| o (department acronym name) | | For information (department acronym name) | |
| --- | --- | --- | --- |
|  |  |  |  |

SAF User Manual

The document describe the SAF test automation framework. The framework use a simple flow to interact with application. The framework tries to imitate the user behaviour as much as possible.

The figure bellow describes the procedure for identifying an elements on the screen and perform an action on the element.

# Architect

The architecture of the framework is made to handle desktop applications such as Catia as well as web application such as LCANAV.

The framework is based on:

* White: An API for test automation of desktop applications.
* Selenium: An API for web browser based application.
* Sikuli: image recognition test tool
* NUnit: A framework for creating unit test and supports White in terms of creating, running and managing test.
* Log4N: A logging framework.
* Perforce: Version control
* Jenkins: Continuous Integration, CI tool
* Written in C#, Visual Studio 2012 Pro

Furthermore the framework is based on a hybrid solution using:

* Test Script Modularity Concept
* Test Library Architecture Concept
* Keyword-Driven or Table-Driven Testing Concept
* Data-Driven Testing Concept
* Selenium Page Object Pattern
* Layer Concept, see figure below

# SAF development environment

The framework is based on .Net 4.5.1. It is recommended to use Microsoft C# Visual Studio 2012 or later.

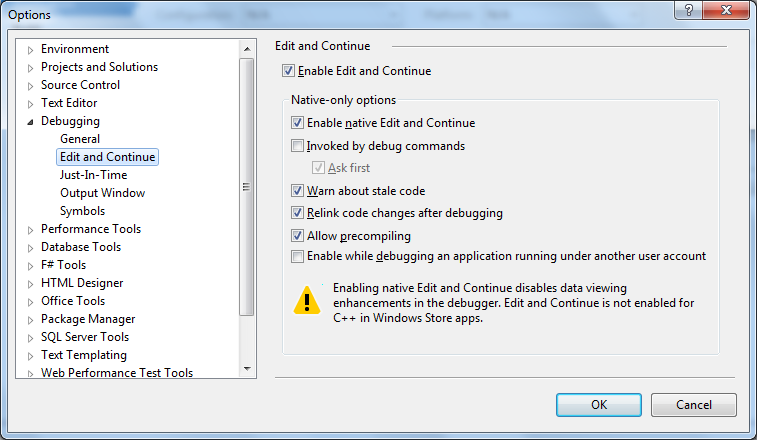
## SAF development environment configuration

Below describes some recommended configuration option:

### Edit and Continue

This configuration makes it possible to edit and continue debugging in the debugging mode.

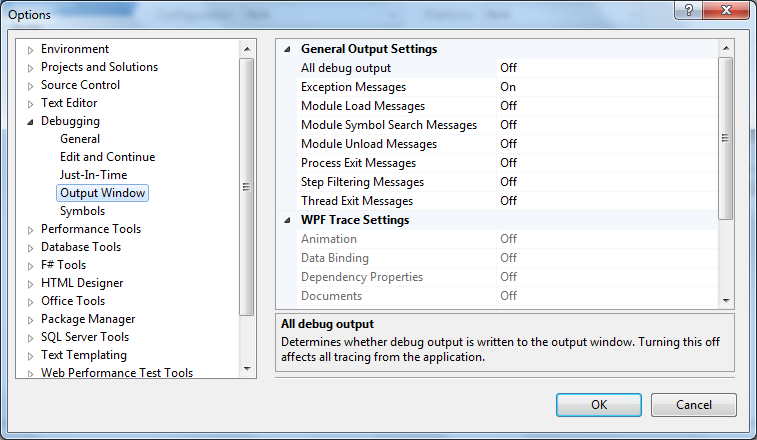
* Debug->Option and settings->Edit and Continue



### Test output window(Console)

This configuration gives a cleaner output to the console

* Debug->Option and settings->OutPut Window



# SAF installation

The steps below describes how to install the SAF test automation framework.

