IDM UID
DSVQSE

VERSION CREATED ON / VERSION / STATUS
20 Jun 2013 / 1.1 / Signed

EXTERNAL REFERENCE

Report

CODAC Operator Tools - Test Plan

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

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 Tools - Test Plan (DSVQSE_v1_1)	v1.1	Signed	20 Jun 2013	Check for SEVERE messages in the log files
CODAC Operator Tools - Test Plan (DSVQSE_v1_0)	v1.0	Signed	31 May 2013	Test plan for CSS Display Tools, Diagnostic Tools, Utilities and SNL Editor
CODAC Operator Tools - Test Plan (DSVQSE_v0_0)	v0.0	In Work	29 May 2013	

CODAC Operator Tools

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

This document describes the tests that should be performed for CODAC Operator Tools 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 – Control System Studio - in order to be installed as part of CODAC Core System. These tests are in addition to the ones described in CODAC Operator Interface - Test Plan (ITER_D_9YL8QC), CODAC PON Archiving System - Test Plan (ITER_D_9KMNAD) or CODAC Alarm Handling System - Test Plan (ITER_D_9KBPN8). The current list of CSS Test Plans is available at https://user.iter.org/?uid=G7PEJT .

The Operator Tools is a set of tools to display EPICS Process Variable (PV) information, debug the JMS messages and the databases, make some diagnostics and edit / pre-compile a State Machine using SNL language.

1.2 Scope

The test items are:

- The operational version of CSS GUI,
- The data, including all the configuration data needed to run the Operator Tools,
- 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

CSS combines various control-system tools into a consistent environment.

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 Tools.

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

Operator Tools Introduction Component Test	**INT**
Test of the Operator Tools Welcome and Help pages	
Operator Display Tools Component Test	**DSP**
Test of Operator Tools Display	
Operator Diagnostic Tools Component Test	**DGC**
Test of Diagnostic Tools	
Operator Debugging Tools Component Test	**DBG**
Test of Debugging Tools	
Operator Utilities Component Test	**UTL**
Test of Utilities	
State Notation Language Component Test	**SNL**
Test of SNL Editor	
Auto completion Component Test	**ATC**
Test of PV Name auto completion	
Log Component Test	**LOG**
Check that no SEVERE alerts have been generated during the tests	

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 Tools is included the following unit:

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

3.1.2 Features to be tested

The main CODAC Operator Tools features to be tested are:

- CSS Graphical User Interface

3.1.3 Features not to be tested

The File and Edit options are not part of this test plan.

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 Tools.

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 INT01 - Welcome

Prerequisite	In a Linux console, clean the environment, 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-CSS
	…
	`A  m-TEST-CSS/src/main/databrowser`
	`A  m-TEST-CSS/src/main/databrowser/temp.plt`
	`A  m-TEST-CSS/src/main/beast`
	`A  m-TEST-CSS/src/main/beast/CWS-TCPH-beast.xml`
	`A  m-TEST-CSS/sdd.xml`
	`A  m-TEST-CSS/pom.xml`
	`Checked out revision xxxx.`
	4.$ cd m-TEST-CSS
	5.$ softIoc -s -d src/main/epics/CWS-TCPHApp/Db/PSH0-CWS-TCPH-Fxxx.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`
	`CWS-TCPH-Fxxx:TEMP`
	`CWS-TCPH-Fxxx:AMPL`
	`CWS-TCPH-Fxxx:FREQ`
	`CWS-TCPH-Fxxx:SIG-GENERATED`
	`CWS-TCPH-Fxxx:ACQ-STATE`
	`CWS-TCPH-Fxxx:SIG-STATE`
	`CWS-TCPH-Fxxx:SIMULATION`
	`CWS-TCPH-Fxxx:START-ACQ`
	`CWS-TCPH-Fxxx:TIME`
	`CWS-TCPH-Fxxx:SINE-CALC`
	`CWS-TCPH-Fxxx:GENERATOR-STATE`
	`CWS-TCPH-Fxxx:WAVEFORM-TYPE`
	`CWS-TCPH-Fxxx:TRACEMSG`
	`CWS-TCPH-Fxxx:VERSION`
	`epics>`

Test Cases	1. Positive confirmation of the Operator Tools introduction pages
Procedure	In another Linux console, start CSS: 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 “Overview” and check the brief description. Then click on the Overview content: the online should be opened to ITER CSS -> Getting Started section

5. From the Welcome toolbar, click on “First Steps”:

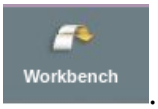
Check the short description of CSS main tools. Click on them to check the detailed information

6. From the Welcome toolbar, click on “What’s New” for the short description and click on the content to access to the Change Log section of the online Help. Check that ITER CSS change log describes the last Core System version

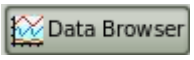
7. From the Welcome toolbar, click on “Samples”, check what are the samples provided

8. From the Welcome toolbar, click on “Tutorials”, and make the different short exercises. To switch from one exercise to another, click on Restore Welcome

9. Close the Online Help windows and Close the Welcome screen by clicking on Workbench icon:



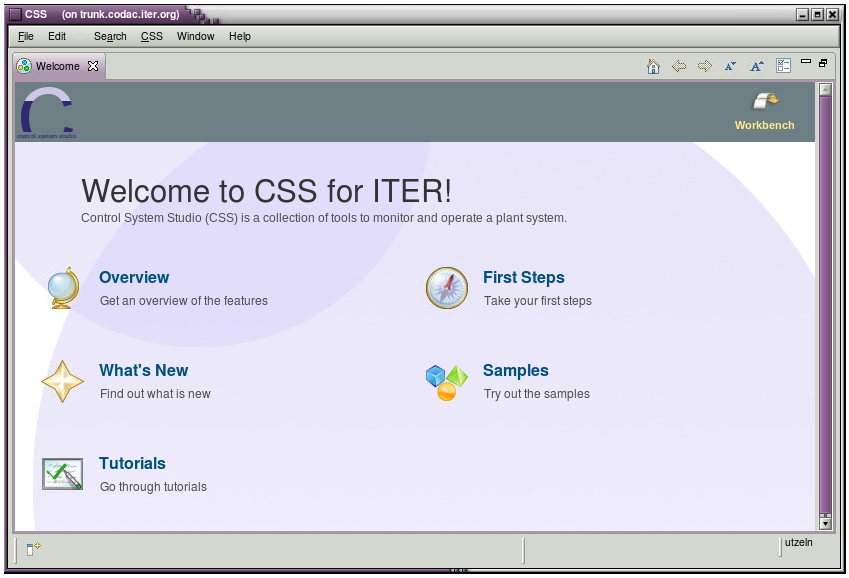
Close the plot file using the cross 

- do not save the plot configuration file and close the perspective via a right-click on the button 

