



Open2Test Test Automation Framework for SilkTest – Coding Standards for Developers

Version 1.0

January 2010

DISCLAIMER

Verbatim copying and distribution of this entire article is permitted worldwide, without royalty, in any medium, provided this notice is preserved.

TABLE OF CONTENTS

1. PURPOSE OF THE DOCUMENT	3
1.1. Scope	3
1.2. Overview	3
2. NAMING STANDARDS	4
2.1. Naming Standards for Variables	4
2.2. Naming Standards for Constants	4
3. FUNCTION/PROCEDURE	5
3.1. Function Name	5
3.2. Function Header	5
3.3. Function Complexity	5
3.4. Function Structure	6
4. COMMENT STANDARDS	7
4.1. Framework Code Header Comments	7
4.2. Line Comments	7
5. GENERAL GUIDELINES	9

1. Purpose of the Document

The purpose of this document is to describe the standards to be followed when designing and developing the framework code. This document will help ensure consistency across the code, resulting in increased usability and maintainability of the developed code.

1.1. Scope

The scope of this document is to provide standards for designing and developing Open Source Test Automation Framework code for various technologies.

1.2. Overview

This document provides guidelines for:

- Naming standards
- Functions and procedures
- Comment standards
- General guidelines

2. Naming Standards

2.1. Naming Standards for Variables

Data Type	Prefix	Length	Example
Boolean	bln	Up to 20 characters	blnFlag
Integer	int	Up to 20 characters	intCount
Object	obj	Up to 20 characters	objCurrent
String	str	Up to 20 characters	strCurPage
Array	arr	Up to 20 characters	arrCellData
User-defined Type	dt	Up to 20 characters	dtTransaction
AnyType	Anytype	Up to 20 characters	AnytypeObject(50)

2.2. Naming Standards for Constants

- The constant names should be initial capped with underscores between words, as shown in the following example.

Example: gstrApplication_Path

3. Function/Procedure

3.1. Function Name

The function name should start with 'Func_' followed by the name of the function.

When arguments are passed to the function, the variable naming standards should indicate whether the arguments are passed by value or passed by reference.

Example: ANYTYPE Func_Context (ARRAY[50] OF Anytype Object, ARRAY[50] OF Anytype ActionValueOne)

3.2. Function Header

The function or procedure header should contain the following:

- Name of the function/procedure
- Description of the function/procedure
- List of input parameters with their description
- Return value of the function with its description (Not applicable for procedures)
- Date of creation
- Name of the person modifying it
- Date of modification

Example:

```
*****
'Function Name:           Func_StrSearch
'Description:            This function is used to search for a
substring in the main string.
'Input parameters:      strMainString - Main string that should be
searched'
strSubString - Substring that should be searched for
'Return Value: True - substring present in the main string'
False - substring not present in the main string
'Date of Creation:      15th November 2009
'Modified By:          Tester
'Date of Modification:  17th November 2009
*****
```

3.3. Function Complexity

Framework code should be designed and developed with minimal possible loops and conditions to reduce complexity and enhance maintainability.

3.4. Function Structure

The following tips provide guidance for creating easy-to-read and easy-to-maintain code.

- Modularize the code for increased reusability and reduced redundancy.
- Code should be well indented with tabs. (Tab width should be 4.)
- Values passed and returned to the functions should use simple variables.
- Reduce the use of global variables within the function. The scope of the variable should be decided based on the standards.

4. Comment Standards

4.1. Framework Code Header Comments

The framework code header should contain the following:

- The copyright and proprietary information
- Name of the framework code
- Author of the code
- Name of the reviewer
- Date of creation
- Version number

Every change to the framework code should be documented in the modification history. A modification history should contain the following:

- Name of the person who changed the code
- Date of change
- Version
- Changed function/event
- Change description

Example:

```
'#####  
#####  
'Copyright Information:  
'Project Name:      Java Framework  
'Author:           Open2Test  
'Version:          V1.0  
'Date of Creation:  17th November 2009  
'#####'  
Modification History  
'Modified By:  
'Reviewed By:  
'Modified Functions/Events:  
'Modification Description:  
'Modification Date:  
'#####'
```

4.2. Line Comments

Significant lines in the code should be provided with inline comments to better explain the line of code's purpose and make it easier for

subsequent developers to understand the code faster and more thoroughly.

Example:

//If the item does not exist, perform first set of actions else perform second set of actions.

```
If item is null then
    // First set of actions
    Statement 1
    Statement 2
    .
    .
    Statement n
Else
    // Second set of actions
    Statement a
    Statement b
    .
    .
    Statement n
```

5. General Guidelines

- Use dynamic arrays.
- Declare only one variable in a line.
- There should not be more than 80 characters per line.
- The code should be properly indented.
- Declare variables using appropriate data types.
- Use procedures instead of functions if there is no return value.

COPYRIGHT

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.