* Create a Dev catalogue, C:\Dev
* Download SAF from //cad-pdm/main/TestFwk/SAF to C:\Dev\SAF
* Doubbleclick SAF.sln

## SAF Configuration

The table below contains some of the configuration variables used in the SAF framework. The configurations facilitate the possibility to run the automated tests on different environment as well as modifying some parameters such as the timeout time.

| **Config Parameter Key** | **Values** | **Comments** |
| --- | --- | --- |
| ENVIRONMENT | * Acc * Prod * Test * Dev | The test environment |
| CATIA\_VERSION | * \_R19 * \_R24 | The version of the Catia application. Useful only during the coexistence of version R19 and R24. |
| **Path to the Catia application** | | |
| PATH\_APPLICATION\_TO\_LAUNCH\_CATIA\_5\_R24 | * C:\DS\B24SP1\win\_b64\code\bin\CATSTART.exe | Path for Catia R24 execution file |
| PATH\_APPLICATION\_TO\_LAUNCH\_CATIA\_5\_EXTENTION\_R24 | -run C:\DS\B24SP1\win\_b64\code\bin\CNEXT.exe -env CATIA.RoD.txt -direnv \\global\catiaconfig\ENV\_VARIABLE\_TO\_REPLACE\B24\CATEnv -nowindow | Extension of the execution path for the Catia R24 Application |
| PATH\_APPLICATION\_TO\_LAUNCH\_CATIA\_5\_R19 | C:\DS\B19SP7\win\_b64\code\bin\CATSTART.exe | Path for Catia R219 execution file |
| PATH\_APPLICATION\_TO\_LAUNCH\_CATIA\_5\_EXTENTION\_R19 | -run C:\DS\B19sp7\win\_b64\code\bin\CNEXT.exe -env CATIA.RoD.txt -direnv \\global\dfs03\ENV\_VARIABLE\_TO\_REPLACE\B19\SP7\_x64\CATEnv -nowindow | Extension of the execution path for the Catia R19 Application |
| APPLICATION\_PROCESS\_NAME\_CATIA\_5 | CNEXT | The name of the process for the Catia application. Used to attach the Catia application as the running application to test and operate on. Enhance the process of finding the window on the desktop. Instead of searching for all windows on the desktop the search for window will be done for the running application(process) |
| MAIN\_WINDOW\_CATIA\_V5\_VPM\_ROD\_R24 | CATIA V5 VPM RoD | The name of the main window for the Catia application R24. The name is valid in the initial state, then the name is dynamic and changed. |
| MAIN\_WINDOW\_CATIA\_V5\_VPM\_ROD\_R19 | CATIA V5.VPM.RoD. | The name of the main window for the Catia application R24. The name is valid in the initial state, then the name is dynamic and changed. |
| <!-- Indicate which sub menus should be activated--> |  |  |
| MAIN\_MENU\_APPLICATION\_DEFAULT\_SUB\_MENU\_NAMES | * LCA\_Icons * EnoviaLCA * SCVPartAndDocToolbar * SCVDataExchangeToolbar | The name of the sub menus on application menu. The SAF framework will check if these sub menus are shown, if not they will be shown in order to able to use them in the tests. |
| <!-- start desktop application true or false(false used for development)--> |  |  |
| START\_DESCTOP\_APPLICATION | * True * False | Indicate whether to start the desktop application or not.  True: A new application will launched and the latest process will be used as the running application  False: An existing application (the latest process) will be used as the running application |
| **<!-- CONTROL M-->** | | |
| PATH\_APPLICATION\_TO\_LAUNCH\_CONTROL\_M\_ENTERPRISE\_MANAGER | C:\Program Files\BMC Software\Control-M EM 7.0.00\Default\bin\emgui.exe |  |
| PATH\_APPLICATION\_TO\_LAUNCH\_CONTROL\_M\_ENTERPRISE\_MANAGER\_EXTENTION | “” | An empty string. No path extension is needed |
| APPLICATION\_PROCESS\_NAME\_CONTROL\_M\_ENTERPRISE\_MANAGER | emgui | The name of the process for the Control M application. Used to attach the Control M application as the running application to test and operate on. Enhance the process of finding the desktop window on the desktop. Instead of searching for all windows on the desktop the search for window will be done for the running application(process) |
| **<!--WEB APPLICATION-->** | | |
| WEB\_BROWSER\_TO\_RUN | * InternetExplorerDriver * FireFoxDriver * ChromeDriver | The type of the browser to launch, e.g: IE, Firefox or Chrome. IE is the preferred browser at Scania and most tested. |
| WEB\_APPLICATION\_URL\_ENOVIA\_PORTAL\_ACC | * Acc: <http://s0269:10929/B19ACC/html/ENOVIAPortal> * Test: http://n15043:12059/B19TEST/html/ENOVIAPortal | The URL to the Enovia web application |
