Unisys

Automation Scripting guideline

Doc No: Unisys/00/00

Ver. 1.0

**Statement of Ownership**

This document contains information that is proprietary to Unisys, The document must, therefore, not be copied, reproduced in any manner or distributed in part or in its entirety without the explicit written permission of the owner.

**For Controlled Circulation Only**

This document must only be made available to persons who have executed Unisys standard non-disclosure, non-compete agreement.

**Template Control**

**Change History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version**  **No.** | **Details of Change** | **Changed Sections** | **Prepared by** | **Reviewed/**  **Approved by** | **Date** |
| 1.0 | Draft |  |  |  | 26/04/2017 |

**Template Review and Approval**

|  |  |  |
| --- | --- | --- |
| **Reviewer / Approver** | **Name** | **Title and Department** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Reference Documents**

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Description** | **Version No.** |
|  |  |  |
|  |  |  |
|  |  |  |

# Objective

This document explains the conventions to be followed while automating the test scripts. TAT (Test Automation Team) of Unisys is primarily responsible to follow this guideline.

# Guidelines

### Guidelines for indenting the Scripts

Indentation should be one **TAB (6spaces)** space with every condition statements, every function, loops etc….

**For Example:**

If <condition>

{

………………

Statement….;

……………….

}

### Business Method Naming Conventions

The conventions are mixed-case format.

Multiple words were separated using uppercase letters.

Method does not return any value.

**public void userLogin**

{

**Statement 1;**

**………………….**

**Statement n;**

**}**

Method returns boolean

**private boolean totalAlerts**

{

**Statement 1;**

**………………….**

**Statement n;**

**}**

### Basic Ideas for Scripting Standards

• Dynamic waits should be used instead of hard waits or pause commands waiting for the application to reach active state

• When using a new script units, ensure that you add the all the unit references required.

### Logging

* Avoid usage of System.out.Println console statements for debug purpose. Instead start using log4j which gives the flexibility (both properties file and xml file) to show or not console statements depending on which environment we are planning to execute the scripts.
* Include logger lines at critical and strategic points in the code, such as:
  + Entry into critical methods
  + Exit from methods that are returning critical data
  + Before and after operations that are important or critical
  + Around database operations

## Exception Handling

We as programmers want to write quality code that solves problems. Unfortunately, exceptions come as side effects of our code. No one likes side effects, so we soon find our own ways to get around them.

public void consumeAndForgetAllExceptions(){

try {

...some code that throws exceptions

} catch (Exception ex){

ex.printStacktrace();

}

}

What is wrong with the code above?

Once an exception is thrown, normal program execution is suspended and control is transferred to the catch block. The catch block catches the exception and just suppresses it. Execution of the program continues after the catch block, *as if nothing had happened.*

How about the following?

public void someMethod() throws Exception{

}

This method is a blank one; it does not have any code in it. How can a blank method throw exceptions? Java does not stop you from doing this. The client code may circumvent the issue by just ignoring exceptions or throwing them, as in the previous two examples.

**Best Practices for implementing exception handling**

#### When deciding on checked exceptions vs. unchecked exceptions, ask yourself, "What action can the client code take when the exception occurs?"

* Try not to create new custom exceptions if they do not have useful information for client code.
* Always clean up after utilization
* Never use exceptions for flow control
* Do not suppress or ignore exceptions
* Log exceptions just once

## Configuration and Constants

All string constants and configuration items in the project must be abstracted out into configuration/global files, such as .properties files, .config files, .ini files or .java files. This includes, but is not limited to:

* String messages meant for display, logging or exception handling
* Database URLs, names connections and endpoints
* Database table names, column names, schema names
* Logging files, properties and configurations
* Numeric constants such as timeout values, port numbers, IP addresses, etc.

## Version Control and Build System

* Ensure that every commit and push to Github contains relevant comments and log message about the changes done.
* Always latest code from Github should be compiled successfully without any errors.
* Latest code push to Github should be performed on daily basis.