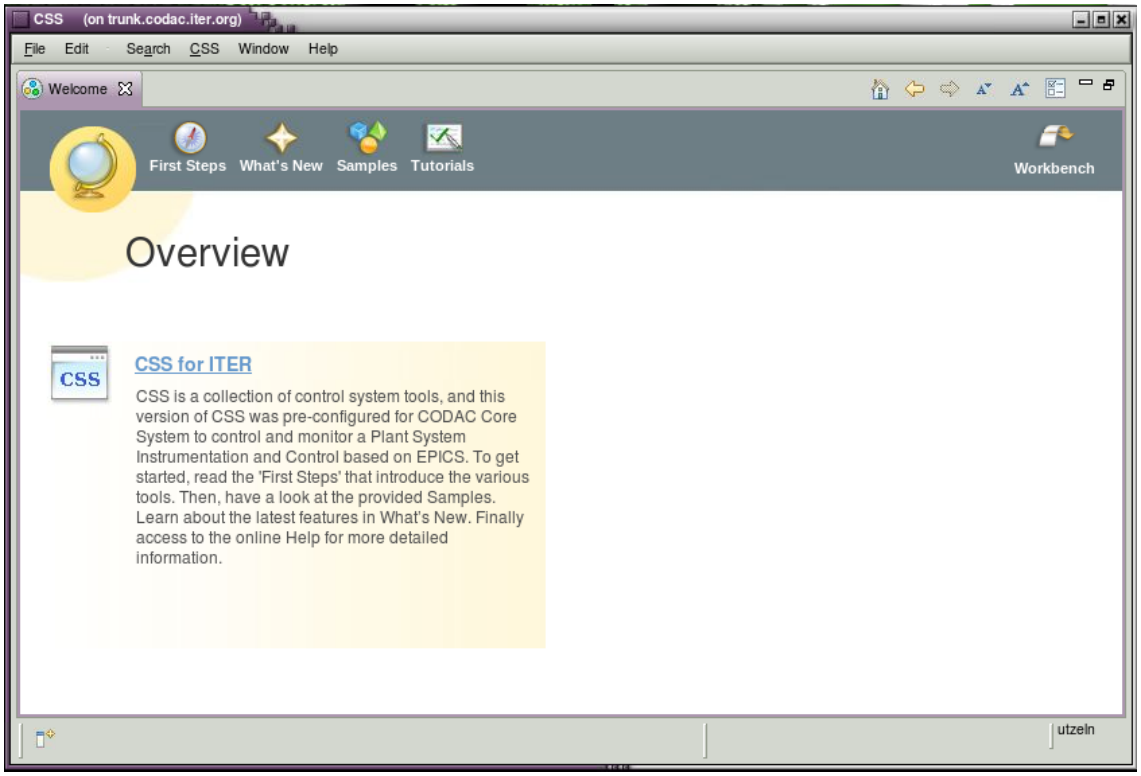
-> Close

Pass Criteria

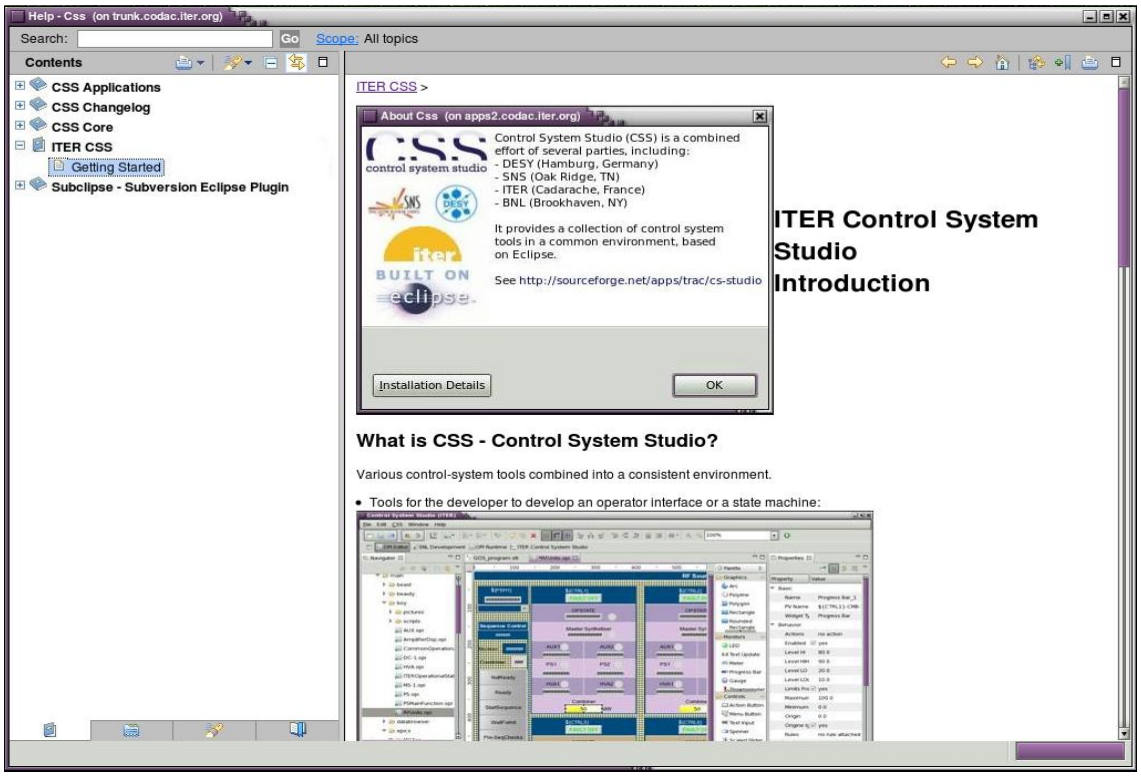
3. The main Welcome page is displayed:



