|  |  |
| --- | --- |
|  |  |
|  |  |

**Automation Scripting Guidelines**

**Cardinal Health Automation**

# Revision History

| **Revision Number** | **Revision Date** | **Editor Name** | **Edit Description** |
| --- | --- | --- | --- |
| 1.0 | 6/20/2014 | Rajib Saha | Initial draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Revision History 2](#_Toc388335753)

[Executive Summary 4](#_Toc388335754)

[1. Purpose Of This document 4](#_Toc388335755)

[2. Scripting Guidelines 4](#_Toc388335756)

[2.1 Variable Declaration 4](#_Toc388335757)

[2.1.1 Local scope variables 4](#_Toc388335758)

[2.1.2 Global scope variables 5](#_Toc388335759)

[2.2 Formatting 5](#_Toc388335760)

[2.3 Business Component 5](#_Toc388335761)

[2.4 Object Repository 5](#_Toc388335762)

[2.5 Function Libraries 5](#_Toc388335763)

[2.6 Synchronization 6](#_Toc388335764)

[2.7 Error handling 6](#_Toc388335765)

[2.8 Reporting 6](#_Toc388335766)

[3. Best Practices 6](#_Toc388335767)

# Executive Summary

# Purpose Of This document

Cardinal Automation Team will follow the scripting guidelines while creating Automation test scripts, Business Components and function libraries. A combination of these scripting techniques will ensure that the best practices are leveraged upon during the automation.

This document details the QTP Scripting Guidelines and Naming Conventions. All the test automation members should follow the defined scripting standards and guidelines. This document will be very helpful to all the new members joined in automation as well as the existing members working in automation while writing scripts.

# Scripting Guidelines

# Variable Declaration

* “Option Explicit” is declared before using any variable in the component. May be used in function libraries but will not work unless all function libraries have option explicit included.
* Add general comments to describe groups of variables usage in the Components/Function.
* ByRef / ByVal declarations should be required in function / sub declarations.
* Variable names are descriptive to aid in understanding the code. For example, the first three letters of a variable indicates the type of the variable. Use of the "\_" (underscore) character should be avoided and instead capital letters should be used.

# Local scope variables

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Prefix** | **Example** |
| Boolean | bln | blnChecked |
| Date | d | dCurrentDate |
| Variant | v | vTblValue |
| Integer | int | intRow |
| Object | o | oTestExcelObj |
| String | str | strUserID |
| Arrays | a | Used in another prefix, as in astrWindow(i) |

# 

# Global scope variables

The values of global variables can be used and changed all over the project within all scripts and libraries.

Syntax:"g" + [Prefix] + [ShortDescription]

Letter “g” indicates that the scope of the variable is global. [Prefix] is a lowercase letter that represents the type of the global variable. The rules for [Prefix] are the same as for “Local scope variables”.

Examples:

gintRow, gstrUserID

# Formatting

* Properly indented code using tabs.
* Sufficient comments are indicated to support any deviation from the work flow

# Business Component

* The format of the business component is attached below:



# Object Repository

* Object Repository should be shared. Objects should be added in OR by using only those properties that are not changeable.
* Mandatory and assistive properties should be given to each object for proper identification.
* Smart Identification should be turned off while recording or learning the objects.
* Ordinal identifier should be set as “None” – don’t use index, location or creation time until these are heavily required.
* Naming Convention – Use meaningful names. Preferably, use the name of the object given in the application for the same object while stored in OR.
* Regular Expressions can be used where applicable.
* Objects under unique pages – each page should have its own page object.

# Function Libraries

* Any piece of code which may be reused in multiple components should be moved into function library. This also includes sub procedures.
* All functions would have a header with the name of the function, purpose, author, date of creation along with change history.
* The header portion would also contain any input, output values being passed.

# Synchronization

* Any use of a wait statement must be avoided except where other synchronization methods don’t work as hard-coded **Wait** increases the execution time.

# Error handling

* Proper error handling should be enforced throughout the scripts for handling recoverable errors for the next row
* The test results need to be clear for anyone to interpret of what step failed and the cause of failure.

# Reporting

* When combining test cases into a single automation script, reporting should indicate pass/fail status for each of the manual test cases.
* Report the beginning and end of each test case if automation script is covering for more than one test case.

# Best Practices

Below is the list of some best practices which should be kept in mind during scripting:

* Ensure that the required Quick Test Professional add-ins have been installed properly.
* Do not use hard-coded paths, Values etc.,
* Avoid using hard-coded *wait ()* statements.
* Try to limit the length of the component / Function library.
* Make your components / functions are readable, maintainable and easily understandable
* While creating looping structure, keep performance in mind.
* Keep as much as possible code into Functions not in components.
* Try to write user-defined functions independent of module.
* Report the PASS / FAIL status of each section of the scripts.
* During verification, if the expected results do not match with actual results then log both the expected and actual results into the results.
* Robust Scripts - Make sure the test script does not fail during execution.
* Use QTP editor while creating functions instead of notepad.
* Follow the naming conventions as per the standard.
* Proper comments should be added for every step explaining its need in the module.
* Review the scripts with others.
* Double click the function library (.vbs) file and ensure whether the function library file is compiled without throwing any error.