| <!-- WEB\_APPLICATION\_CLEAR\_IE\_CACHE BOOLEAN VALUE true/false--> |  |  |
| WEB\_APPLICATION\_CLEAR\_IE\_CACHE | * True * False | True: Clear the cache for web browser application |
| <!--MOD ARC--> |  |  |
| WEB\_APPLICATION\_URL\_MOD\_ARC | * Acc: <http://modarc-acc.scania.com/enovia/WebClient/MatrixAppletXML.jsp> * Prod: http://modarc.scania.com/enovia/WebClient/MatrixAppletXML.jsp | The URL to the ModArc application. |
| APPLICATION\_PROCESS\_NAME\_MOD\_ARC | * java | The name of the process for the Mod Arc application. Used to attach the Mod Arc application as the running application to test and operate on. Enhance the process of finding the window on the desktop. Instead of searching for all windows on the desktop the search for window will be done for the running application(process) |
| **<!-- WEB AND DESKOP APPLICATION PROCESSES TO KILL-->** | | |
| <!-- BOOLEAN(true/false) INDICATING WHETHER TO KILL THE APPLICATION PROCESS--> |  |  |
| KILL\_APPLICATION\_PROCCESS" value="true"/> | * True * False | True: the application processes will be killed at the beginning of the test execution. |
| <!-- GIVE ALL THE PROCESSES SEPERATED BY COMMA "," --> |  |  |
| APPLICATION\_PROCESS\_NAMES\_TO\_KILL" value="CNEXT,CATSTART,CATSysDemon,CATJPESStarter,emgui"/> | * CNEXT * CATSTART * CATSysDemon * CATJPESStarter * Emgui * iexplore * internetexplorerdriver * Firefox * chome | The name of the processes o kill. Comma separated list. |
| **<!--TEST DATA ENV-->** | | |
| TEST\_DATA\_SHEET\_NAME | * TEST\_DATA\_DEV * TEST\_DATA\_ACC * TEST\_DATA\_PROD * TEST\_DATA\_TEST | The name of Excel sheet containing the test data. .. SAF\SAF\ParameterAndUserDataMgn\Test\_Data.xlsx  The test data in a cell is fetched based on the number of the row and the column name.  New columns can be added. The name of the column have to be unique |
| USER\_DATA\_SHEET\_NAME | * USER\_DATA\_DEV * USER\_DATA\_ACC * USER\_DATA\_PROD * USER\_DATA\_TEST | The name of Excel sheet containing the test data. .. SAF\SAF\ParameterAndUserDataMgn\User\_Data.xlsx  The user data in a cell is fetched based on the number of the row and the column name.  New columns can be added. The name of the column have to be unique |
| **<!-- TIME OUT PARAMETERS FOR WEB AND DESKTOP APPLICATIONS -->** | | |
| GLOBAL\_TIMEOUT\_TIME\_IN\_MSEC | * 0,1,2,3,... (in msec) | This value is used as a global time out for waiting for windows(desktop and browser applications).  An exception is thrown in case of time out.  Note this value can be changed dynamically by assigning GLOBAL\_TIMEOUT\_TIME\_IN\_MSEC a new value, e.g: ConfigParameters. GLOBAL\_TIMEOUT\_TIME\_IN\_MSEC=30000 |
| GLOBAL\_TIMEOUT\_TIME\_IN\_MSEC\_WITHOUT\_FAILING | * 0,1,2,3,... (in msec) | This value is used as a global time out for waiting for windows(desktop and browser applications).  NO exception is thrown in case of time out.  Note this value can be changed dynamically by assigning GLOBAL\_TIMEOUT\_TIME\_IN\_MSEC\_WITHOUT\_FAILING a new value, e.g: ConfigParameters. GLOBAL\_TIMEOUT\_TIME\_IN\_MSEC\_WITHOUT\_FAILING =10000 |
| START\_APPLICATION\_TIMEOUT\_TIME\_IN\_MSEC | * 0,1,2,3,... (in msec) | This value is used as a global time out for waiting for desktop applications to start.  An exception is thrown in case of time out.  Note this value can be changed dynamically by assigning START\_APPLICATION\_TIMEOUT\_TIME\_IN\_MSEC a new value, e.g: ConfigParameters. START\_APPLICATION\_TIMEOUT\_TIME\_IN\_MSEC =60000 |
| **<!--WHITE INERNAL TIMEOUT CONFIGURATION-->** | | |
| WHITE\_BUSY\_TIMEOUT | * 0,1,2,3,... (in msec) | This value is used to change the White-API time out. This is a general time out, e.g. WHITE\_BUSY\_TIMEOUT=50000, will throw an exception if try to click a button and the button is not found. |
| WHITE\_UI\_AUTOMATION\_ZERO\_WINDOW\_BUG\_TIMEOUT | * 0,1,2,3,... (in msec) | This is a general time out to handle a bug in the .Net automation framework. |
|  |  |  |
|  |  |  |
| MEASURE\_RESPONSE\_TIME | * True * Fals | This value could be used to measure response time e.g. for clicking the button. |

