IDM UID
9YL8QC
VERSION CREATED ON / VERSION / STATUS
20 Jun 2013 / 2.4 / Signed
EXTERNAL REFERENCE

Report

CODAC Operator Interface - Test Plan

Test of CODAC Operator Interface

Approval Process			
	Name	*Action*	*Affiliation*
Author	**Utzel N.**	**20-Jun-2013:signed**	**IO/DG/DIP/CHD/CSD/CDC**
Co-Authors			
Reviewers			
Approver			
Document Security: level 1 (IO unclassified) *RO: Di Maio Franck*			
Read Access	**RO, project administrator, LG: SOPRA extra, AD: ITER, AD: External Collaborators, AD: Division - Control System Division, AD: Section - CODAC, AD: Auditors, AD: ITER Management Assessor**		

Change Log				
Title (Uid)	***Version***	***Latest Status***	***Issue Date***	***Description of Change***
CODAC Operator Interface - Test Plan (9YL8QC_v2_4)	v2.4	Signed	20 Jun 2013	Check for SEVERE messages in the log files
CODAC Operator Interface - Test Plan (9YL8QC_v2_3)	v2.3	Signed	04 Jun 2013	Web OPI Test Case Start an OPI from command line Test Case
CODAC Operator Interface - Test Plan (9YL8QC_v2_2)	v2.2	Signed	13 Feb 2013	Additional tests: - data browser widget - vertical/enabled tabbed container - setpoint adjustement with array widget - CODAC DDD requirement for Update rate on HMI (PON) = 200 PVs at 5Hz
CODAC Operator Interface - Test Plan (9YL8QC_v2_1)	v2.1	Signed	01 Feb 2013	SVN Demo Unit
CODAC Operator Interface - Test Plan (9YL8QC_v2_0)	v2.0	Signed	22 Jan 2013	Additional tests: - Symbol and Image widgets rotation - Symbol and Image widgets flip/flop - Symbol and Image file selection - On/Off Symbol color - Performance of Symbol widgets
CODAC Operator Interface - Test Plan (9YL8QC_v1_3)	v1.3	Signed	10 Oct 2012	Test of civil engineering and I&C symbols as well as CODAC BOY Schema.
CODAC Operator Interface - Test Plan (9YL8QC_v1_2)	v1.2	Signed	07 Jun 2012	Pagination problem due to the automatic cover page
CODAC Operator Interface - Test Plan (9YL8QC_v1_1)	v1.1	Signed	07 Jun 2012	At CSS startup, the Welcome screen should be closed
CODAC Operator Interface - Test Plan (9YL8QC_v1_0)	v1.0	Signed	06 Jun 2012	

CODAC Operator Interface

Software Test Plan (STP)
Based on QA Template Version <1.0>

This document describes the tests that should be performed for CODAC Operator Interface in order to be installed as part of Core System release. Different test cases are described, as well as and test pass-fail criteria.

Contents

1 INTRODUCTION

1.1 Purpose

This document describes the tests that should be performed for CSS BOY - Best Operator Interface (OPI), Yet - in order to be installed as part of CODAC Core System. These tests will ultimately compare the capabilities of BOY against these described in CODAC System Requirement (SRD) Document [IDM 28C2HL].

The Operator Interface is the graphical user interface that displays live control system data to operators and allows them to input data to the control. Particular functions to be tested are Operator Interface (OPI) development and runtime features.

1.2 Scope

The test items are:

- The operational version of BOY,
- The data, including all the configuration data needed to run the operator interface,
- The documentation, including the online help and the release notes.

The installation and uninstallation of the components are not part of this test plan.

1.3 System/Software overview and key features

BOY is a development and runtime environment:

- Edition of Operator Interface screen using graphical widgets and CODAC electrical and fluid standardised symbols. BOY Editor has most of the modern editing features which facilitate the OPI editing greatly, such as copy, paste, undo, redo, arrange multiple widgets, snap to grid or other widgets (geometry), ruler, guide, zoom in/out, change order, create…
- Open and run the Operator Interface, draw widgets on the screen, connect to the control system, update for example text fields, meters, and bar graphs to reflect the current values of EPICS PVs, offer dials or text boxes to enter or adjust PV values, which are then written to the control system.

1.4 References

CODAC Quality assurance plan - https://user.iter.org/?uid=6J7RW4

2 DETAILS OF THE TESTING PROCESS

2.1 Definition of test levels

The described component tests will focus on the desired features of CODAC Operator Interface.

Following test levels are defined in this test plan to organize the testing activity.

Operator Interface Edition Component Test	**EDT**
Test of the operator interface creation	
Operator Interface Runtime Component Test	**RNT**
Test of running the operator interface	
Operator Web Interface Runtime Component Test	**WEB**
Test of running the operator interface using a web navigator	
Symbol Library Component Test	**SMB**
Test of archived data plot in CSS	
Operator Interface Performance Test	**PRF**
Test of at least 200 widgets running on the operator interface	

2.2 Test administration

2.2.1 Anomaly resolution and reporting

Anomaly Reports shall be submitted in Bugzilla.

2.2.2 Test reporting requirements

The test logs shall be generated to record the outcome of test procedures as described in section *.4 and *.5 of the level test plans.

2.2.3 Test deliverables

The test deliverables include:

- Component Test Logs / Reports
- Anomaly Reports with Bugzilla bug references.

Test input data are registered in SVN code repository.

No other test tool is needed.

The test reports may be submitted on ITER IDM.

3 COMPONENT TEST PLAN

3.1 Scope

3.1.1 Test items and their identifiers

CODAC Operator Interface is included the following unit:

- m-css in the product:
  - org.csstudio.iter.css.product/

CODAC Operator Interface mainly depends on:

  - org.csstudio.iter.opibuilder.feature

CODAC Operator Interface includes ITER specific development:

  - org.csstudio.opibuilder.widgets.symbol
  - org.csstudio.opibuilder.imagelib

3.1.2 Features to be tested

The main CODAC Operator Interface features to be tested are:

- Operator Interface Edition and Run
- Web Operator Interface Run
- CODAC Standard Symbol Library

3.1.3 Features not to be tested

Scripting using Python and Javascript will not be tested.

3.2 Approach

3.2.1 Testing Methods

The overall approach for the level of testing is the Black box method to test the functionality of CODAC Operator Interface.

3.2.2 Item pass/fail criteria

Each major anomaly found determines whether each test item has passed or failed testing.

3.3 Environment / Infrastructure

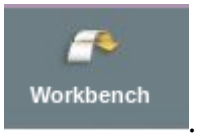
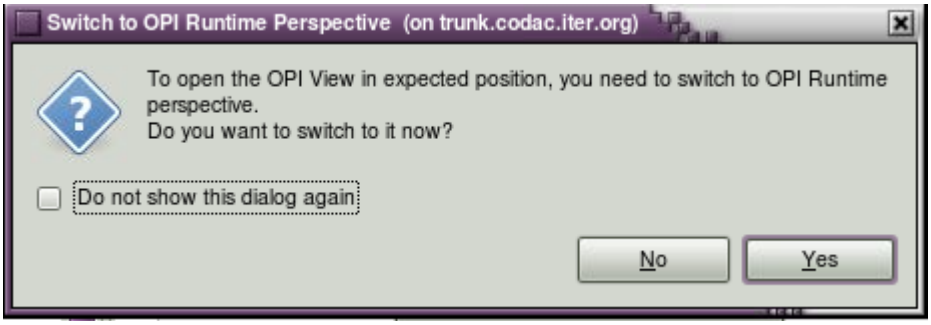
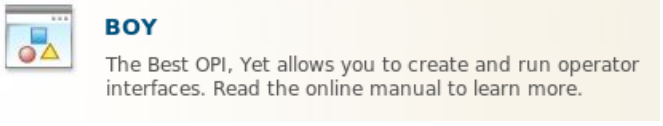
Core System in its development role version should be installed on a CODAC standard machine. Access to SVN is required.

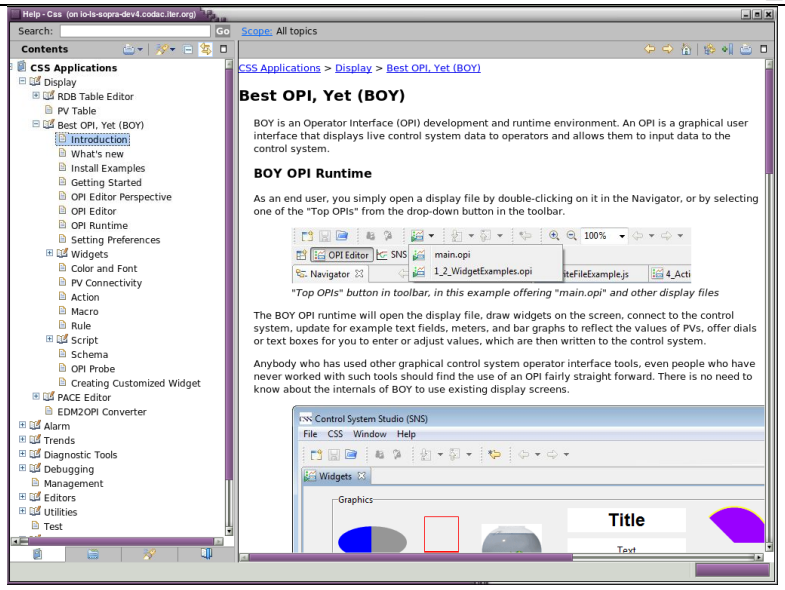
3.4 Component Test Procedures

	3.4.1 EDT01 - Operator Interface Examples

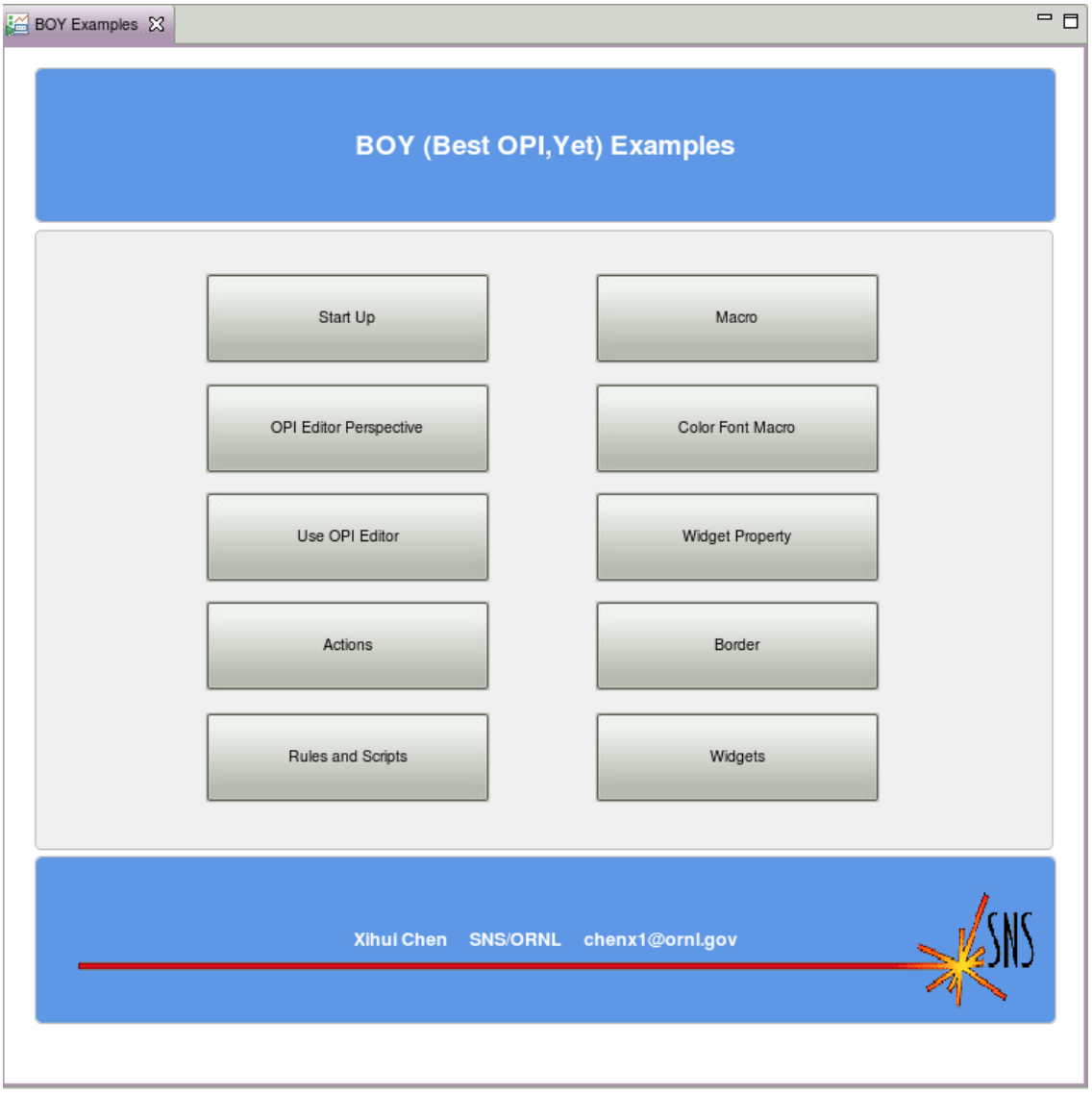
Prerequisite	In a Linux console, download and start a demo IOC:
	0. $ rm -Rf ~/.css
	1.$ mkdir test
	2.$ cd test
	3.$ svn co https://svnpub.iter.org/codac/iter/codac/dev/units/m-css/trunk/products/ITER/products/org.csstudio.iter.css.product/demo/m-TEST-BOY
	`A m-TEST-BOY`
	`A m-TEST-BOY/.project`
	`A m-TEST-BOY/doc`
	`A m-TEST-BOY/doc/STP-CODAC_Operator Interface.pdf`
	`A m-TEST-BOY/src`
	…
	`A m-TEST-BOY/sdd.xml`
	`A m-TEST-BOY/pom.xml`
	`Checked out revision xxxx.`
	4.$ cd m-TEST-BOY
	5.$ softIoc -s -d src/main/epics/TEST-BOY0App/Db/PSH0-TEST-BOY0.db
	`Starting iocInit`
	`############################################################`
	`## EPICS R3.14.12.3-rc1 $Date: Mon 2012-12-03 16:39:27 -0600$`
	`## EPICS Base built Dec 10 2012`
	`############################################################`
	`cas warning: Configured TCP port was unavailable.`
	`cas warning: Using dynamically assigned TCP port 37073,`
	`cas warning: but now two or more servers share the same UDP port.`
	`cas warning: Depending on your IP kernel this server may not be`
	`cas warning: reachable with UDP unicast (a host's IP in EPICS_CA_ADDR_LIST)`
	`iocRun: All initialization complete`
	6. List of all defined PVs in this IOC:
	`epics> dbl`
	`TEST-BOY0:AI1`
	`TEST-BOY0:AI2`
	`TEST-BOY0:BI`
	`TEST-BOY0:BO`
	`TEST-BOY0:RAMP1`
	`TEST-BOY0:RAMP2`
	`TEST-BOY0:RNDM-AI`
	`TEST-BOY0:RNDM-BI`
	`TEST-BOY0:RNDM-MBBI`
	`TEST-BOY1:RNDM-STATE`
	`TEST-BOY0:COMPRESS`
	`TEST-BOY0:LONGIN`
	`TEST-BOY0:MBBI`
	`TEST-BOY0:SWITCHSTATE`
	`TEST-BOY0:CMDSWITCH`
	`TEST-BOY0:MBBO`
	`TEST-BOY0:STRING`
	`TEST-BOY0:SETPOINT`
	`TEST-BOY0:WAVEFORM`
	`epics>`