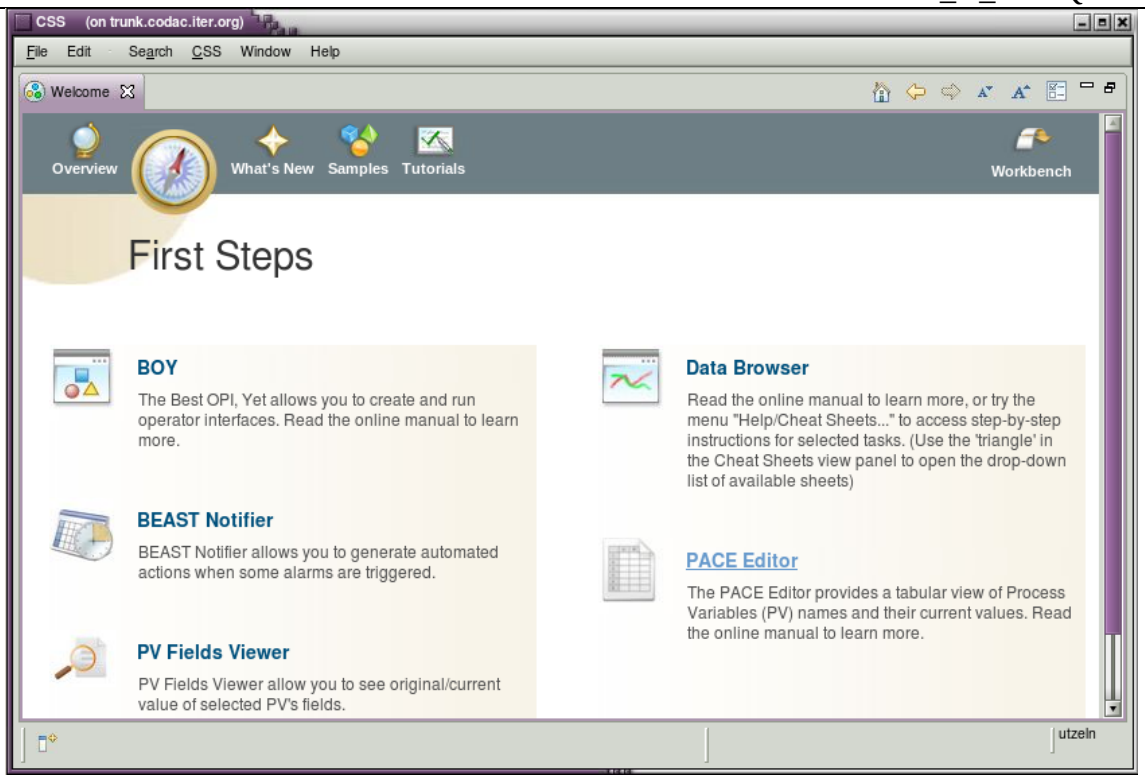
4. Overview page:



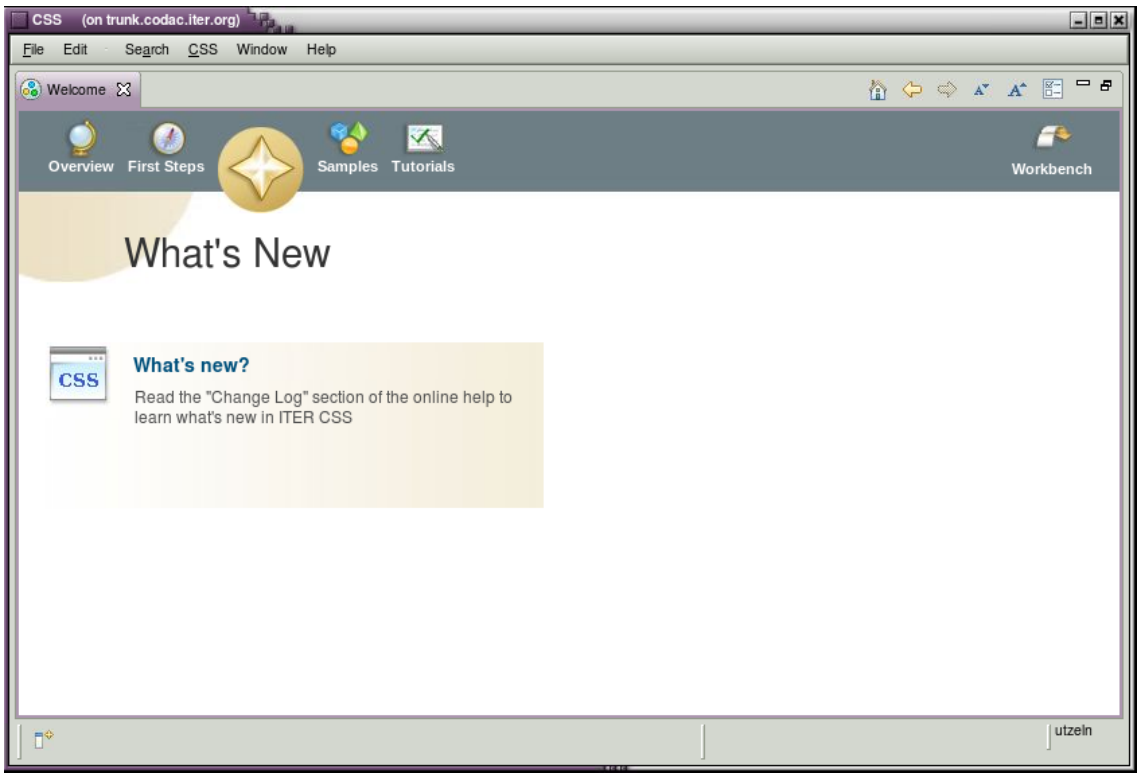
ITER CSS Getting Stated section in the online Help:



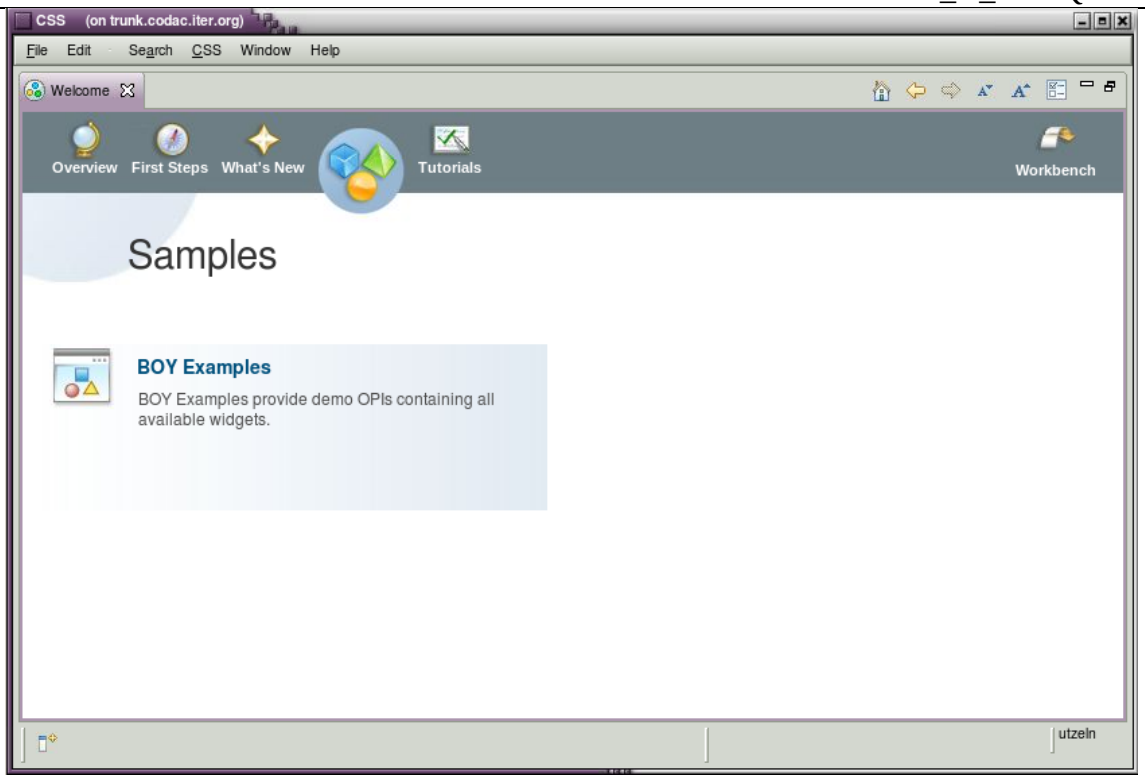
5. First Steps page:



