# Selenium Open Source Test Automation Framework Coding Standards for Developers

**Version 1.0**

**September 2009**

# Table of Contents

# 1. Purpose of the Document

The purpose of this document is to describe standards to be followed when designing and developing 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 tools and 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 | Example |
|---|---|---|
| Boolean | bln | blnFlag |
| Integer | int | intCount |
| Long | lng | lngRowNumber |
| Double | dbl | dblWeight |
| Object | obj | objCurrent |
| Single | sng | sngPosition |
| String | str | strCurPage |
| Array | arr | arrCellData |
| Variant | vnt | vntPropValue |
| User-Defined Type | udt | udtTransaction |

| Scope | Prefix | Example |
|---|---|---|
| Global | g | gStrAppPath |
| Module-Level | m | mintRowCount |
| Static | s | svntFlag |
| Variable Passed by Reference | r | rintValue |
| Variables Passed by Values | v | vintValue |
| Local to the Function | None | strCurPage |

## 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 'def' followed by the name of the function.

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

**Example:** def GetObject (strObjectname)

## 3.2. Function Header

The function or procedure header should contain the following:

- Name of the project
- Name of the function
- Name of the author
- Description of the function/procedure
- Date of creation
- List of input parameters with their description
- Name of the person modifying it
- Date of modification

**Example:**

'**********

#Project Name : Innovez - Selenium Framework

#Function Name : importdata.rb()

#Author : Keane India

#Description : This function handles the Suite Run, Object Repository, Keyword Driver and creating Reports.

#Date of creation : 1-Jan-09

#Input Parameters:

#Name of person modifying: Tester

#Date of modification: 11-Sep-09

'**********

## 3.3. Function Complexity

Framework code should be designed and developed with minimal possible loops and conditions for reduced complexity and enhanced 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:**

```
#Project Name  : Web Framework
#Author  : Innovez
#Version : V1.0
#Reviewed By: Manager 1
#Date of Creation    : 19-Dec-09
#Modification History:
#Date of change: 13-Sep-09
#Version: V1.1
#changed function: def func1
#change description:
#Modified By   : Tester 1
```

## 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:**

```
#Open the Object Repository Excel sheet
   @@ObjBook = @@excel.workbooks.open(@@Object_Rep)
```

# 5. General Guidelines

- Use an ampersand (&) for concatenating strings instead of '+'.
- Set the objects to nothing for cleaning the memory.
- 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.
- Avoid using the variant data type.
- Use procedures instead of functions if there is no return value.