# **Test Automation Framework Coding Standards**

Table of Contents

[**Test Automation Framework Coding Standards** 1](#_Toc500433297)

[**1 Purpose of the Document** 4](#_Toc500433298)

[**1.1 Scope** 4](#_Toc500433299)

[**1.2 Overview** 4](#_Toc500433300)

[**2 Naming Standards** 5](#_Toc500433301)

[**2.1 Naming Standards for Variables** 5](#_Toc500433302)

[**2.2 Naming Standards for Constants** 5](#_Toc500433303)

[**3 Function/Procedure** 6](#_Toc500433304)

[**3.1 Function Name** 6](#_Toc500433305)

[**3.2 Function Header** 6](#_Toc500433306)

[**3.3 Function Complexity** 6](#_Toc500433307)

[**3.4 Function Structure** 7](#_Toc500433308)

[**4 Comment Standards** 8](#_Toc500433309)

[**4.1 Framework Code Header Comments** 8](#_Toc500433310)

[**4.2 Line Comments** 9](#_Toc500433311)

[**5. General Guidelines** 10](#_Toc500433312)

[**6 Java Coding Naming Guidelines & Best Practices for Error-Free Code** 11](#_Toc500433313)

[**6.1 General Concepts in Naming** 11](#_Toc500433314)

[**6.2 Article Naming Convention** 11](#_Toc500433315)

[**6.2.1 Arguments or Parameters** 11](#_Toc500433316)

[**6.2.2 Fields & Variables** 11](#_Toc500433317)

[**6.2.3 Constants** 11](#_Toc500433318)

[**6.2.4 Classes And Interface** 11](#_Toc500433319)

[**6.2.5 Compilation Unit Files** 11](#_Toc500433320)

[**6.2.6 Component** 11](#_Toc500433321)

[**6.2.7 Packages** 12](#_Toc500433322)

[**6.2.8 Member Function** 12](#_Toc500433323)

[**7 Source Code Style Guidelines** 13](#_Toc500433324)

[**7.1 Line Spacing** 13](#_Toc500433325)

[**7.2 Blank Space** 13](#_Toc500433326)

# **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

Below are some of the key parameters that a software tester needs to keep in mind, while developing a test automation framework.

* Handle scripts and data separately
* Create libraries
* Follow coding standards
* Offer high extensibility
* Less maintenance
* Script/Framework version control

**Start Small**

• Don’t rush to automate every test case

• Identify test case priority then automate

• Don’t automate end-to-end

• Divide and conquer

**Avoid Dependencies**

• Don’t allow tests to be dependent on each other

• One test’s actions should not drive another tests assertion criteria

**Don’t Forget to Wait!**

• Pages don’t load instantaneously, and not all elements load at the same time

• A lot of your failures in finding an element will disappear if you use these

• Use Set Timeout functions for asynchronous calls.

**Collect Metrics**

• How long did test runs take before and after automation?

• How many bugs do automated tests identify per release?

• How many engineers does a test run require?

**Objects Naming**

• Objects should be easily identified by their logical name

• Consider naming of Ok button existing on Login Page

• pgPageName\_objectTypeElementName E.g. pgLogin\_btnOk

* Use object classes as prefix like chk, edt, lst, btn, lnk, txt, img,

**Locators order**

• CSS Selector

• ID

• Name

• Class Name

• Tag Name

• Link Text

• Partial Link Text

• XPath

## **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 as shown in the following example.

**Example**: gstrApplicationPath , gstrProjectPath

## **3 Function/Procedure**

## **3.1 Function Name**

The function name should start with ‘func’ followed by the name of the function.

**Example**: funcVerifyPageTitle (strPageName)

## **3.2 Function Header**

The function or procedure header should contain the following:

• 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**:

‘\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Function Name : funcLogout()

#Author : xyz India

#Description : This function logs out from the application

#Date of creation : 1-Jan-18

#Input Parameters:

#Name of person modifying: Tester

#Date of modification: 11-Jan-18

‘\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## **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 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**: #Import the Object Map using require

Var objectMap = require(‘../ObjectMap/AndroidOR.js’)

## **5. General Guidelines**

* Destroy the objects if created 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.

## **6 Coding Naming Guidelines & Best Practices for Error-Free Code**

It’s one of the coding guidelines which depends on the context you are in. Let’s read more about this.

## **6.1 General Concepts in Naming**

1- Follow domain related naming strategy.  
2- Use sentence case to make names readable.  
3- Be reluctant while using abbreviations.  
4- Prevent using redundant names that differ only in case.

## **6.2 Article Naming Convention**

## **6.2.1 Arguments or Parameters**

Use a related name for the value/object being passed, and prefixing with **<arg> or <param>**.

**Example** argEmpName, paramSalary etc.

## 

## **6.2.2 Fields & Variables**

Start field/variable name in lower case and then continue in sentence case.

Example  viewInfo, openWindow.

Don’t use underscores to start or separate the words.

## **6.2.3 Constants**

Use upper case and underscores to form constants.

**Example** static final int MAX\_SIZE = 256;

static final string BROWSER\_TYPE = “Chrome”;

## **6.2.8 Member Function**

Have a function name that relates to the task it meant for. Start it with an active verb whenever possible.

**Some good naming practices.**

**Good method names.**

funcShowStatus(), funcDrawCircle(), funcAddLayoutComponent().

**Bad method names.**

menuButton() – noun phrase; doesn’t describe function.

OpenTab() – starts with upper-case letter.

click\_first\_menu() – uses underscores.

**Boolean getter member functions.**

It’s a good practice to prefix boolean getter functions with <is>.

**Example** isVisible(), isChecked(), isNumeric().

**Getter member functions.**

Usually all getter functions should start with <get> prefix.

**Example** getLocalDate(), getMonth(), getDayOfMonth().

**Setter member functions.**

Usually all setter functions should start with <set> prefix.

**Example** setLocalDate(), setMonth(), setDayOfMonth().

## **7 Source Code Style Guidelines**

## **7.1 Line Spacing**

**1)** Limit each line under 80 characters.  
**2)** Limit comment length up to 70 characters.  
**3)** Keep tab sizes equal to 4 spaces.