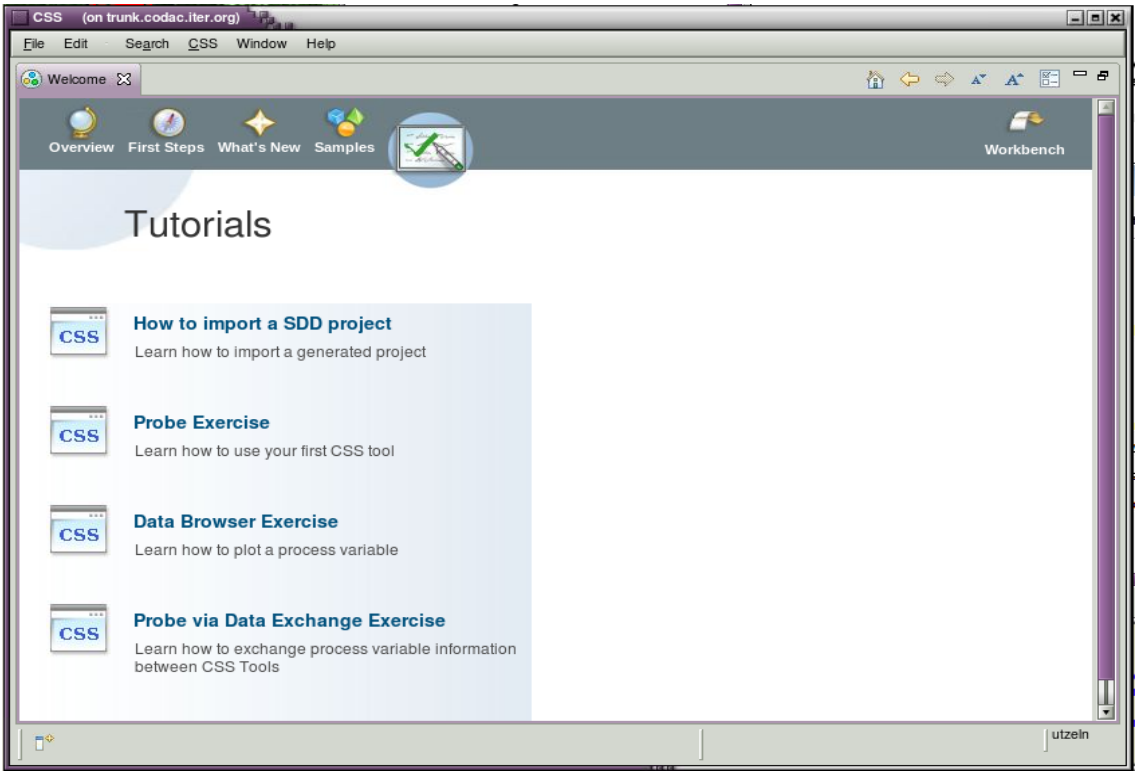
6. What's New page:



7. Samples page:

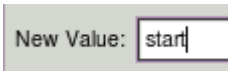


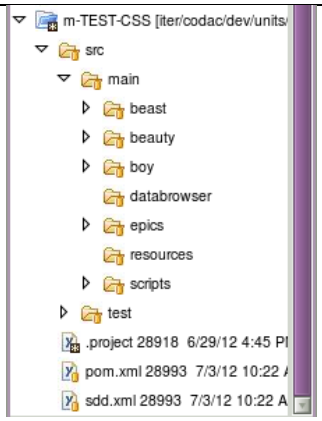
8. Tutorials page:



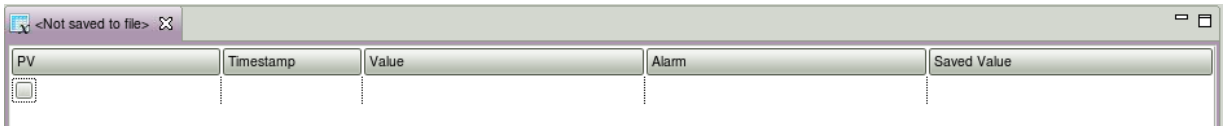
3.4.2 DSP01 - Display Tool: PV Table

Prerequisite	1. IOC running 2. Development environment started

Test Cases	1. Positive execution of PV Table	
Procedure	In CSS, import first the m-TEST-CSS project within the workspace: 1. From the Navigator View, right-click and select the option Import… and then General -> "Existing Projects into Workspace". Click on Next button. To select the root directory, click on Browse button, select m-TEST-CSS and click OK. To import the selected project, click on Finish Start the Operator Tool PV Table: 2. Menu CSS -> Display -> PV Table 3. In the PV field of the first line of the table, start typing CWS- and check what CSS auto-completion proposes 4. Click on the first matched PV name – `CWS-TCPH-Fxxx:ACQ-STATE` and click Enter to validate the selection 5. In the PV field of the second line of the table, start typing *GEN, select the matched PV Name `CWS-TCPH-Fxxx:GENERATOR-STATE` and click Enter to validate the selection 6. In the PV field of the third line of the table, start typing *WA, select the matched PV Name `CWS-TCPH-Fxxx:WAVEFORM-TYPE` and click Enter to validate the selection 7. In the PV field of the fourth line of the table, start typing *TEM, select the matched PV Name `CWS-TCPH-Fxxx:TEMP` and click Enter to validate the selection 8. Take a snapshot of the initial values by clicking on the button  9. In the PV field of the fifth line of the table, start typing *STAR, select the matched PV Name `CWS-TCPH-Fxxx:START-ACQ` and click Enter to validate the selection. Make a right-click on the PV Name `CWS-TCPH-Fxxx:START-ACQ` -> Process Variable -> Probe. In the Probe tool, click on Adjust button to modify the current PV value:  Into the new value "start":  Click Enter to validate the new value 10. Check the update of the PV Table once the acquisition is started 11. Save the PV Table configuration with <CTRL>+S. Select the parent folder as m-TEST-CSS/src/main/boy and enter the File name: pvTable. Click on OK to save the configuration 12. Close the PV Table Editor using the cross  . Close also the Probe tool  	
Pass Criteria	1. In the Navigator View, a new folder is added "m-TEST-CSS":	