Test Cases	1. Positive confirmation of the operator interface examples installation and run

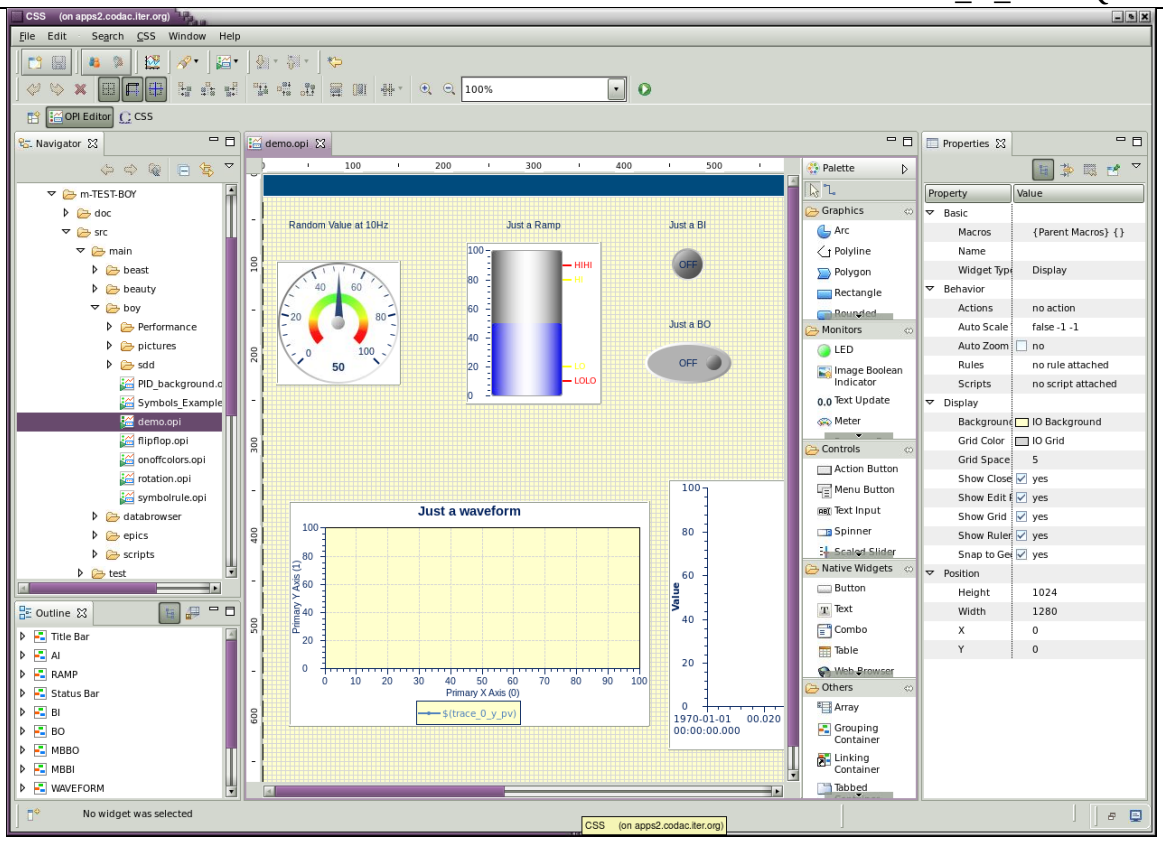
Procedure	In another Linux console, start the development environment to edit a new operator interface: 1.$ cd test 2.$ css& 3. Browse to select the working directory test Check the Welcome Pages and online Help: 4. From the “Welcome to CSS for ITER!” Page, click on “First Steps”. A short description of CSS BOY should be given. Click then on the link “Boy” in this “First Steps” page, just before the short description. The Online Help is displayed. 5. Close the Online Help windows and Close the Welcome screen by clicking on Workbench icon:  Install BOY Examples using the menu CSS: 6. CSS -> Display -> Install OPI Examples 7. From the Navigator View, browse the new folder “BOY Examples”  And double-click on the file named “main.opi” to open the top page of BOY Examples. Then click on the button Widgets, confirm that you want to switch to OPI Runtime perspective in order to have a better view:  Finally click on different BOY Widgets buttons – Graphics, Monitors, Controls and Others. Close the “Widgets” View before moving on the next step.	
Pass Criteria	4. Welcome First Steps for CSS BOY should appear:  After clicking on BOY link, the Online Help is opened on the BOY topic:	



7. The main BOY Examples screen allows to explore the main features of BOY:

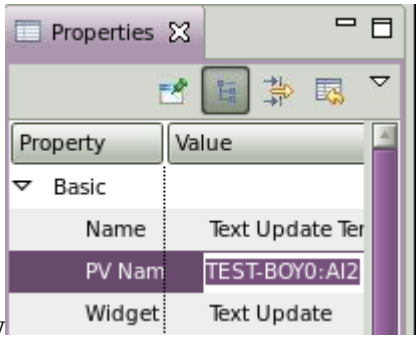


	3.4.2 EDT02 - Operator Interface Edition	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the operator interface edition	
Procedure	In CSS, switch back to ITER Control System Studio perspective by clicking on the dedicated button  . Close all previously opened OPI files with a right-click on any OPI tab and by selecting the option  . Import the project checked out from SVN: 1. In the Navigator View, right-click and select the contextual menu Import… -> General -> Existing Projects into Workspace. Click on Next>. To specify the root directory, use the Browse button and select m-TEST-BOY and click on OK. Click on Finish 2. Open the Editor Perspective: menu Window -> Open Perspective -> Other… and select OPI Editor 3. In the Navigator View, browse m-TEST-BOY -> src -> main -> boy. Right-click on demo.opi -> Open With -> OPI Editor 4. From the Palette -> Monitors, use the little arrows to browse the widgets, drag and drop the widget “Text Update” on the grid – for instance below the Gauge displaying a random value at 10Hz 5. Select the new widget on the grid. The small square handle on top left of the widget allows the edition directly of the PV Name: click on it and replace the current simulated PV by “TEST-BOY0:AI2”. PV Auto Complete should help! Press on Enter 6. From the Palette -> Controls, use the little arrows to browse the widgets, drag and drop the widget “Boolean Symbol Control” on the grid below “Just a BO” button. Click on the widget and modify in the Properties View the PV Name to “TEST-BOY0:BO”. Press on Enter. Finally, under Display, change the property Symbol Image and choose from the Symbol Library the following symbol: “SymbolLibrary/SVG/Electrical/Protection_and_Cut-Shut_Off-On/JF Fuse Switch Off.svg”. Change the Boolean Label Position property to “Bottom” 7. Select the 2 new added widgets on the grid and use the toolbar to align them horizontally in the middle and distributing them by horizontal compress gaps:  8. Save the modified OPI file using the menu File -> Save (Ctrl+S).	
Pass Criteria	3. The demo.opi BOY screen should be displayed in CSS OPI Editor Perspective:	

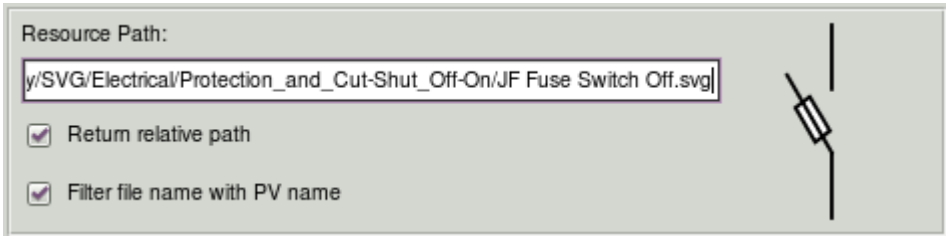


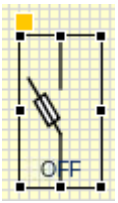
4-5. The Text Update widget should appear on the grid and its basic Properties should refer to the PV Name to be displayed:

Widget on the Grid 

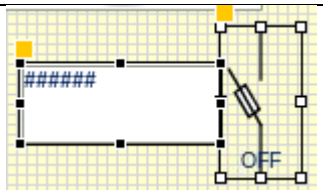
Properties View 

6. Select the fuse electrical symbol in its svg format in Off position:

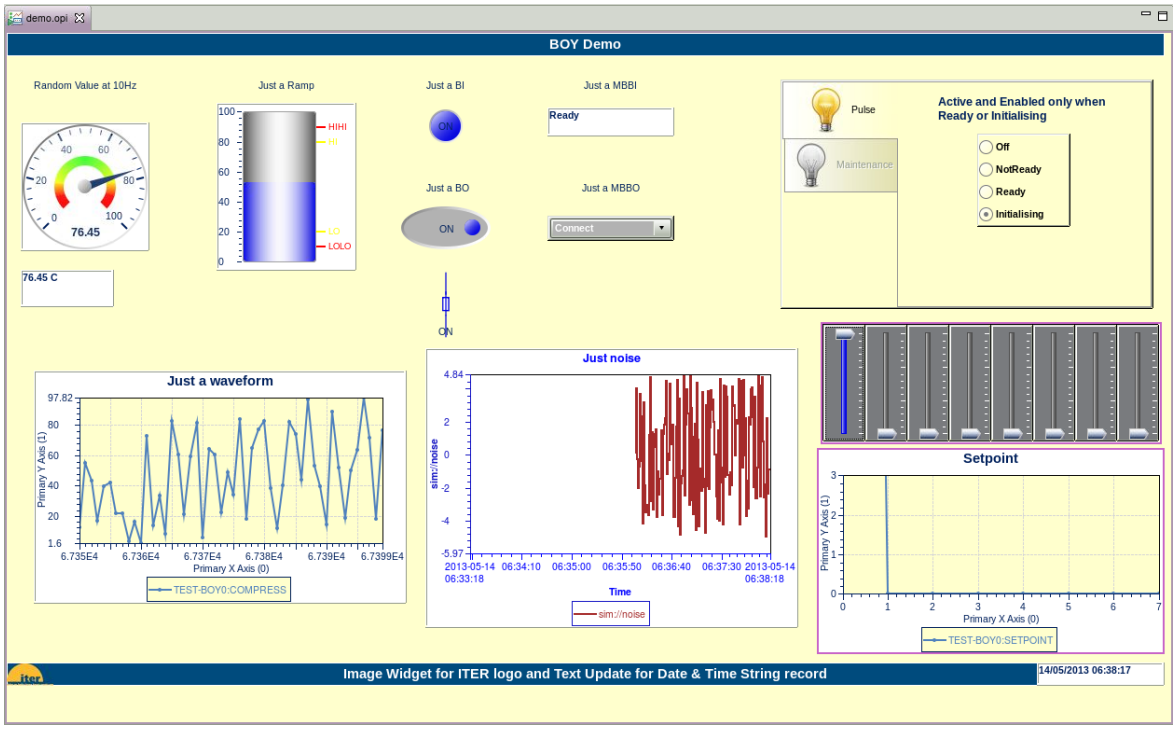


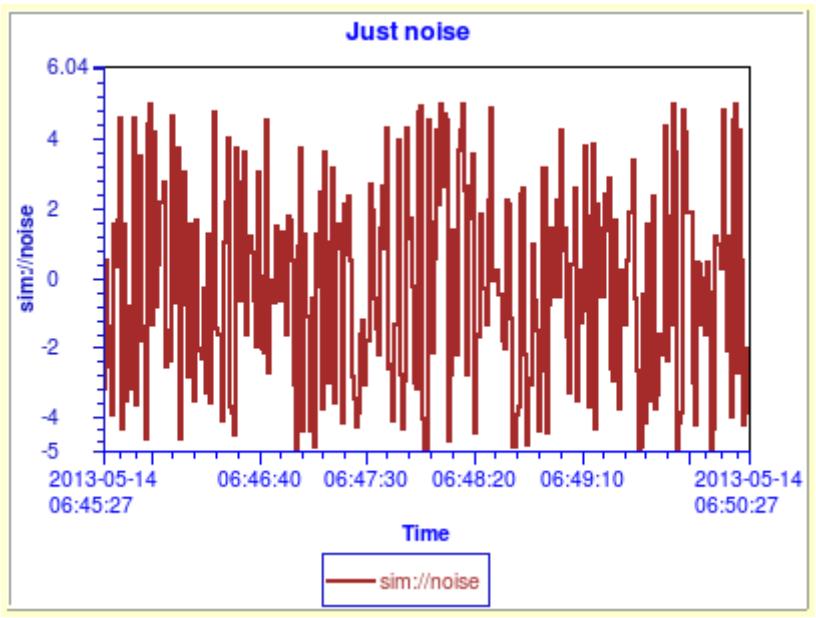
Label position = Bottom: 

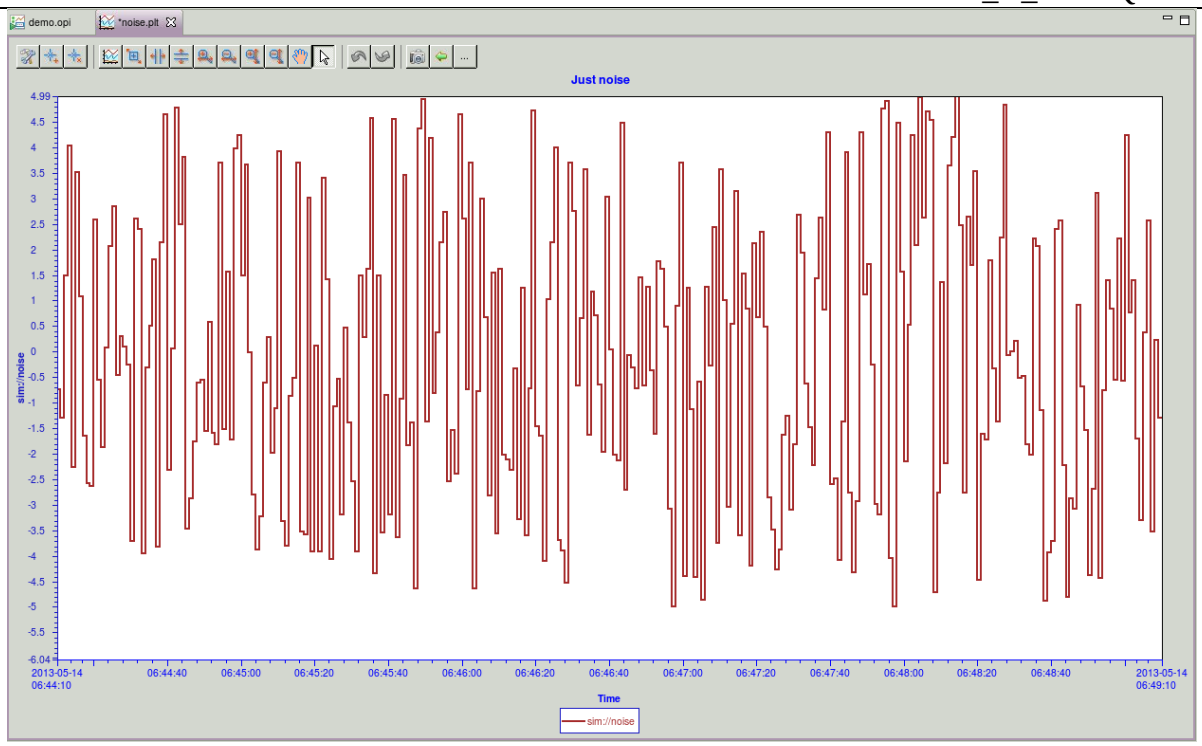
8. Widgets aligned and distributed:

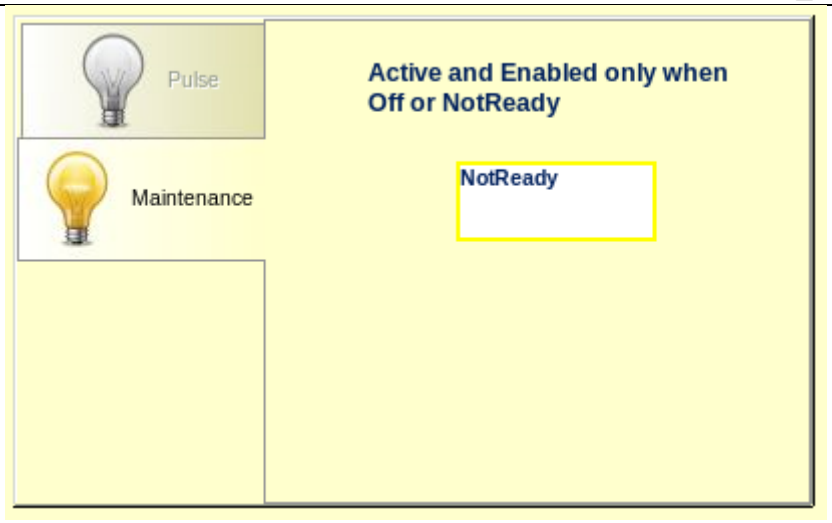


3.4.3 RNT01 - Operator Interface Runtime

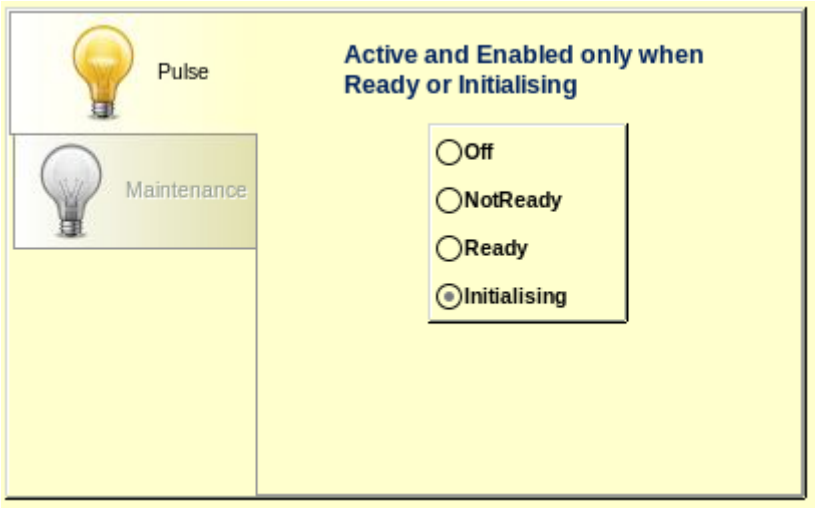
Prerequisite	1. Demo IOC running 2. CSS started 3. BOY Runtime opened
Test Cases	1. Positive confirmation of the operator interface running
Procedure	1. Run the Operator Interface using the button from the tool bar  . Check on the Operator Interface the Title displayed using a Label widget, the AI random value displayed with a Gauge and its border alarm sensitive, the CALC record producing a ramp displayed using a vertical "Progress Bar" widget and its alarm limits, the BI displayed with a LED widget, the BO represented by a Button, the MBBI value displayed with a "Text Update" widget, the MBBO presented as a Combo box, the Waveform of 50 elements displayed on a "XY Graph", the ITER logo displayed using an Image widget and the Date & Time String record. Finally check the 2 new widgets displaying TEST-BOY0:AI2 at 10Hz and . TEST-BOY0:BO. Click on the fuse symbol, confirm the action and check that the BO is in its On position. The Tabbed Container, Data Browser and Array widgets will be tested afterwards.
Pass Criteria	1. The Operator Interface to monitor the different types of records should be at Runtime: 

	3.4.4 EDT03 - Data Browser widget
Prerequisite	1. Demo IOC running 2. CSS started 3.BOY Runtime opened
Test Cases	1. Positive confirmation of the integration of a plot configuration into BOY
Procedure	In CSS, from the previous ran Operator Interface demo demonstrating some BOY widgets in the OPI Runtime perspective: 1. Check the noise signal displayed using BOY Data Browser widget. Then, compare the previous display in BOY with the signal plotted directly in the Data Browser: 2. From the bottom bar, click on  icon, select Other…, then General -> Navigator  A Fast View Icon for the Navigator appears in the bottom bar  3. From the Navigator View, browse m-TEST-BOY -> src -> databrowser - and double-click on “noise.plt” to open the plot
Pass Criteria	1. The simulated noise signal should be displayed in the previous ran demo OPI:  3. The plot noise.plt should have the same configuration as the noise signal displayed in BOY: same Title label, colour and font, same axis properties and same noise signal:

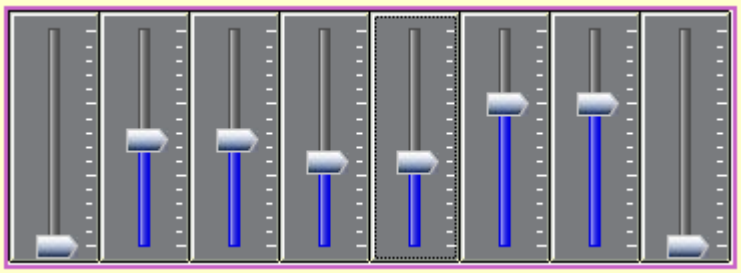
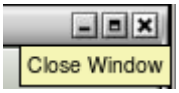
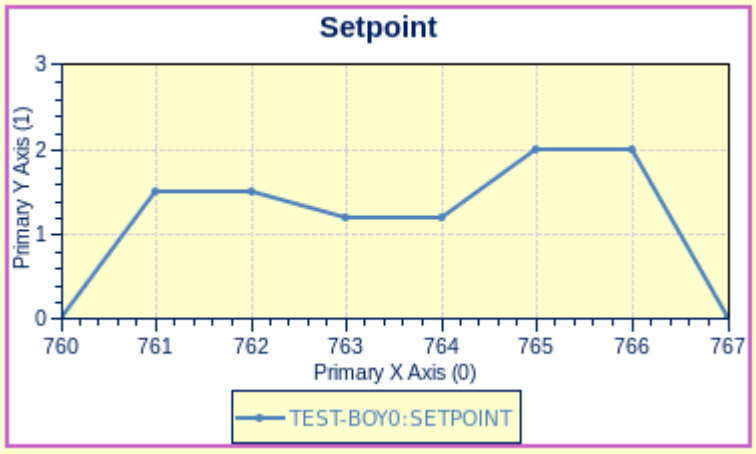
		
	3.4.5 EDT04 - Vertical and Enabled Tabbed Container	
Prerequisite	1. Demo IOC running 2. CSS started 3.BOY Runtime opened	
Test Cases	1. Positive confirmation of the horizontal position and enable/active tab	
Procedure	In CSS, from the previous ran Operator Interface, click on demo.opi tab: 1. Check the vertical position of the tabs 2. Check that the Maintenance tab is activated only when the state is Off or NotReady 3. Check that the Pulse tab is activated only when the state is Ready or Initialising	
Pass Criteria	1-2. The 2 tabs are in vertical position and Maintenance tab is activated when NotReady:	

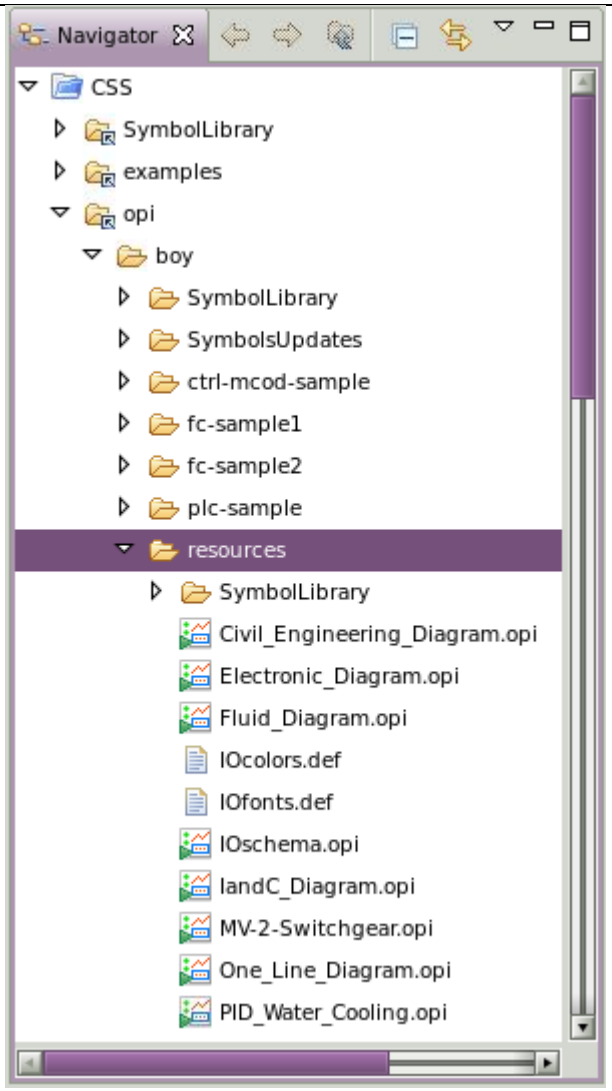


3. Pulse tab is activated when Initialising:



	3.4.6 EDT05 - Setpoint adjustment
Prerequisite	1. Demo IOC running 2. CSS started 3.BOY Runtime opened
Test Cases	1. Positive confirmation of the setpoint adjustment
Procedure	In CSS, from the previous ran Operator Interface demo.opi tab: 1. Adjust the sliders to modify the setpoint: first slider remains 0.0, the second and third are 1.5, fourth and fifth are 1.15, sixth and seventh are 2.0 and the eight one remains 0.0:

	 2. Check the setpoint waveform 3. Just close the Runtime environment by clicking on the cross close button of the window:  If a “Save Resource” popup asks for a confirmation before saving cahnges on “noise.plt”, click on “No”.	
Pass Criteria	2. The setpoint waveform should look like that:  	
	3.4.7 SMB01 - CODAC Standard Symbol Lib	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the standard symbol library	
Procedure	In CSS, in the OPI Editor perspective, run the Operator Interfaces demonstrating for example the use of all electrical and fluid symbols: 1. In the Navigator View, browse CSS -> opi -> boy -> resources which point to the location /opt/codac/opi/boy/resources	



2. From the Navigator View, double-click on Fluid_Diagram.opi file to display all fluid symbols defined in the library. In order to have a better view, Switch to OPI Runtime Perspective

3. Switch back to OPI Editor Perspective and from the Navigator View, double-click on One_Line_Diagram.opi file to display all electrical symbols defined in the library. Switch to OPI Runtime Perspective

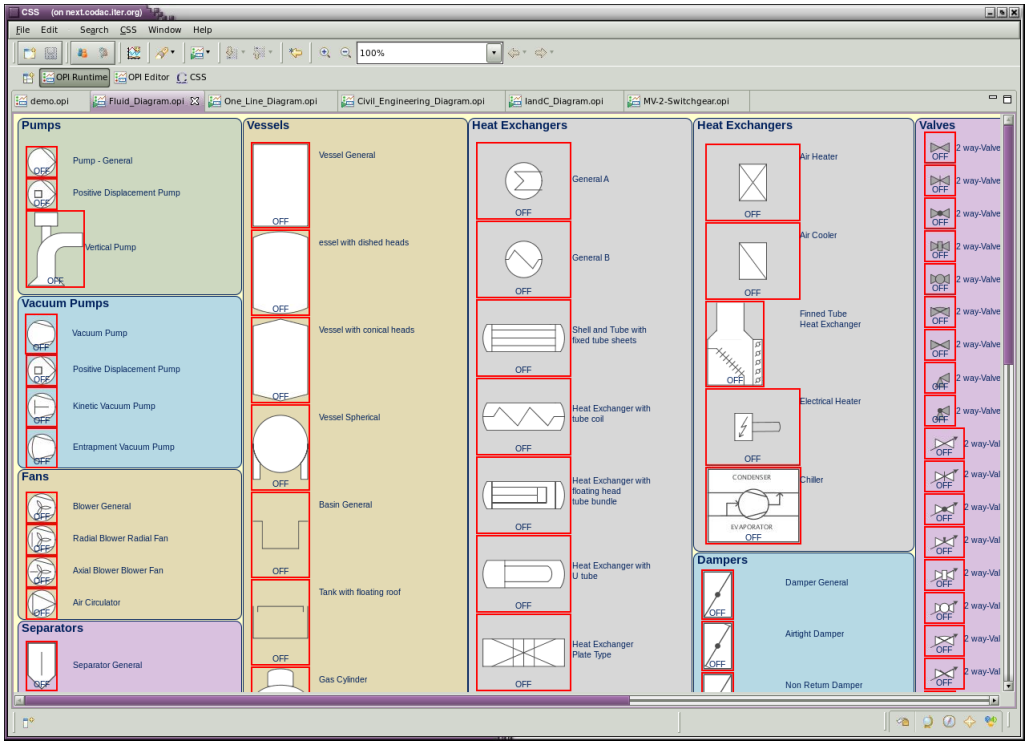
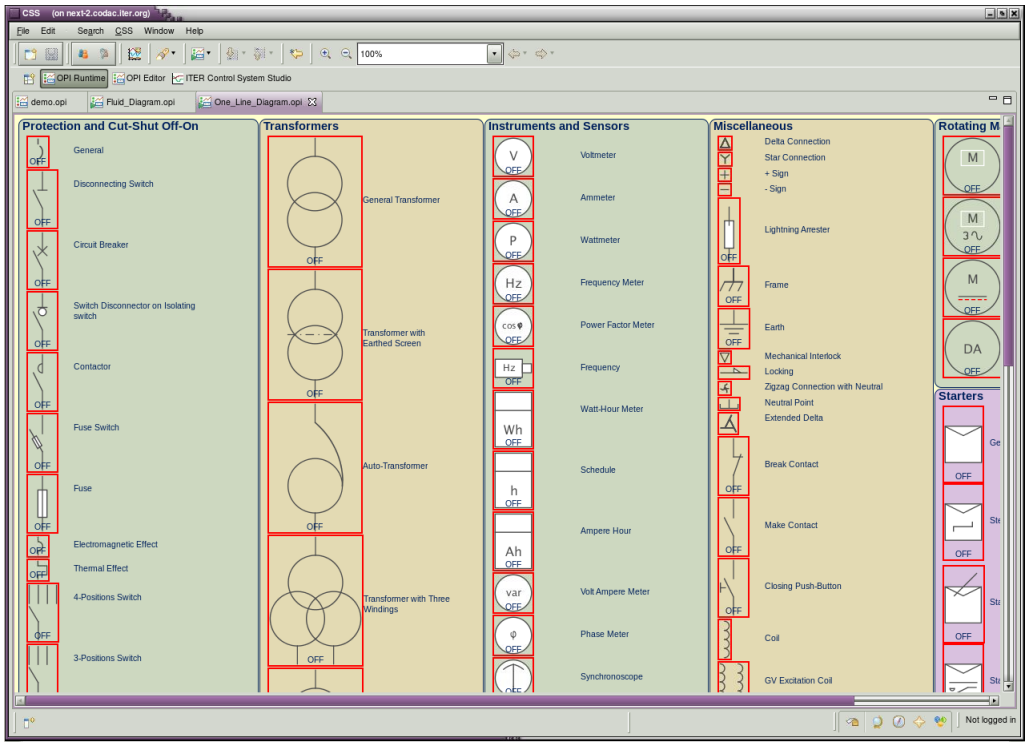
4. Switch back to OPI Editor Perspective and from the Navigator View, double-click on Civil_Engineering_Diagram.opi file to display the related symbols defined in the library. As the BOY screen is small, there is no need to switch to OPI Runtime Perspective to have a better look

