Open2Test Test Automation Framework Extensibility for Developers – TestPartner

Version 1.0

July 2009

Table of Contents

1. Purpose of the Document

The purpose of this document is to share guidelines for customizing Open Source Test Automation Framework code. This document will help with adding or modifying functions or keywords in the framework code.

1.1. Scope

The scope of this document is to provide guidelines for customizing Open Source Test Automation Framework code.

1.2. Overview

This document provides guidelines for:

- Adding new functions
- Adding new keywords
- Modifying functions
- Modifying keywords
- Adding new objects
- Adding new actions

2. Adding New Functions/Adding New Keywords

2.1. Adding New Functions

The Open Source Test Automation Framework's easily extensible features include custom functions. New functions can be added into the framework in two ways.

2.1.1. Include the function in the framework and call the function from the framework code

To add new custom functions to the framework follow the steps below:

- Refer to coding standards for naming the function.
- Define the function and declare the arguments that are passed to the function.
- Assign value to the function if it has to return the value.
- Call the function in the framework code.

2.1.2. Include the function in the framework and call the function from the keyword script

To add new functions and call those from keyword scripts follow the steps below:

- Refer to coding standards for naming the function.
- Define the function and declare the arguments that are passed to the function.
- Assign value to the function if it has to return the value.
- Call the function from the keyword script.

Syntax:

Call function	FunctionName:<input arguments>	Variable<holds return value>

2.2. Adding New Keywords

New keywords can be added to the framework by following these steps:

- Design the keyword.
- Use ";" as a delimiter in the third column.
- Use ":" as a delimiter in the fourth column.
- Include the second column cell value as a case in the main (keyword()) function.
- Call any existing function or a new function in the case statement.

- Use the values of the third and fourth columns to pass parameters to the function and to handle the value that is returned by the function (depending on the function definition).

Example:

To add a keyword for comparing two strings:

1. Design the keyword syntax.

StrCompare	<String1>;<String2>	Variable<holds return value>

2. Add the case statement “strcompare” to the select case in the function keyword_dotnet and call the function Func_StringOperations inside the case statement.
3. Add the case statement “strcompare” to the select case in the function Func_StringOperations.
4. Inside the case statement “strcompare”, write code for comparing the two input strings.
5. Write the code logic in the case statement “strcompare”.
6. The return value should be true if the two strings match and otherwise should be false.

3. Modifying Functions and Keywords

3.1. Modifying Functions

Refer to the Open Source Test Automation Framework description document for a list of functions in the framework code. These functions can be modified or customized by following the below steps:

- Identify the function that needs to be modified.
- Identify where this functions is called throughout the framework code.
- Modify the function.
- Based on the modifications, update the framework code where this function is called.

3.2. Modifying Keywords

Refer to the Open Source Test Automation Framework Keyword Naming conventions document for a list of keywords that are available in the Open Source Test Automation Framework. The keyword functionality can be modified or customized by following the below steps:

- Identify the keywords that need to be modified.
- Identify where these keywords are used in the framework code and modify them accordingly.

4. Adding New Objects and Actions

4.1. Adding New Objects

Refer to the Open Source Test Automation Framework Keyword Naming Conventions document for a list of objects that are handled in the framework code. When you encounter new or custom objects that are not handled in framework you can include them in the framework code by following the below steps:

- Identify the object class and name it based on object naming conventions.
- Identify the list of actions that are performed on the object.
- Identify the list of check points that are needed.
- Include code for setting context on the object.
- Include code for performing actions on the object.
- Include code for performing checks on the object.

Example:

Adding an object: “DotNetObject”

1. Add a case statement “dotobject” to the select case in the function Func_ObjectSet.
2. Inside the case statement, write the code “Set object = DotNetObject ((curObjClassName), tpAttachNoWait)”. This code will ensure that the object type is set.

4.2. Adding New Actions

When you encounter new actions that are not handled in the framework you can add them into the framework code by following the below steps:

- Identify the action and name it.
- Identify the list of objects where this action should be performed.
- Include code for performing this action for all the required objects.

Example:

Adding a new action: “Rightclick”

1. Add a case statement “rightclick” in the select case of function Func_Perform.
2. In the case statement, mention the code “object.rightclick”. This will perform the required action on the required object.

5. Use of Wildcards

TestPartner supports the use of wildcards in objects. If the object attach names are changing dynamically, then to identify these kinds of objects we can use wildcard or regular expression in combination with the attach name properties.

Example:

Attach Name for .NET Form = “Form1 DotNETForm”

Properties for .NET Form: Caption=Form1 Name=Form1

If attach Name changed to “Form2 DotNETForm”, or “Form3 DotNETForm” in runtime, then we can identify the same using the following properties with wildcards.

Caption=Form*

Caption=?orm*

Name=Form?

Name=For?*

Line	A	B	C	D	Remarks
10	r	Perform	form;Caption=Form*	attach	Attach form
12	r	Perform	form;Caption=Form*	close	Close form

6. General Guidelines

- Follow coding standards for defining and naming functions.
- Refer to the Open Source Test Automation Framework Keyword Naming Conventions document for including new objects or new actions in the framework code.
- Before adding or modifying functions, add necessary comments such as date of modification, modified by or created by, as well as what was modified or added.
- Use but don't change global variables within the function.
- Use local variables as much as possible.
- Use pass by value and pass by variable for passing values and variables.
- Add new keywords or functions only when existing functions are not able to perform the desired action.