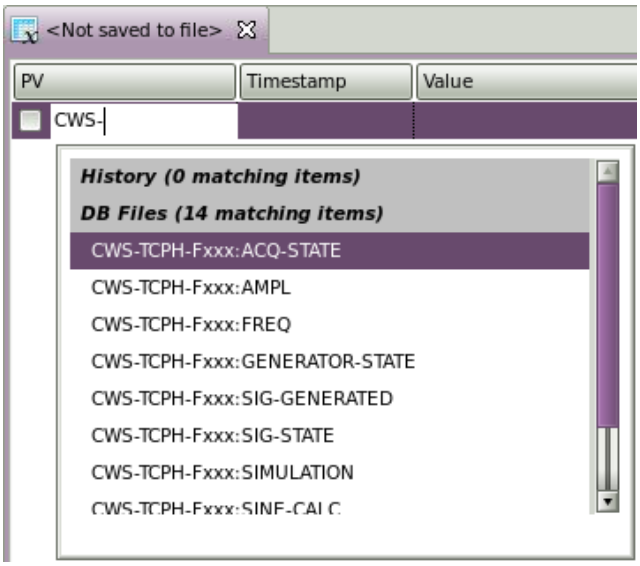


2. PV Table Editor should start:

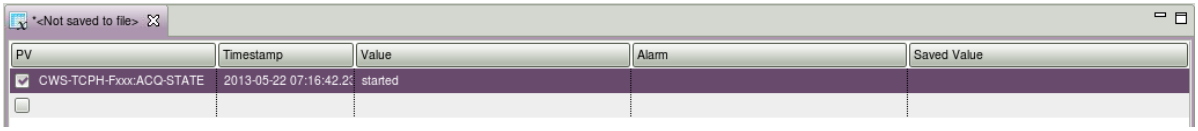


With its own tool bar:

3. CSS PV auto-completion looks for PV Names parsing the db files within the workspace:



4. The PV Table should provide a tabular view of the PV name `CWS-TCPH-Fxxx:ACQ-STATE` and its current value with time stamp and alarm state:



5-7. PV Table should look like that:

*<Not saved to file>

PV	Timestamp	Value	Alarm	Saved Value
CWS-TCPH-Fxxx:ACQ-STATE	2013-05-22 07:16:42.	started		
CWS-TCPH-Fxxx:GENERATOR-STATE	2013-05-22 06:51:03.	NotReady		
CWS-TCPH-Fxxx:WAVEFORM-TYPE	2013-05-22 06:51:03.	None		
CWS-TCPH-Fxxx:TEMP	2013-05-22 07:31:51.	0.0	INVALID/UDF_ALARM	

8. After taking a snapshot of the initial values, the column Saved value should contain a copy of the Value column:

*<Not saved to file>

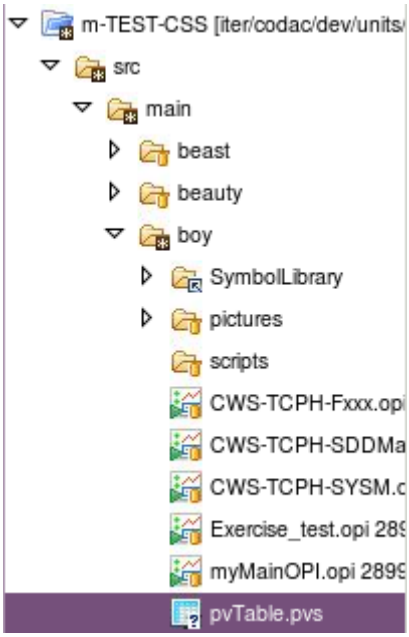
PV	Timestamp	Value	Alarm	Saved Value
CWS-TCPH-Fxxx:ACQ-STATE	2013-05-22 07:16:42.	started		started
CWS-TCPH-Fxxx:GENERATOR-STATE	2013-05-22 06:51:03.	NotReady		NotReady
CWS-TCPH-Fxxx:WAVEFORM-TYPE	2013-05-22 06:51:03.	None		None
CWS-TCPH-Fxxx:TEMP	2013-05-22 07:31:51.	0.0	INVALID/UDF_ALARM	0.0

10. The temperature is changing at 0.1 second – blue line:

*<Not saved to file>

PV	Timestamp	Value	Alarm	Saved Value
CWS-TCPH-Fxxx:ACQ-STATE	2013-05-22 07:44:01.	stopped	MINOR/STATE_ALARM	started
CWS-TCPH-Fxxx:GENERATOR-STATE	2013-05-22 06:51:03.	NotReady		NotReady
CWS-TCPH-Fxxx:WAVEFORM-TYPE	2013-05-22 06:51:03.	None		None
CWS-TCPH-Fxxx:TEMP	2013-05-22 07:46:13.	-3.4803249858398546		0.0
CWS-TCPH-Fxxx:START-ACQ	2013-05-22 07:44:01.	start		

11. The PV Table configuration file pvTable.pvs is saved within the current project:



3.4.3 DSP02 - Display Tool: PACE Editor

Prerequisite	1. IOC running 2. CSS started
Test Cases	1. Positive execution of the Operator Tool PACE Editor
Procedure	Create a PACE configuration file:

1. From the Navigator View, select m-TEST-CSS -> src -> main -> boy, then <CTRL>+N, select the wizard General -> File, click on Next and type the File name: “pvEditor.pace”. Click on Finish

2. Close the PACE Editor using the cross 

and from the Navigator View, make a right-click on the empty PACE configuration file 

, select Open With -> XML Editor.

