Script assets: DataBank, Object Library, Script and Functions
Agenda

• Application Testing Suite (ATS) & OpenScript Overview

• Starting with OpenScript Functional Testing
  ✓ Recording your Web Transaction with OpenScript
  ✓ Reading the Script Commands
  ✓ Playing back the recorded Script
  ✓ Customizing the Script

• OpenScript Product Demo

• OpenScript Features – Current and Future Releases
  ✓ OpenScript FT 12.1 - New Features and Updates
  ✓ Coming Features and Current Limitation in OpenScript 12.1

• ADF Functional Testing Introduction

• Bug Submission and Problem Reporting
OpenScript FT 12.1 - New Features and Updates

OpenScript Functional & Load Testing

- OpenScript Dynamic Loading of Databanks
- OpenScript Dynamic Loading of Script and Functions
- OpenScript Enumerated List Function Arguments
- OpenScript UI support for List and Map Function Arguments
- OpenScript Function Libraries with Code Assist
  - How to Create Function Library in 12.1
  - How to Convert 9.x Function Library
  - How to Call Function library From a Script
- OpenScript IE 9 Enhancements
- OpenScript Firefox 4 Enhancements (support up to FireFox ver.6)
- OpenScript Shortcut Key Preferences
- OpenScript Preferences Dialog enhancement