5. From the Navigator View, double-click on IandC_Diagram.opi file to display the related symbols defined in the library

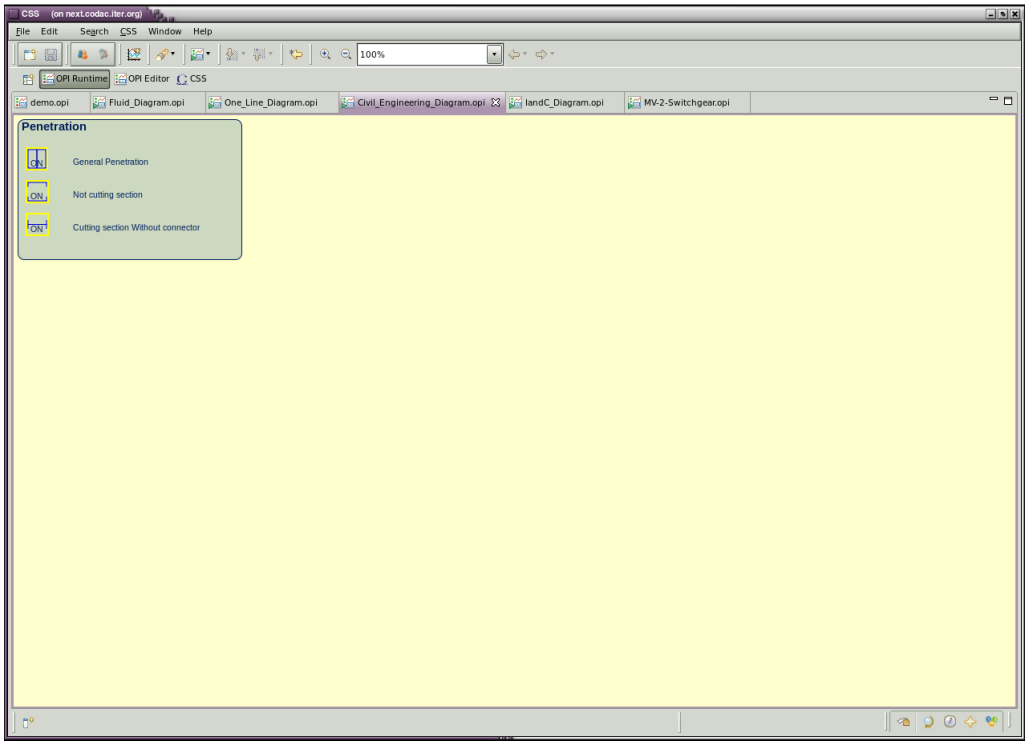
6. From the Navigator View, double-click on MV-2-Switchgear.opi file to display an electrical use case schema. Switch to OPI Runtime Perspective

7. Switch back to OPI Editor Perspective and from the Navigator View, double-click on PID_Water_Cooling.opi file to display a fluid use case schema. Switch to OPI Runtime Perspective

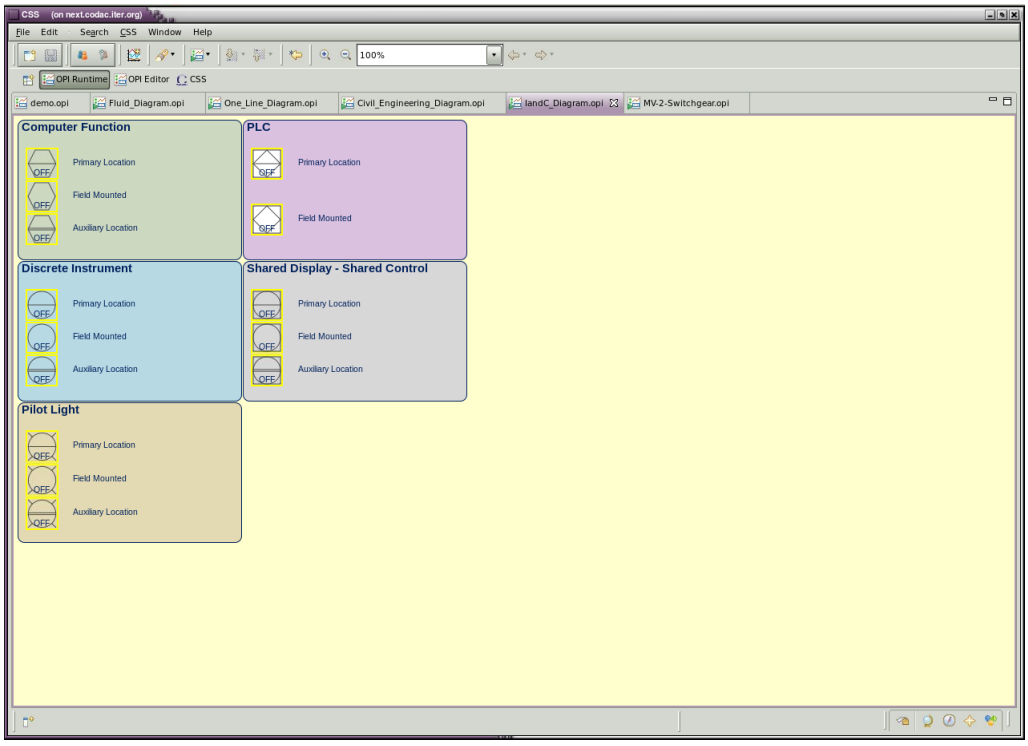
8. Switch back to OPI Editor Perspective and from the Navigator View, double-click on

	IOschema.opi file to display the default layout of all BOY widgets for CODAC. Switch to OPI Runtime Perspective 9. Switch back to OPI Editor Perspective and close all screens opened
Pass Criteria	2. *After some seconds – the time to load the ~130 symbols*, the fluid symbols should be animated on the screen at 1 second:  3. The output for the electrical symbols should be: 

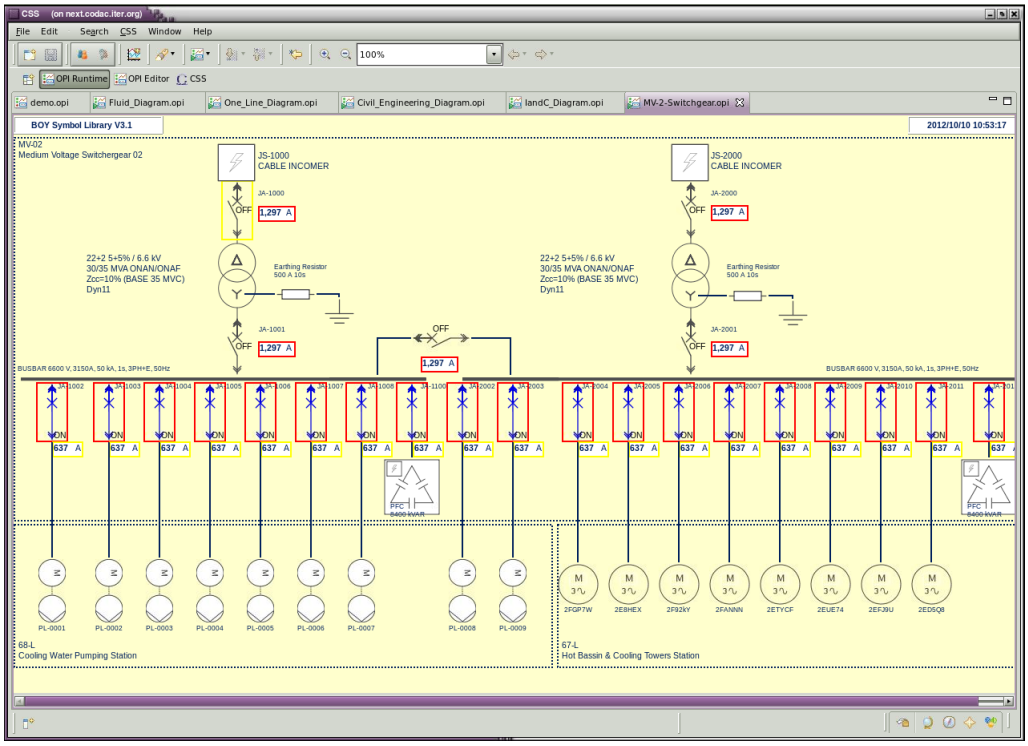
4. The output for the civil engineering symbols should be:



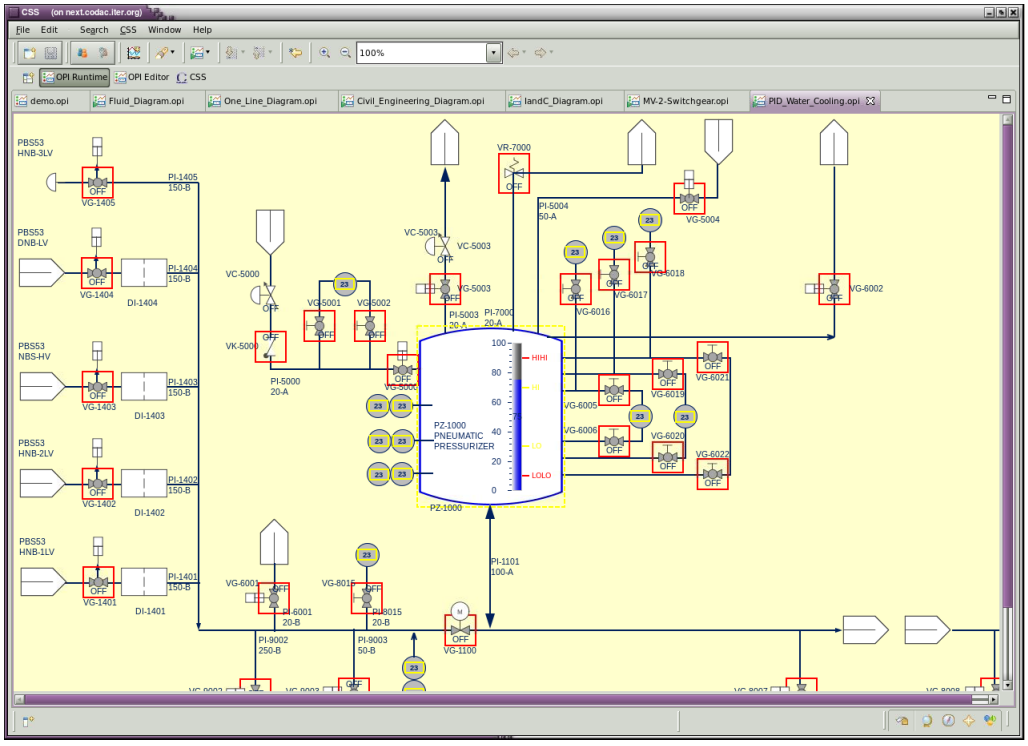
5. The output for the I&C symbols should be:

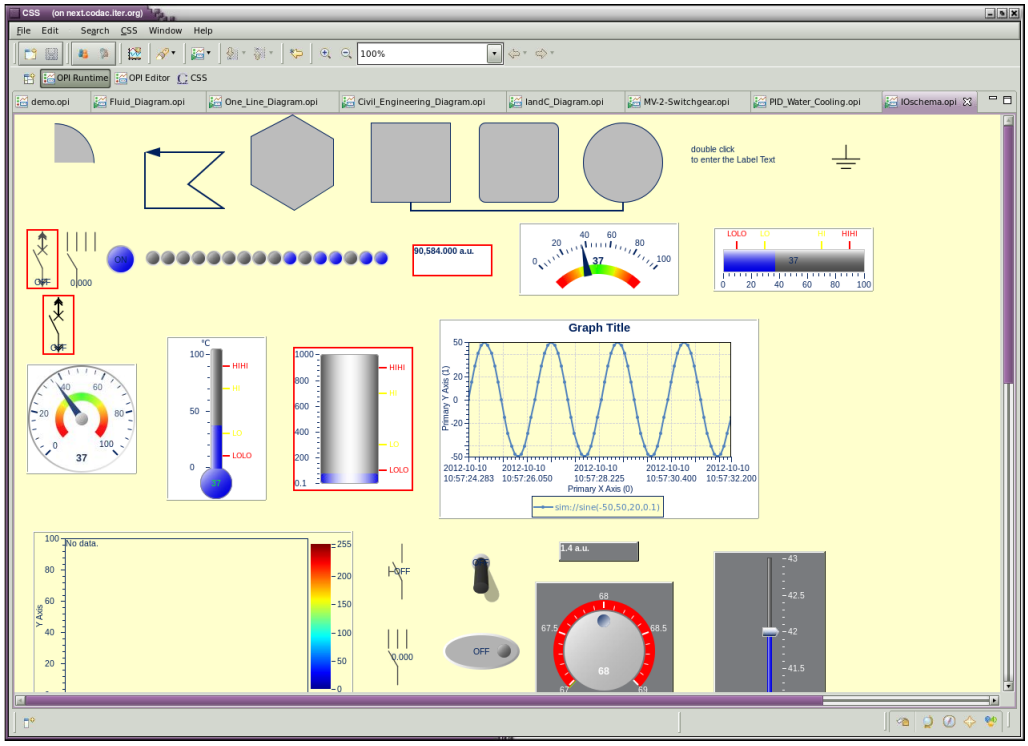
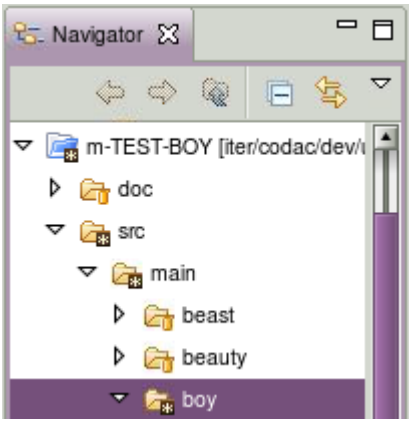