# Name convention

This section contains the name convention of the test classes, test cases, action method, etc.

## Test classes and test cases name convention

The name of a test class and test should follow the conventions below. A common name makes the code easier to understand and maintain.

* Test class and test case names should end with Test, e.g: CatiaLoginPageTest
* Web application classes ends with Page, e.g: LoginPage (includes operation on elements)
* Desktop application classes ends with Window, e.g: LoginWindow (includes operation on elements)

## Operation on Element Name Comventoin

* Element: <Element><ElementName><Operation>,
  + E.g. Button: ButtonLoginClick
  + E.g. Field(text box) FieldUserNameInsert

# Test class and test case

This section describes the construction of the test classes and test cases

## Test class

A test class should include:

* pre condition on a class level
* post condition on a class
* pre condition on test case level
* post condition on test level

[TestFixtureSetUp] //Actions to be taken before each test class

public void PreConditionTestSuite()

{

}

[SetUp] //Actions to be taken before each test case

public void PreConditionTestCase()

{

}

[TearDown] //Actions to be taken after each test case

public void PostConditionTestCase()

{

}

[TestFixtureTearDown] //Actions to be taken after each test class

public void PostConditionTestSuite()

{

}

## Test case

A test case contains the notation [Test] according to NUnit, e.g:

[Test]

public void SimpleLoginPageTest()

{

loginpage.FieldUserNameInsert("user1");

loginpage.FieldUserPasswordInsert("user1");

loginpage.ButtonConnect();

}

# Jenkins

The following section describes how setup and use Jenkins in the framework in order to facilitate the automation and scheduling.

## Setup Jenkins

Jenkins is a continuous integration tool. The following steps describes how setup Jenkins:

* Create a catalogue named Dev under C, c:/Dev
* Download the Jekins projects under Dev, c:/Dev/Jenkins
* Make sure JAVA\_HOME is giver
* Make sure Path is given, %JAVA\_HOME%\bin
* Edit the file c:/Dev/Jenkins/jenkins.bat, if you need to start Jekins on different port than httpPort=8090.
* Dubble click jenkins.bat to start the jenkins service.
* Start firefox and go to. [http://localhost:8090](http://localhost:8090/)
* Rename DevJob.
* Reconfigure DevJob:

# InternetExplorerDriver

The IEDriver needs some special configurations in order to make it work. One very important configuration is, Enable Protected Mode which needs to be set to True:

* Tools->Internet Options->
  + Enable Protected Mode = True.

# Sikuli

Sikuli uses image recognition technologie to identify and control GUI components. It is useful when there is no easy access to a GUI's internal or source code. Sikuli is an open-source research project originally started at the User Interface Design Group at MIT, <http://www.sikuli.org>.

The Sikuli action is run as shell execution by calling the Sikuli-IDE.bat. The Sikuli command is created within the framework saved in file temp.py and then the Sikuli-IDE.bat is called with temp.py as argument.

The figure below shows the flow used in the SAF framework in order to execute a Sikuli action on an element on the screen.

For more information about the source code and class diagram, see the HTML documentation.

# Silenium

Selenium is a portable software testing framework for web applications. Selenium supports the most common browsers, such as IE, Chrome, Firefox, [www.seleniumhq.org](http://www.seleniumhq.org)

The SAF framework uses the Selenium API mainly for the web applications, ControlM and Modarc.

The figure below shows the flow for how to use the Selenium API in the SAF framework.

# White

White supports functional testing for Win32, WinForm, WPF and SWT applications. White also makes use of the Windows Messages where UIA doesn’t provide all the required support. But these usages are abstracted from the users of white API. White is built on .NET platform using C#, <http://teststack.net/White/>.

The SAF framework uses the White API mainly for the desktop application, such as Catia.

The figure below shows the flow for how to use the White API in the SAF framework.