Then copy the following content into the XML Editor:

```
<paceconfig>
<title>Demo of PACE</title>
<columns>
   <column>
     <name>Name</name>
     <access>ro</access>
     <pv>${signal}.NAME</pv>
   </column>
   <column>
     <name>Desc</name>
     <access>rw</access>
     <pv>${signal}.DESC</pv>
   </column>
   <column>
     <name>EGU</name>
     <access>rw</access>
     <pv>${signal}.EGU</pv>
   </column>
   <column>
     <name>VAL</name>
     <access>ro</access>
     <pv>${signal}.VAL</pv>
   </column>
   <column>
     <name>HIHI</name>
     <access>rw</access>
     <pv>${signal}.HIHI</pv>
   </column>
   <column>
     <name>HHSV</name>
     <access>rw</access>
     <pv>${signal}.HHSV</pv>
   </column>
   <column>
     <name>HIGH</name>
     <access>rw</access>
     <pv>${signal}.HIGH</pv>
   </column>
   <column>
     <name>HSV</name>
     <access>rw</access>
     <pv>${signal}.HSV</pv>
   </column>
   <column>
     <name>LOLO</name>
     <access>rw</access>
     <pv>${signal}.LOLO</pv>
   </column>
   <column>
     <name>LLSV</name>
     <access>rw</access>
     <pv>${signal}.LLSV</pv>
   </column>
   <column>
     <name>LOW</name>
     <access>rw</access>
     <pv>${signal}.LOW</pv>
   </column>
   <column>
     <name>LSV</name>
     <access>rw</access>
     <pv>${signal}.LSV</pv>
```

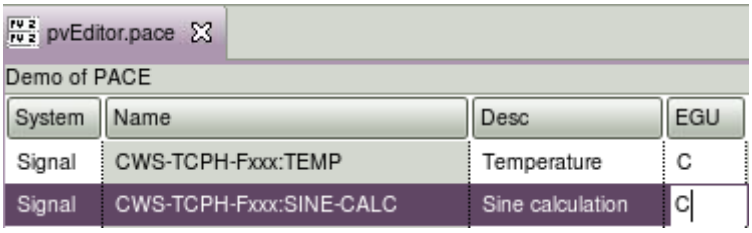
```
   </column>
</columns>

 <instances>
   <instance>
      <name>Signal</name>
      <macros>signal=CWS-TCPH-Fxxx:TEMP</macros>
   </instance>
   <instance>
      <name>Signal</name>
      <macros>signal=CWS-TCPH-Fxxx:SINE-CALC</macros>
   </instance>
   <instance>
      <name>Signal</name>
      <macros>signal=CWS-TCPH-Fxxx:SIG-GENERATED</macros>
   </instance>
</instances>
</paceconfig>
```

3. Close the XML Editor 

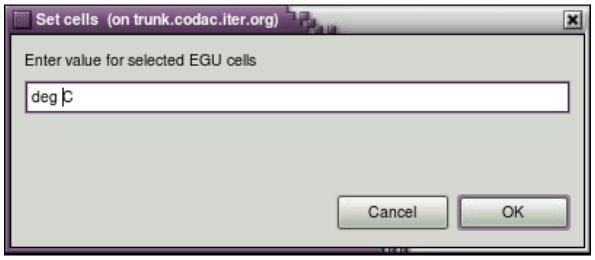
and from the Navigator View, right-click on pvEditor.pace -> Open With -> PACE. The columns in grey are read-only

4. Change the Engineering Unit EGU field of the PV `CWS-TCPH-Fxxx:SINE-CALC` to “C”:



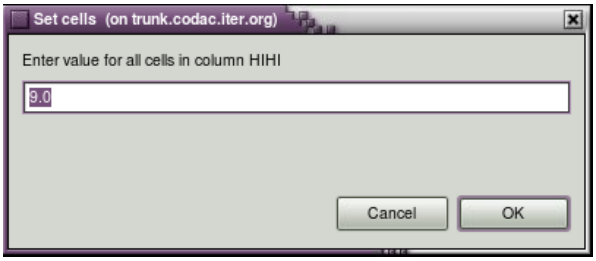
Press Enter

5. Change Multiple Limits To The Same Value: select all the rows, make a right-click in the column EGU -> Set Value. Enter the common unit “deg C”:



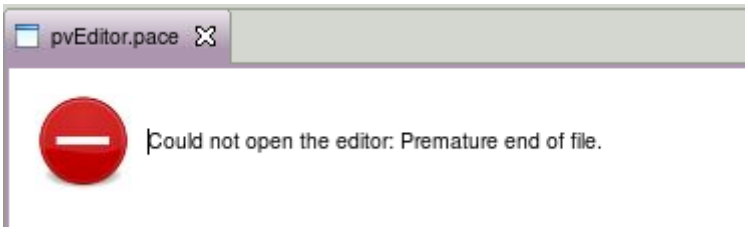
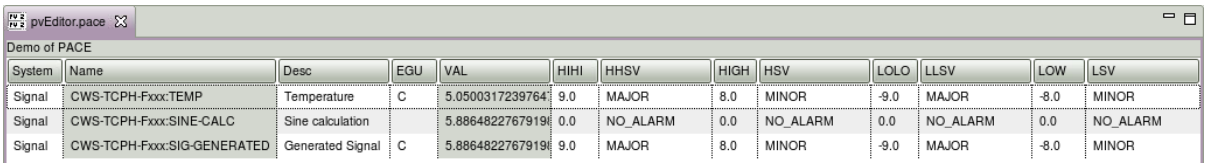
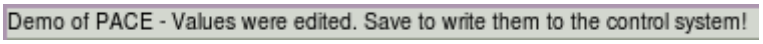
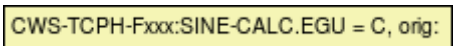
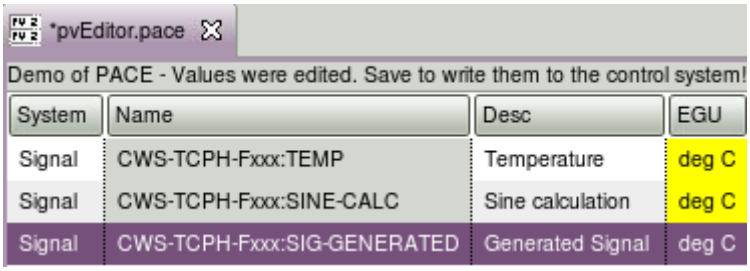
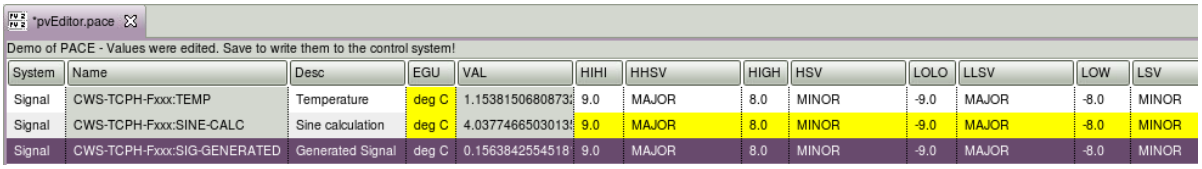
Click on OK to validate the new Engineering Unit