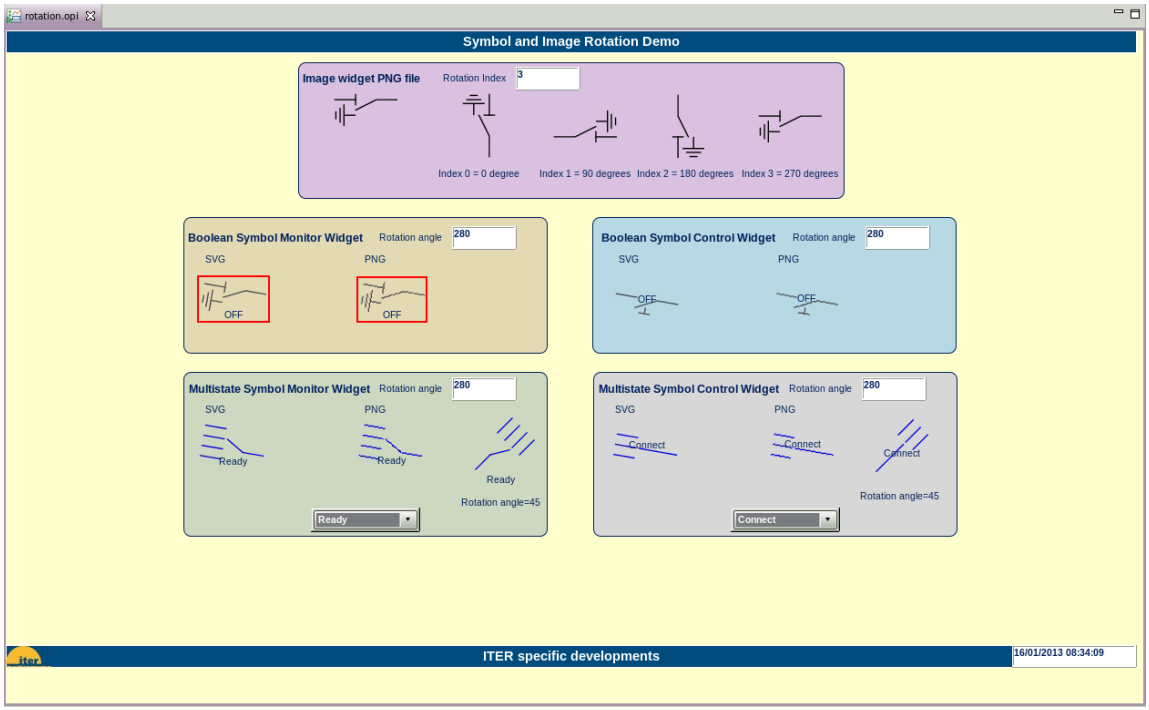
6. The output for the electrical use case should be:

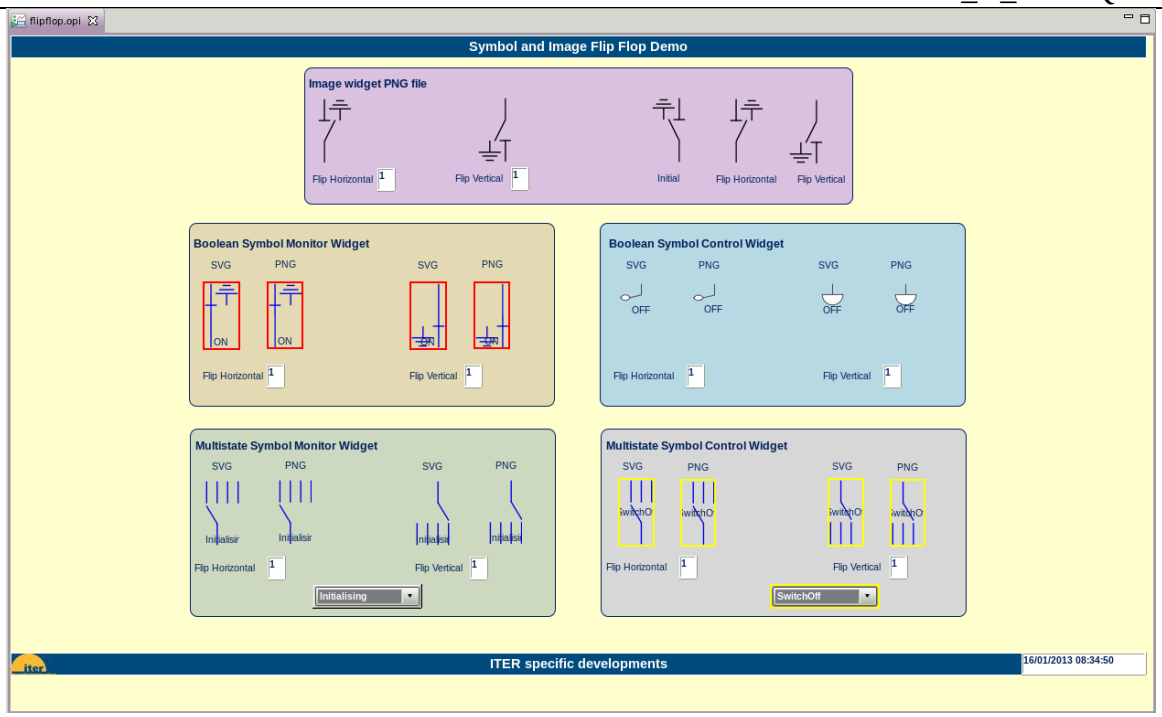


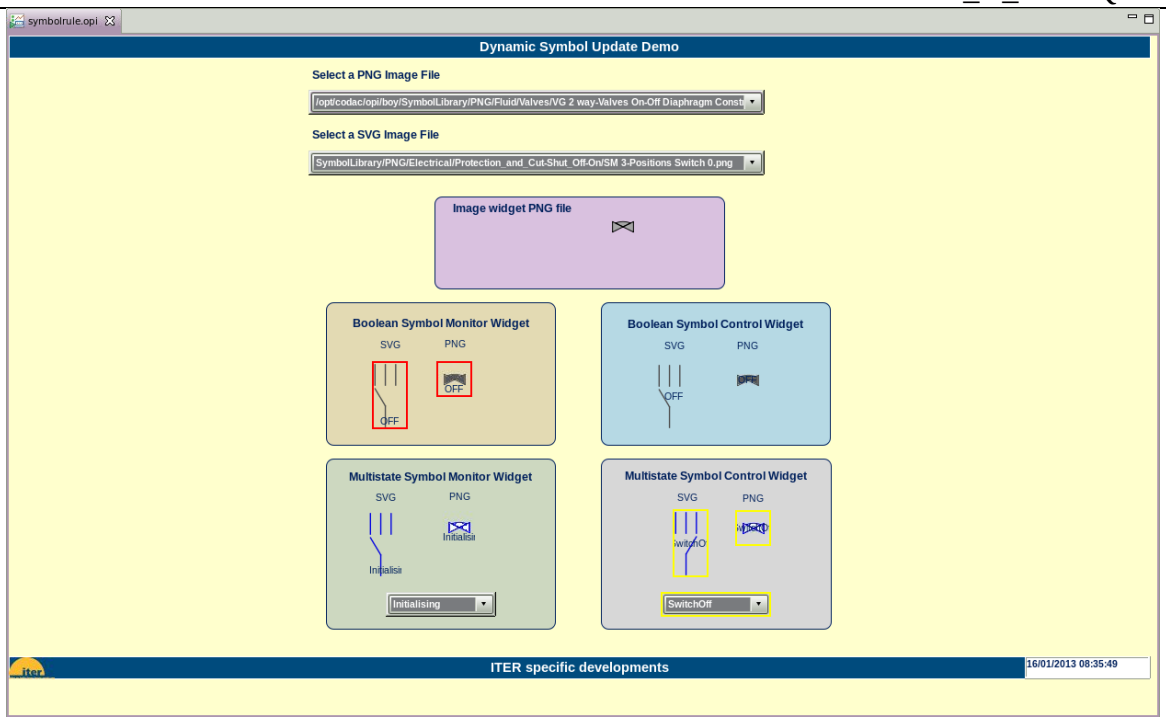
7. The output for the PID use case should be:

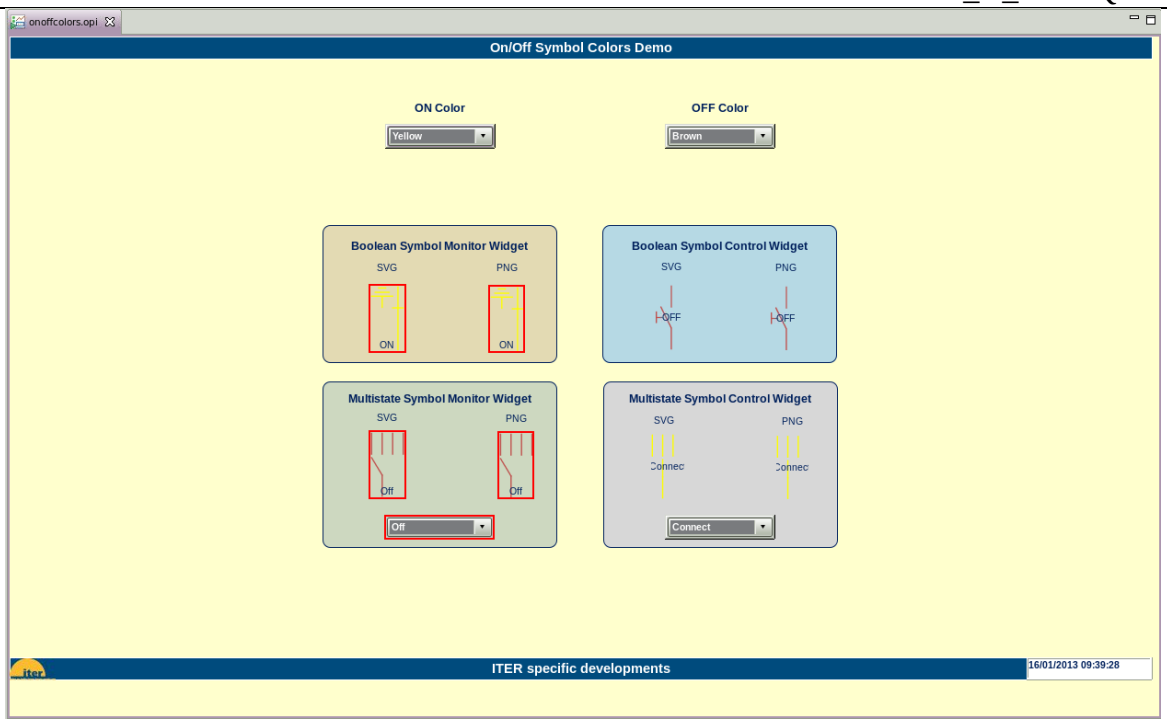


	8. The output for CODAC BOY Schema should be: 
	3.4.8 SMB02 - Symbol and Image widgets rotation
Prerequisite	1. Demo IOC running 2. Development environment started
Test Cases	1. Positive confirmation of the rotation of the Boolean and Multistate Symbol and Image widgets
Procedure	In CSS, run the Operator Interfaces demonstrating the rotation of the image: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy  2. From the Navigator View, double-click on rotation.opi file. In order to have a better view, switch to OPI Runtime Perspective 3. Check that the Image widget PNG file rotates from 0 to 270 degrees by 90 degrees. A screen

	shot of expected positions is given to help the verification. Then check that the Boolean Symbol widgets - Monitor and Control – in SVG and PNG format, rotate by 20 degrees. Finally do the same verification for the Multistate Symbol widgets. 4. Switch back to OPI Editor Perspective and close the rotation BOY OPI	
Pass Criteria	2. The rotation demo BOY screen looks like that: 	
	3.4.9 SMB03 - Symbol and Image widgets flip/flop	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the horizontal and vertical flip of the Boolean and Multistate Symbol and Image widgets	
Procedure	In CSS, run the Operator Interfaces demonstrating the horizontal and vertical flip of the image: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy 2. From the Navigator View, double-click on flipflop.opi file. In order to have a better view, switch to OPI Runtime Perspective 3. Check that the Image widget PNG file flips horizontally and vertically. A screen shot of expected positions is given to help the verification. Then do the same verification for Boolean Symbol widgets - Monitor and Control – in SVG and PNG format and for the Multistate Symbol widgets 4. Switch back to OPI Editor Perspective and close the OPI screen	
Pass	2. The flip-flop demo BOY screen looks like that:	

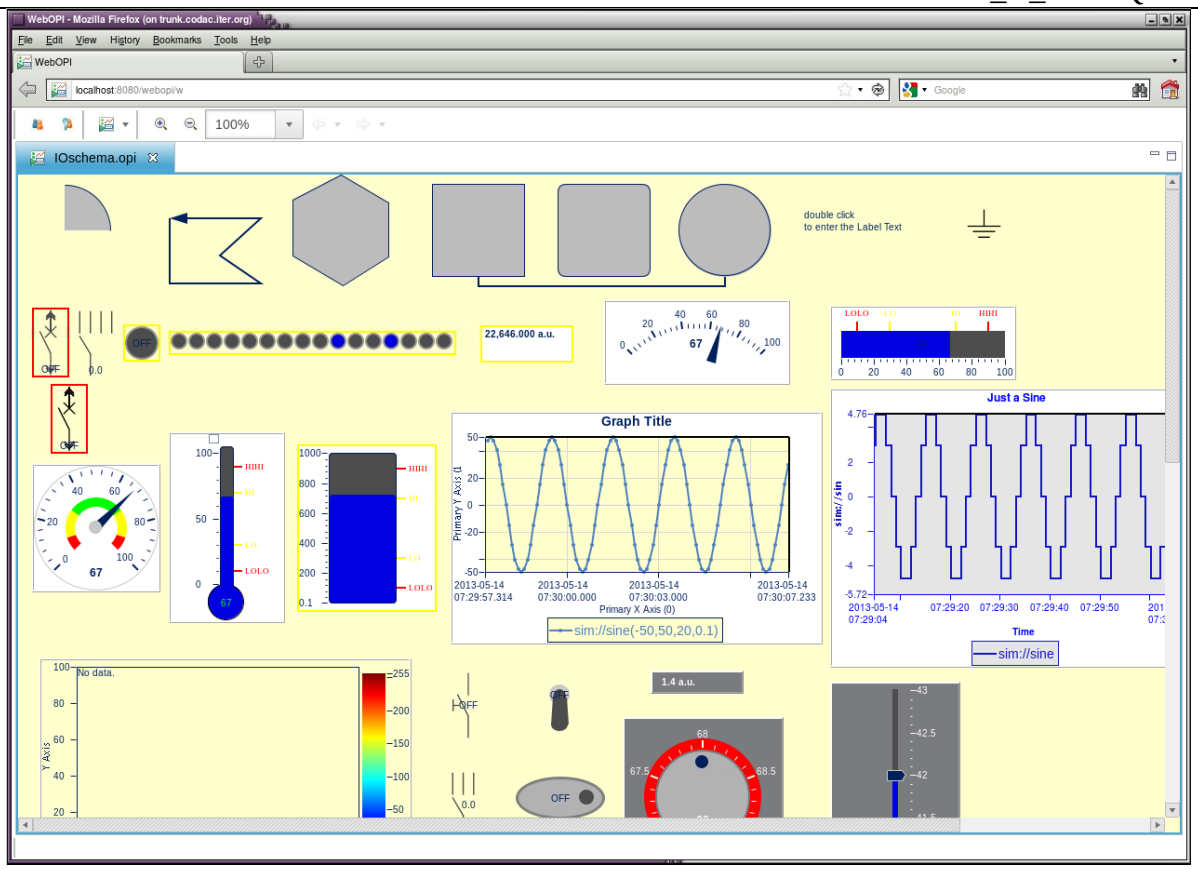
Criteria		
	3.4.10 SMB04 - Symbol and Image File Selection	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the dynamic change of the image file for the Boolean and Multistate Symbol and Image widgets	
Procedure	In CSS, run the Operator Interfaces demonstrating the dynamic change of the image: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy 2. From the Navigator View, double-click on symbolrule.opi file. In order to have a better view, switch to OPI Runtime Perspective 3. Change the PNG Image file using the Combo Box and check that all widgets displaying a PNG file are updated. Do the same with a SVG Image file 4. Switch back to OPI Editor Perspective and close the OPI screen	
Pass Criteria	2. The selection of an PNG and SVG Image file in demo BOY screen looks like that:	

		
	3.4.11 SMB05 - On/Off Symbol Color	
Prerequisite	1. Demo IOC running 2. Development environment started	
Test Cases	1. Positive confirmation of the dynamic change of the On and Off color for the Boolean and Multistate Symbol and Image widgets	
Procedure	In CSS, run the Operator Interfaces demonstrating the dynamic change of the ON/OFF color of the image: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy 2. From the Navigator View, double-click on onoffcolors.opi file. In order to have a better view, switch to OPI Runtime Perspective 3. Change the ON Color using the Combo Box and check that all symbol widgets displaying an ON value (or index > 0 for a multistate) are updated. Do the same with the Off Color 4. Switch back to OPI Editor Perspective and close the OPI screen	
Pass Criteria	2. The selection of and On and Off color in demo BOY screen looks like that:	



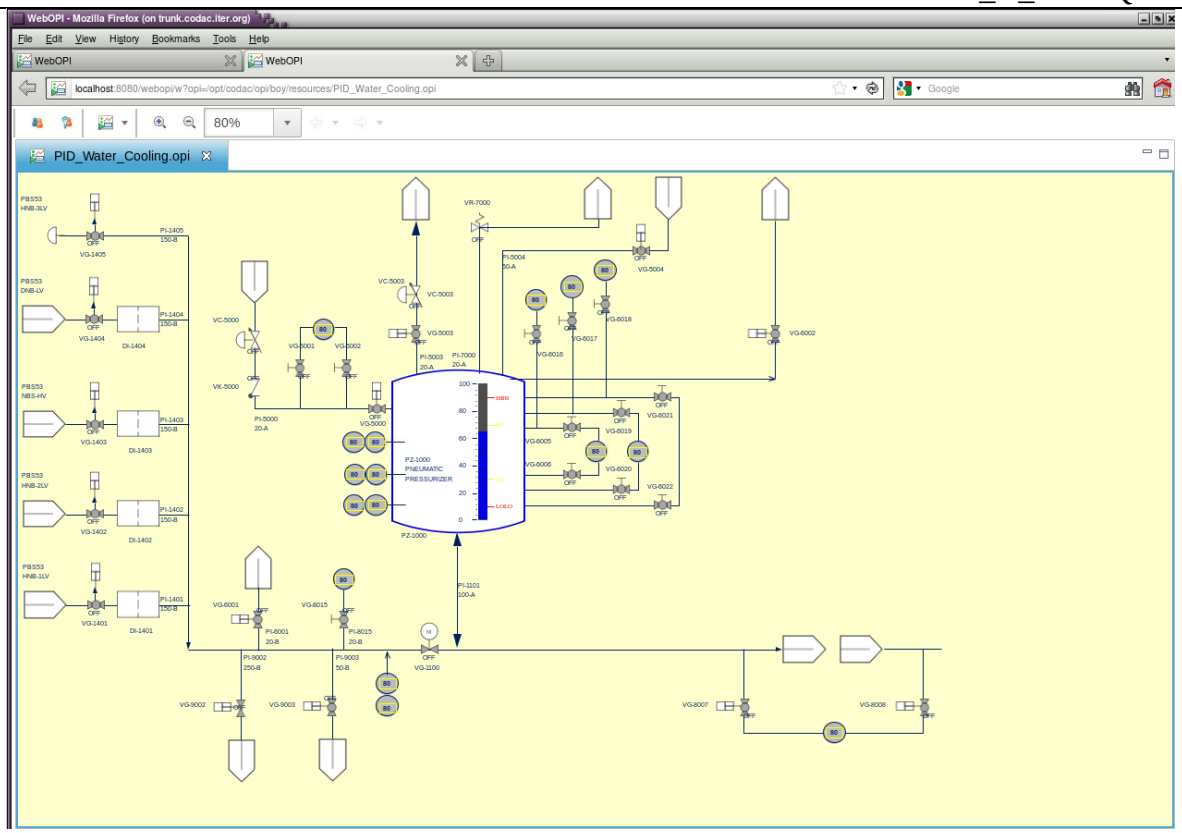
3.4.12 WEB01 - Web Operator Interface

Prerequisite	1. Demo IOC running
Test Cases	1. Positive confirmation of the demo opi displayed with a web browser
Procedure	In the previous Linux console, start the web browser and specify the default URL of WebOPI: 1.$ firefox http://localhost:8080/webopi &
Pass Criteria	1. After few seconds the IOschema opi containing the definition of all the widgets is displayed:



3.4.13 WEB02 - Open an opi file with a web browser

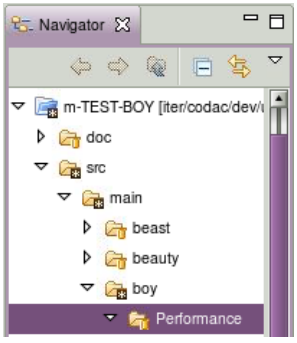
Prerequisite	1. Demo IOC running
Test Cases	1. Positive confirmation of the display of an opi file on web browser
Procedure	In the previous Linux console, start the web browser with the following URL: 1.$ firefox http://localhost:8080/webopi/w?opi=/opt/codac/opi/boy/resources/PID_Water_Cooling.opi& 2. Change the Zoom factor using the combo box or the + and – buttons, in order to see all the symbols in the page:  3. Close the web browser
Pass Criteria	1. After few seconds the PID_Water_Cooling.opi is displayed:



3.4.14 PRF01 - Performance of Symbol Widgets

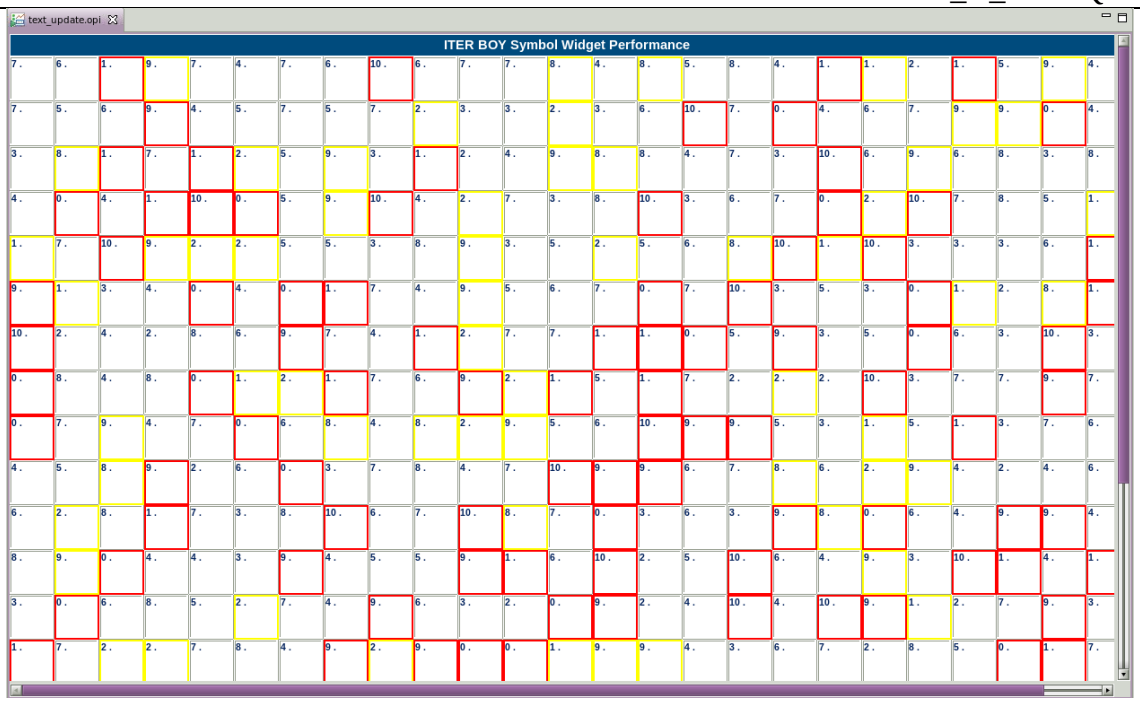
Prerequisite	1. Performance IOCs downloaded from SVN
	2. You can close the previous EPICS database
	`epics> exit`
	3. Start the EPICS IOC Performance Database:
	`$ softIoc src/main/epics/SharedTemplateApp/Db/all-A-IOCs-start.cmd`

Test Cases	1. Positive confirmation of the loading of more than 250 Boolean and Multistate Symbol widgets
Procedure	In CSS, run the Operator Interfaces demonstrating the performance of the symbol widgets: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy/Performance

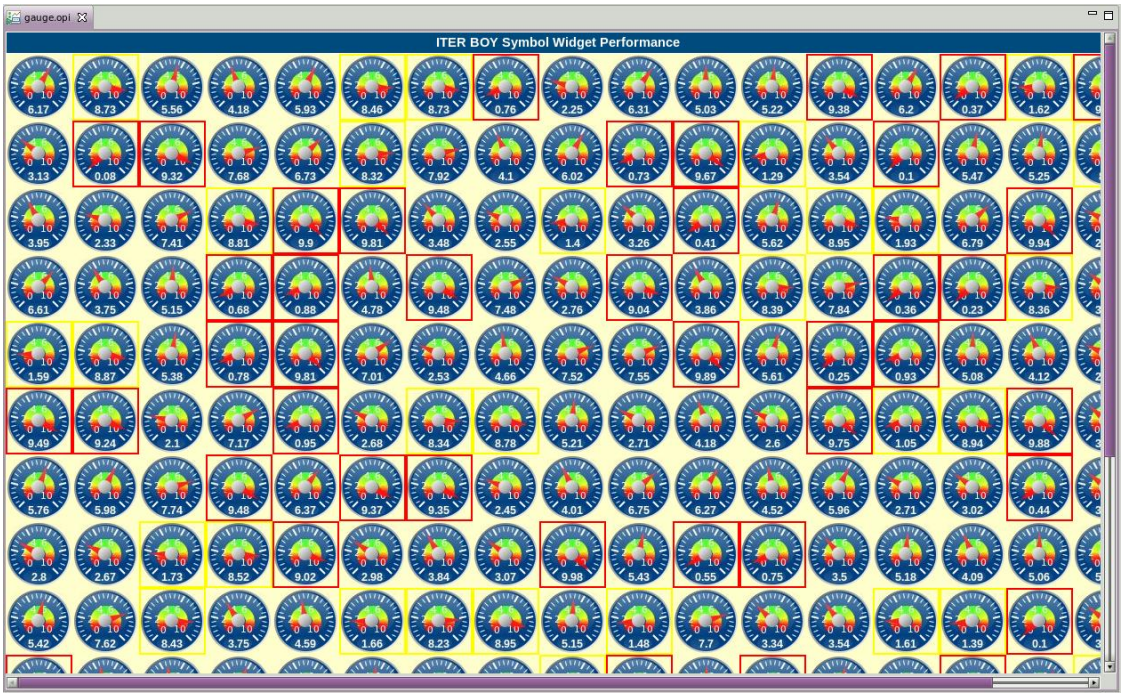


	2. From the Navigator View, right-click first on text_update.opi file and select Open With -> OPI Editor. State how fast the Editor is able to load more than 500 Text Update widgets. Then run the OPI using the button from the tool bar  . Scan rate = 0.1 second.
	Close CSS runtime instance, go back to OPI Editor and close text_update.opi
	3. From the Navigator View, right-click first on gauge.opi file and select Open With -> OPI Editor. State how fast the Editor is able to load more than 250 graphical widgets. Then run the OPI using the button from the tool bar  . Scan rate = 0.1 second.
	Close CSS runtime instance and go back to OPI Editor to close gauge.opi. If you fail to close the Operator Interface which is too busy, stop the IOC:
	`epics> exit`
	4. From the Navigator View, right-click first on boolean_symbol.opi file and select Open With -> OPI Editor. State how fast the Editor is able to load more than 250 Boolean Symbol Image widgets. Then run the OPI using the button from the tool bar  . Scan rate = 0.1 second.
	If you have stopped the IOC in the previous step, you need to restart it:
	`$ softIoc src/main/epics/SharedTemplateApp/Db/rndmIOC-A-start.cmd`
	Close CSS runtime instance and go back to OPI Editor to close boolean_symbol.opi. If you fail to close the Operator Interface which is too busy, stop the IOC:
	`epics> exit`
	5. From the Navigator View, right-click first on multistate_symbol.opi file and select Open With -> OPI Editor. State how fast the Editor is able to load more than 250 graphical widgets. Then run the OPI using the button from the tool bar  . Scan rate = 0.1 second.
	If you have stopped the IOC in the previous step, you need to restart it:
	`$ softIoc src/main/epics/SharedTemplateApp/Db/rndmIOC-A-start.cmd`
	Close CSS runtime instance and go back to OPI Editor to close multistate_symbol.opi.	