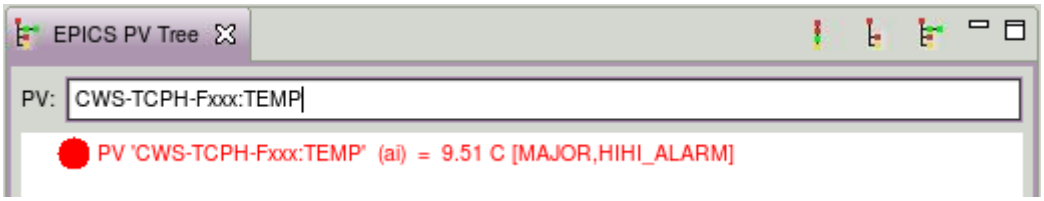
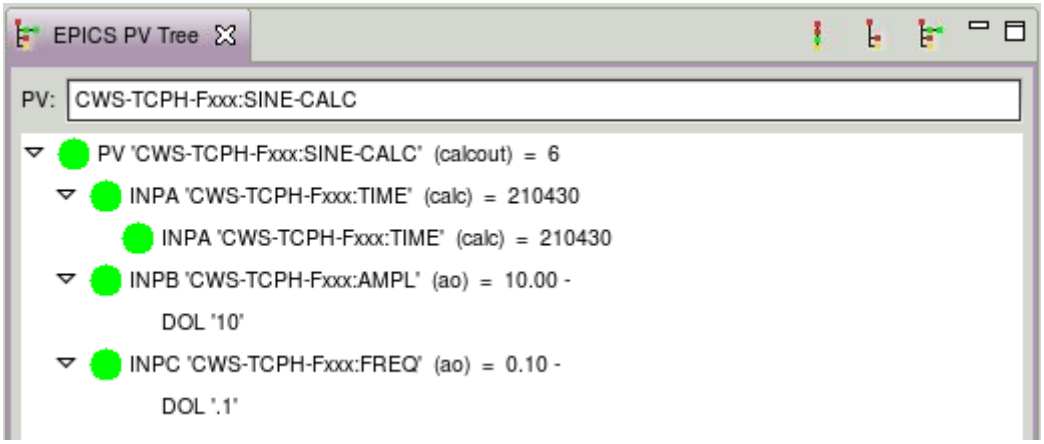
6. Change All Limits In A Column To The Same Value: click on the column header of HIHI and validate the proposed value:

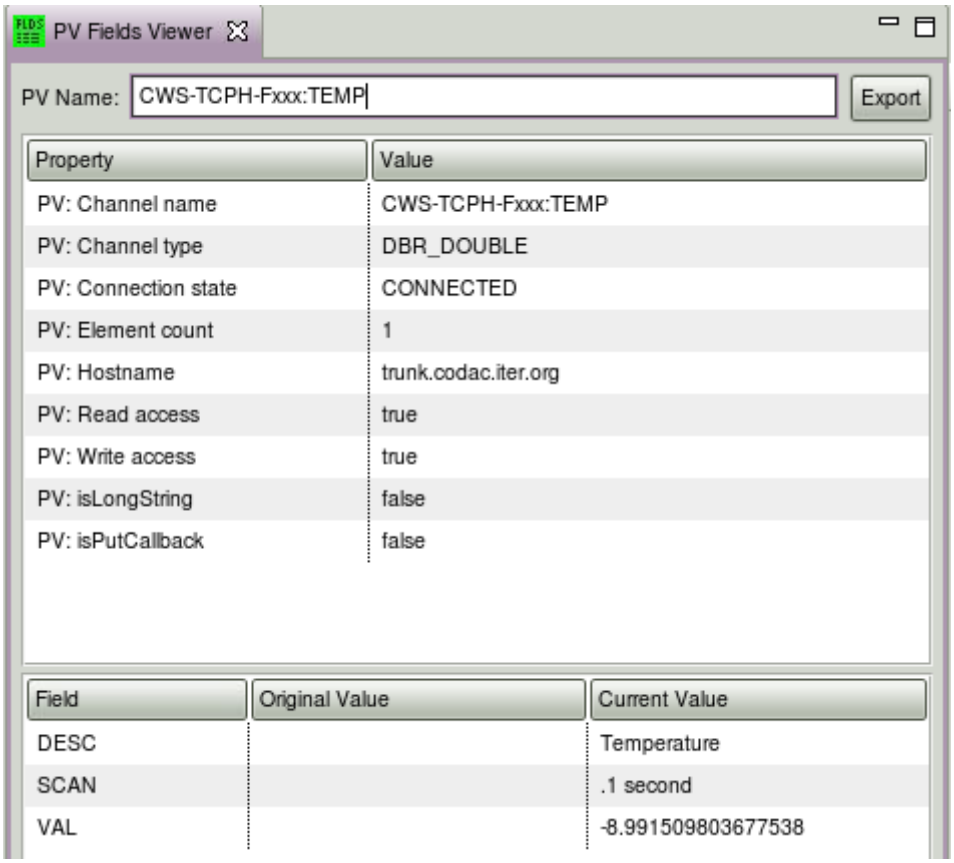


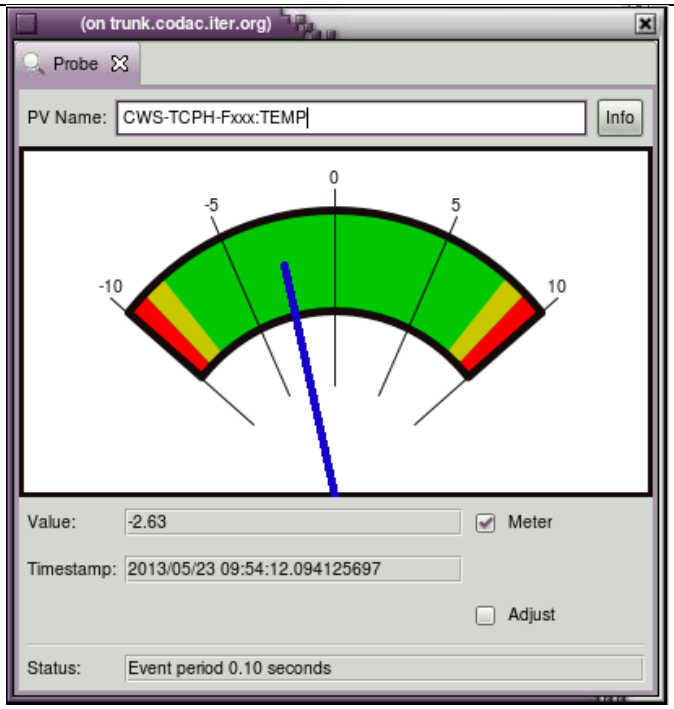
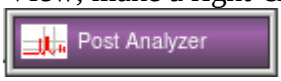
Do the same on the column headers: HHSV, HIGH, HSV, LOLO, LLSV, LOW, LSV