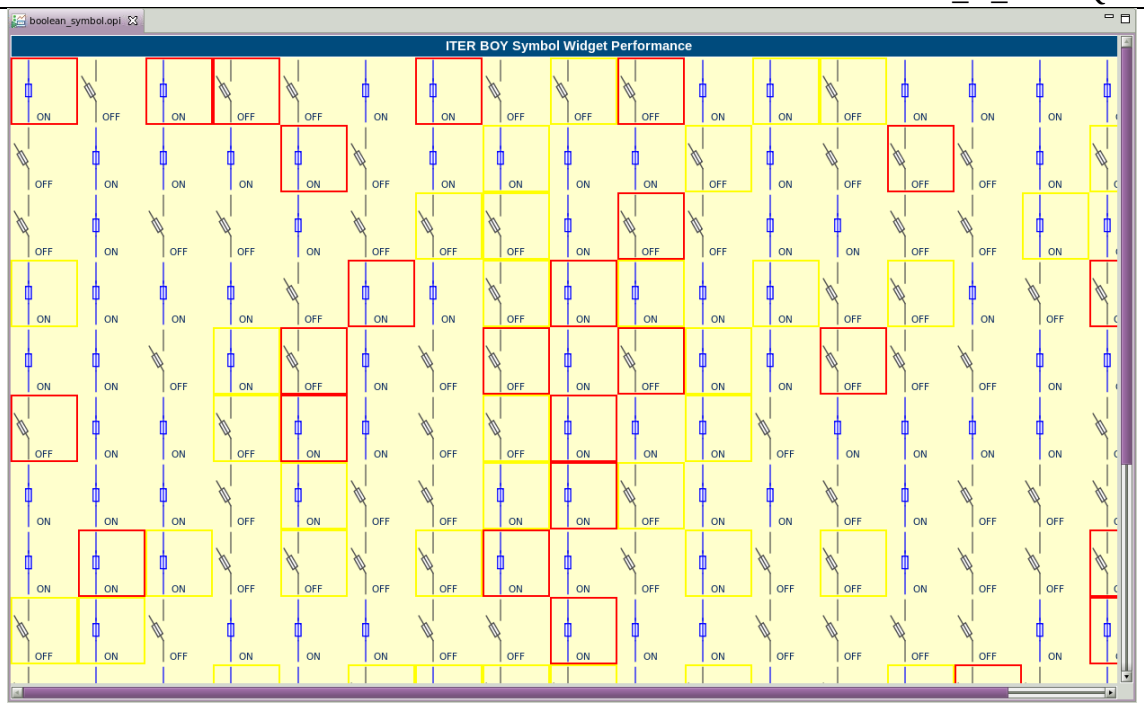
Pass Criteria	2. Text Update widgets used to displayed PV values at 10Hz:	



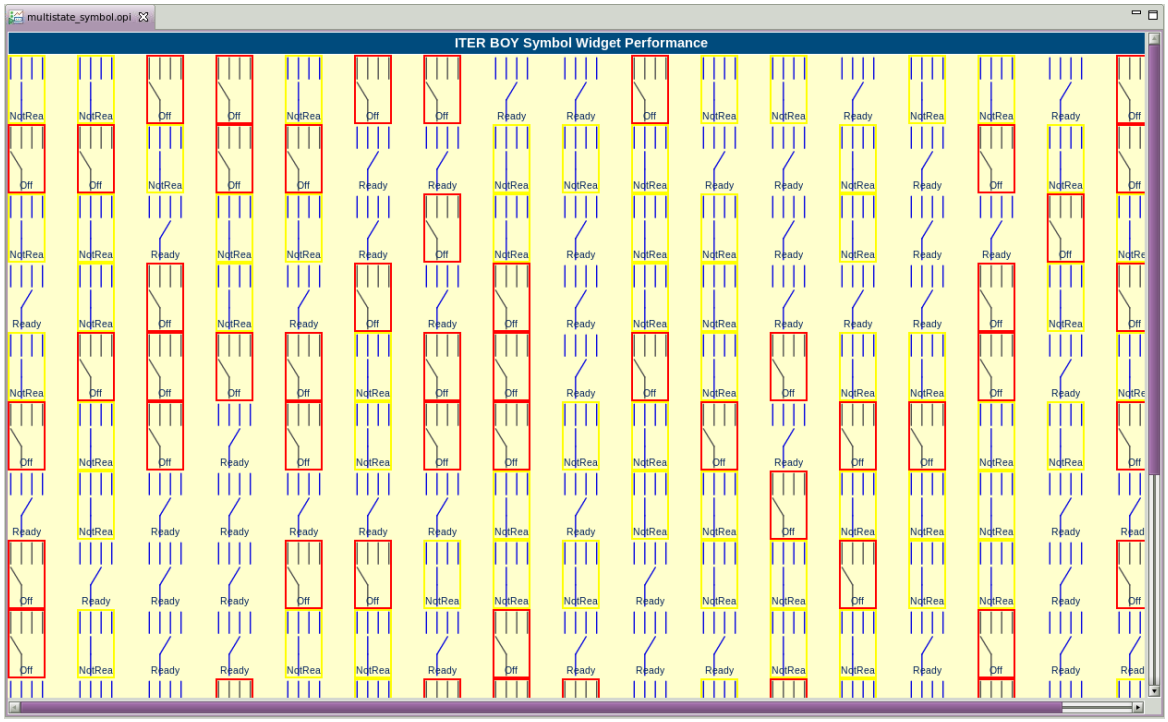
3. The time to load, connect to EPICS PV and animate the graphical widget is definitively longer:



3. The label should be displayed first then the symbol image after some seconds:

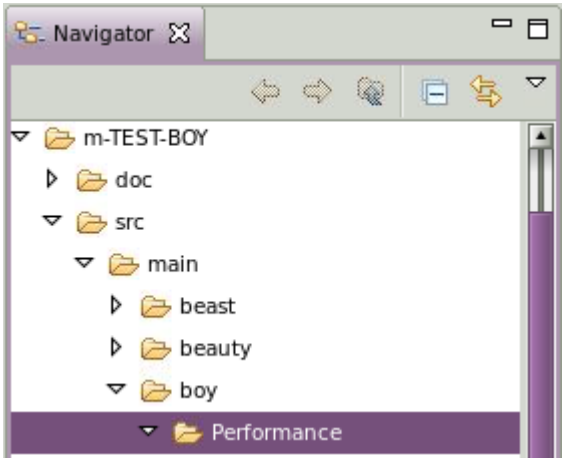


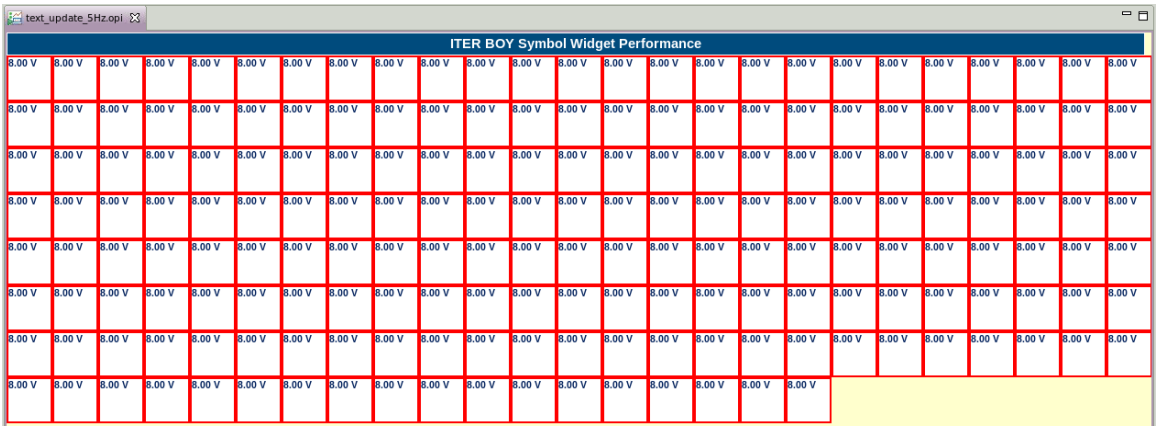
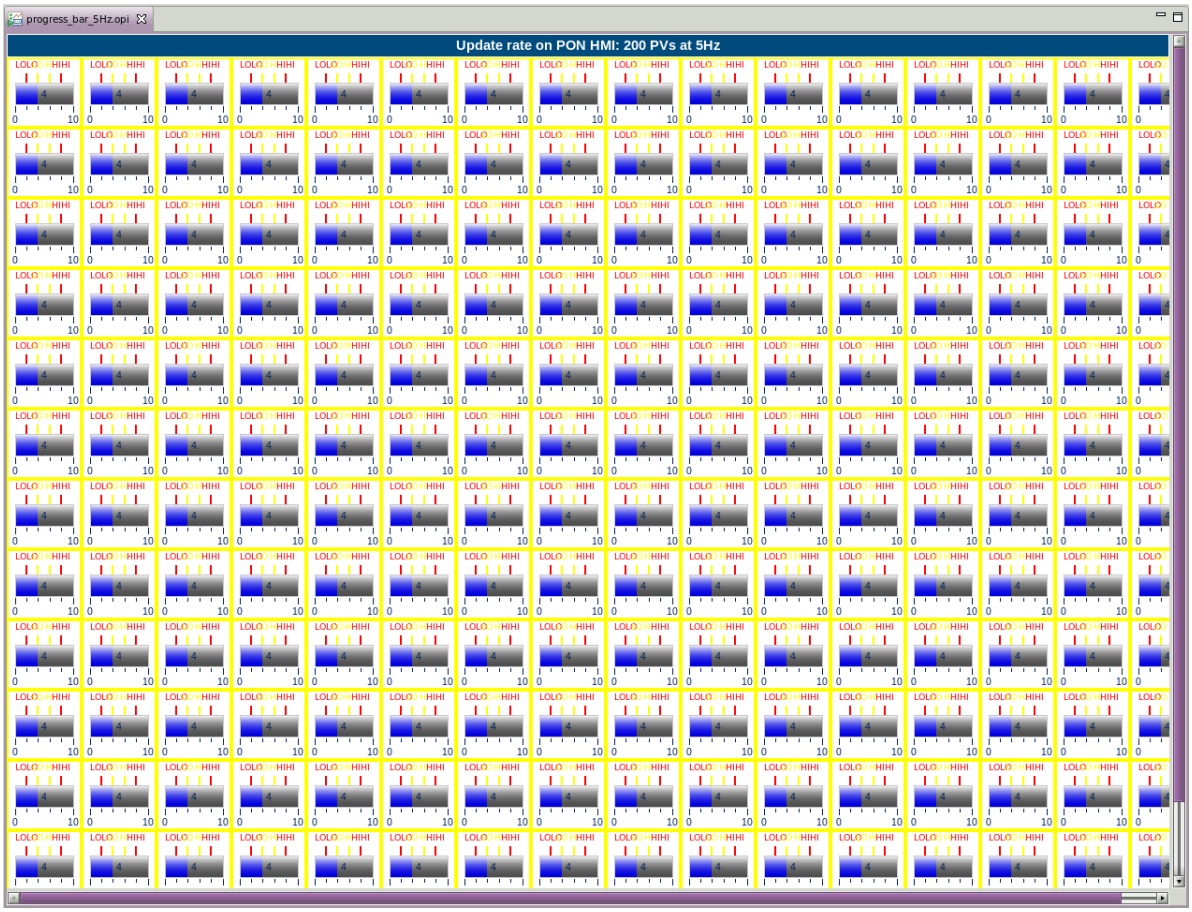
4. Finally the Multistate symbol widgets should be animated at 10Hz from Off state, NotReady state, Initialising state and Ready state:

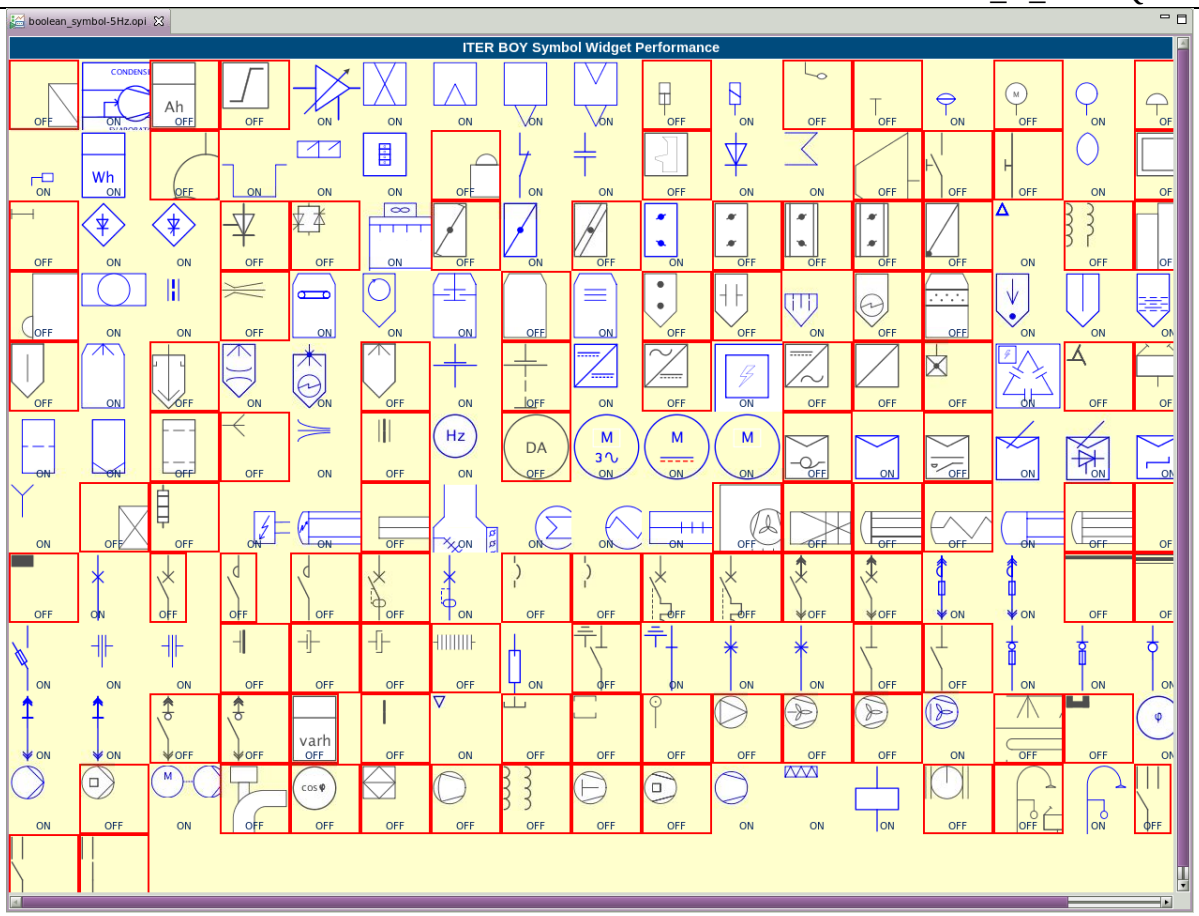


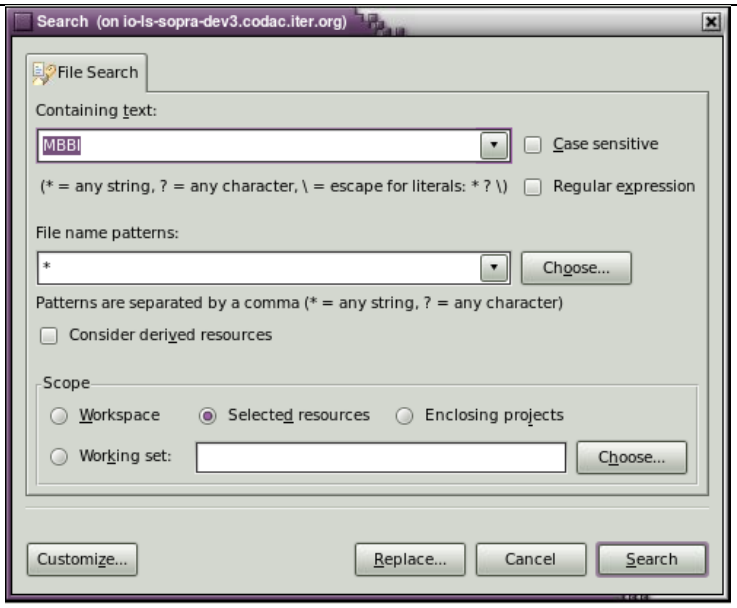
3.4.15 PRF02 - Update rate on PON HMI: 200 PVs at 5Hz

Prerequisite	1. Performance IOCs running If you need to restart them: `$ softIoc src/main/epics/SharedTemplateApp/Db/all-A-IOCs-start.cmd`

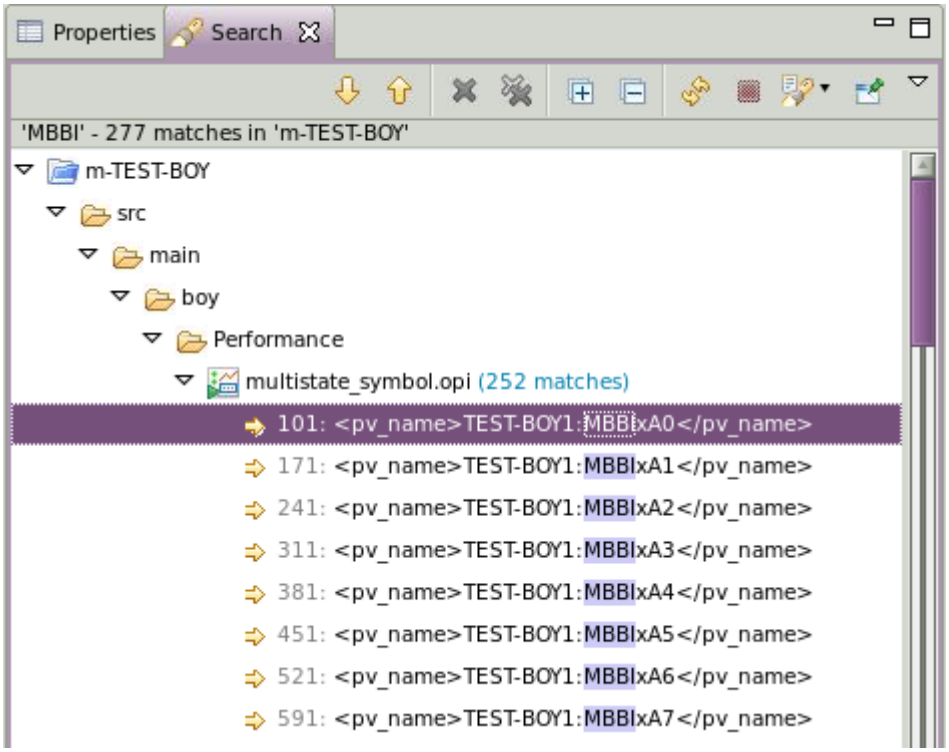
Test Cases	1. Positive confirmation of ITER control system main parameters - 4.6 Update rate on HMI screen (PON) = 200 PV at 5 Hz (ITER_D_6M58M9 v1.3)	
Procedure	In CSS, run the Operator Interfaces demonstrating the performance of the symbol widgets: 1. In the Navigator View, browse m-TEST-BOY/src/main/boy/Performance  2. From the Navigator View, right-click first on text_update_5Hz.opi file:  And select Open With -> OPI Editor. State how fast the Editor is able to load more than 200 Text Update widgets. Then run the OPI using the button from the tool bar  Close CSS runtime instance and go back to OPI Editor to close progress_bar_5Hz.opi. If you fail to close the Operator Interface which is too busy, stop the IOC: `epics> exit` 3. From the Navigator View, right-click first on progress_bar_5Hz.opi file:  And select Open With -> OPI Editor. State how fast the Editor is able to load more than 200 Progress Bar widgets. Then run the OPI using the button from the tool bar  . Make a right-click on the Operator screen to switch to Full Screen in order to display and animate a maximum of widgets. Exit the Full Screen mode by pressing F11. Close CSS runtime instance and go back to OPI Editor to close progress_bar_5Hz.opi. If you fail to close the Operator Interface which is too busy, stop the IOC: `epics> exit` 4. From the Navigator View, right-click first on Boolean_symbol_5Hz.opi file:  And select Open With -> OPI Editor. State how fast the Editor is able to load more than 200 different symbols. Then run the OPI using the button from the tool bar  . Make a right-click on the Operator screen to switch to Full Screen in order to display and animate a maximum of widgets. Exit the Full Screen mode by pressing F11. Close CSS runtime instance and go back to OPI	

	Editor to close progress_bar_5Hz.opi. If you fail to close the Operator Interface which is too busy, stop the IOC: `epics> exit` 5. Stop the IOC (if not already done): `epics> exit`
Pass Criteria	2. OPI for 200 PVs at 5Hz using exclusively Text Update widgets. The 200 ramps from 0 to 10 are displayed smoothly without missing any increment:  3. OPI for 200 PVs at 5Hz using exclusively Progress bar widgets – one of the most demanding widgets. The 200 ramps from 0 to 10 are not displayed smoothly:  4. Boolean information at 5Hz is displayed smoothly using symbol images:

		
	3.4.16 EDT06 - Eclipse Search in OPI file	
Prerequisite	1. Development environment started	
Test Cases	1. Positive confirmation of the search feature in OPI files	
Procedure	In CSS, run the Operator Interfaces demonstrating the performance of the symbol widgets: 1. In the Navigator View, click on m-TEST-BOY 2. From the menu bar, Search -> Search… (Ctrl+H) 3. Specify “MBBI” as searched text and for the Scope, click on “Selected resources” in order to restrict the search to m-TEST-BOY selected previously. Click on Search	

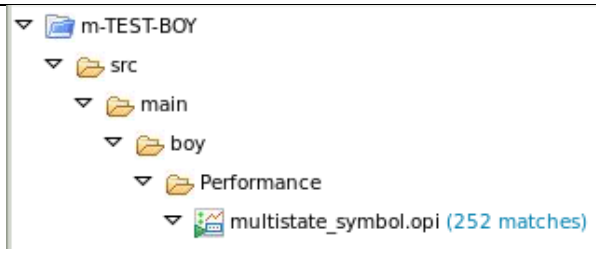
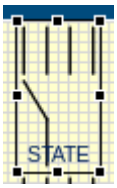
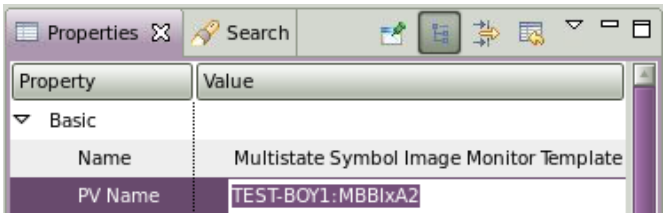


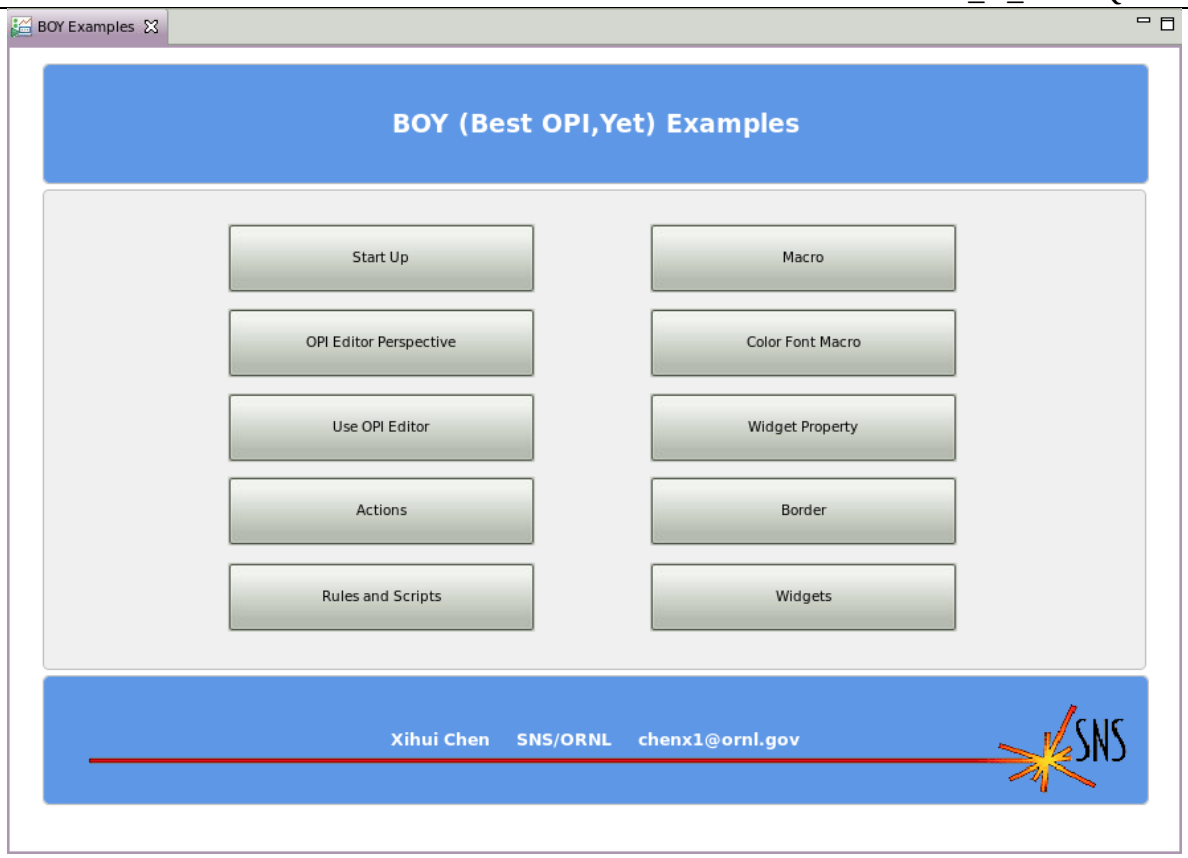
4. From the Search View, double-click on one of the match and the corresponding OPI will be opened and the widget selected



5. Close CSS

Pass Criteria	4. If the following match is selected for example:  The OPI is open corresponding opi is opened:

	 And the widget displaying TEST-BOY1:MBBIxA2 is selected:  To check that the PV name corresponds to the match, go to the Property view: 	
	3.4.17 RNT02 - Start an OPI from command line	
Prerequisite	None	
Test Cases	1. Positive confirmation of the OPI started from the command line	
Procedure	In the previous Linux console: 1. $ css --launcher.openFile "/BOY Examples/main.opi"& 2. Confirm the selected Workspace 3. Check that the main screen of BOY Examples is displayed correctly 4. Close CSS	
Pass Criteria	3. BOY Examples main OPI should be opened in CSS:	



3.4.18 LOG01 – LOG: Look for any SEVERE message

Prerequisite	1. None
Test Cases	1. No SEVERE alert in the CSS log files
Procedure	In a Linux console, check the log of CSS general services: 1. $ grep -r 'SEVERE' /var/opt/codac/css/ Now check the log of the services started manually for the demo applications: 2. $ grep -r 'SEVERE' ~/.css/
Pass Criteria	1 - 2. No SEVERE messages except for: `~/.css/css/console.log:<date> SEVERE [Thread 1] org.csstudio.logging.PluginLogListener (logging) - Invalid preference page path: XML Syntax`

To terminate the tests, stop the slow IOC and close css:

1. $ epics> exit

2. Close CSS using the menu File -> Exit

3.5 Component Test Log

	3.5.1 EDT01 - Operator Interface Examples	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.2 EDT02 - Operator Interface Edition	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.3 RNT01 - Operator Interface Run	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.4 EDT03 - Data Browser widget	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.5 EDT04 - Vertical and Enabled Tabbed Container	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.6 EDT05 - Setpoint adjustment	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		

	3.5.7 SMB01 - CODAC Standard Symbol Lib	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.8 SMB02 - Symbol and Image widgets rotation	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.9 SMB03 - Symbol and Image widgets flip/flop	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.10 SMB04 - Symbol and Image File Selection	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.11 SMB05 - On/Off Symbol Color	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.12 WEB01 - Web Operator Interface	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		

	3.5.13 WEB02 - Open an opi file with a web browser	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.14 PRF01 - Performance of Symbol Widgets	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.15 PRF02 - Update rate on PON HMI: 200 PVs at 5Hz	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.16 EDT06 - Eclipse Search in OPI file	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.17 RNT02 - Start an OPI from command line	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		

Software Test Plan Checklist

For Assessment of:	
Agency Name	
Project Name	
Document Name	
Date	

Criteria	Yes / No / NA
DOCUMENT STANDARDS COMPLIANCE	
1 Have standards/guidelines been identified to define the work product?	
2 Does the work product format conform to the specified standard/guideline (Template)?	
3 Has the project submitted any request for deviations or waivers to the defined work product?	
4 Have the following areas been addressed completely:	
4a Approval authority?	
4b Revision approval?	
4c Revision control?	
TECHNICAL REFERENCE	
5 Is there evidence that the work product was reviewed by all stakeholders?	
6 Have acceptance criteria been established for the work product?	
7 Does the work product have a clearly defined purpose and scope?	
8 Are references to policies, directives, procedures, standards, and terminology provided?	
9 Does the work product identify any and all constraints/limitations?	
S/W TEST PLAN CONTENTS	
10 Does the S/W Test Plan address the following required information:	
10a Test levels?	
10b Test types (e.g., unit testing, software integration testing, systems integration testing, end-to-end testing, acceptance testing, regression testing)?	
10c Test classes?	
10d General test conditions?	
10e Test progression?	
10f Data recording, reduction, and analysis?	
10g Test coverage (breadth and depth) or other methods for ensuring sufficiency of testing?	
10h Planned tests, including items and their identifiers?	
10i Test schedules, Requirements traceability (or verification matrix)?	

Criteria	Yes / No / NA
10j Qualification testing environment, site, personnel, and participating organizations?	
11 Does the S/W Test Plan identify the environmental exposure as well as requirements for comprehensive, functional, aliveness, end-to-end, and mission simulation testing?	
12 Does the S/W Test Plan provide a System Overview that describes the unique complexities of the system?	
13 Does the S/W Test Plan address user guide, operations / maintenance validation?	
16 Does the S/W Test Plan identify any elements that will not be tested according to the test plan (e.g., externally developed software)?	
17 Does the S/W Test Plan address software architecture in terms of which software components will be based on heritage and which will be mostly or entirely new developments?	
18 Does the S/W Test Plan identify any software reuse? If so, is the extent of reuse or the anticipated modification described?	
S/W TEST ENVIRONMENT	
19 Does the S/W Test Plan include a figure of each system test environment? If so, does it reflect the system hardware approach, simulators, and special development?	
20 Does the S/W Test Plan identify specific test hardware and simulators for each external interface?	
TEST TOOLS	
21 Does the S/W Test Plan address test execution tools?	
TEST PROBLEM REPORTING & CORRECTIVE ACTION	
22 Does the S/W Test Plan provide a description of the problem reporting system to be used by the test team to report problems and/or recommended changes cited during the test activities?	
TEST PROGRESS PLANNING & TRACKING	
23 Does the S/W Test Plan describe the routine test progress reporting approach?	
24 Does the S/W Test Plan describe the Build Test verification methodology? If so, does the description address build verification test level objectives, environment, roles & responsibilities, entry/exit criteria, general guidelines, build test planning, build test scenario development, build test procedure preparation & dry run, build test execution, reporting, and archiving?	