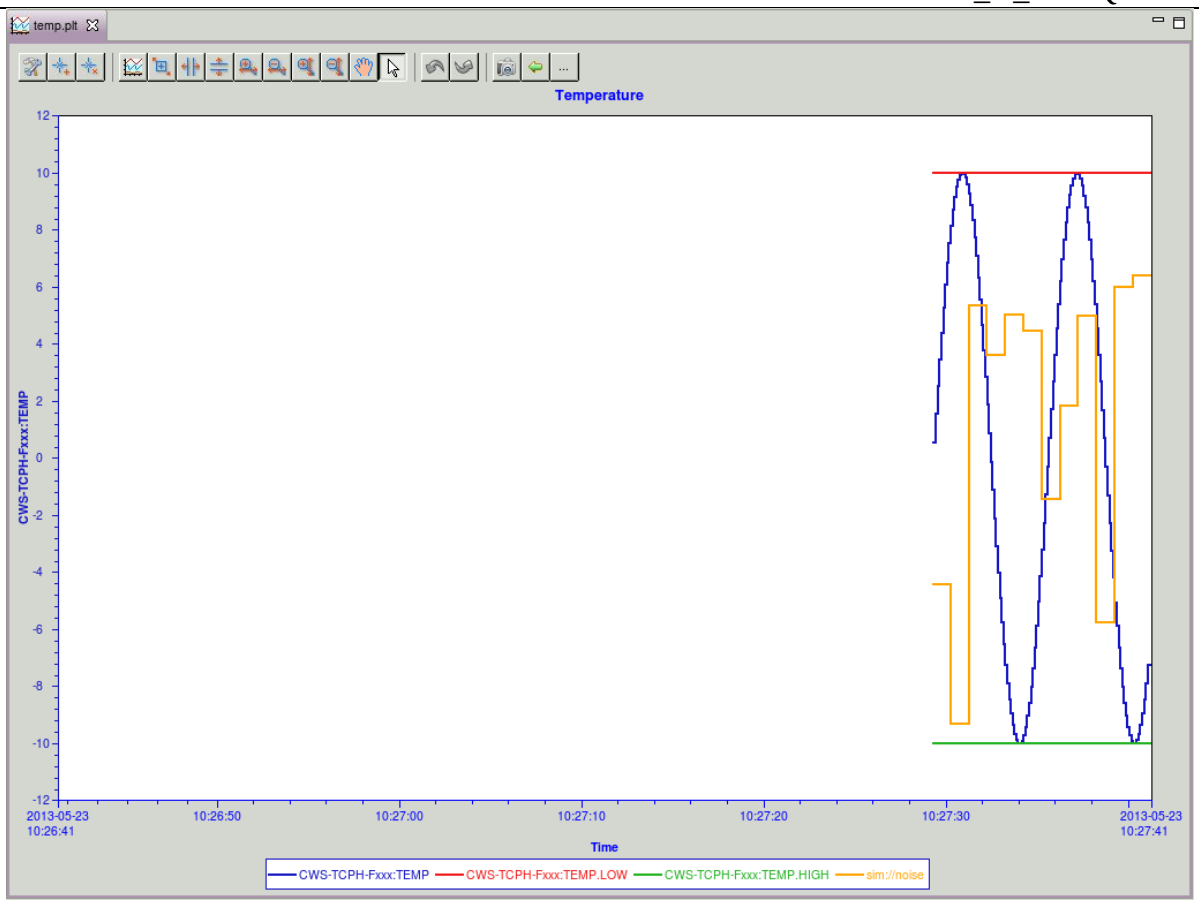
7. Save to write the new values to the EPICS database: <CTRL>+S. Before actually writing the changes to the PVs, a dialog will prompt for information to be entered into the Electronic LogBook. Click on OK to validate the new log entry

	8. Close PACE editor using the cross 	
Pass Criteria	1. PACE Editor should try to open the new empty configuration file and display an error:  3. PACE should display a tabular view of the configured PV Names:  pvEditor.pace Demo of PACE System | Name | Desc | EGU | VAL | HIHI | HHSV | HIGH | HSV | LOLO | LLSV | LOW | LSV Signal | CWS-TCPH-Fxxx:TEMP | Temperature | C | 5.0500317239764 | 9.0 | MAJOR | 8.0 | MINOR | -9.0 | MAJOR | -8.0 | MINOR Signal | CWS-TCPH-Fxxx:SINE-CALC | Sine calculation | | 5.8864822767919 | 0.0 | NO_ALARM | 0.0 | NO_ALARM | 0.0 | NO_ALARM | 0.0 | NO_ALARM Signal | CWS-TCPH-Fxxx:SIG-GENERATED | Generated Signal | C | 5.8864822767919 | 9.0 | MAJOR | 8.0 | MINOR | -9.0 | MAJOR | -8.0 | MINOR 4. After the edition of the Engineering Unit EGU field, an information is displayed on top of the table:  Moving the mouse on the field shows the following tooltip indicating the change:  If you select another row, the changed field is displayed in yellow:  5. All Engineering Units should have been changed:  *pvEditor.pace Demo of PACE - Values were edited. Save to write them to the control system! System | Name | Desc | EGU Signal | CWS-TCPH-Fxxx:TEMP | Temperature | deg C Signal | CWS-TCPH-Fxxx:SINE-CALC | Sine calculation | deg C Signal | CWS-TCPH-Fxxx:SIG-GENERATED | Generated Signal | deg C 6. The new values should be:  *pvEditor.pace Demo of PACE - Values were edited. Save to write them to the control system! System | Name | Desc | EGU | VAL | HIHI | HHSV | HIGH | HSV | LOLO | LLSV | LOW | LSV Signal | CWS-TCPH-Fxxx:TEMP | Temperature | deg C | 1.1538150680873 | 9.0 | MAJOR | 8.0 | MINOR | -9.0 | MAJOR | -8.0 | MINOR Signal | CWS-TCPH-Fxxx:SINE-CALC | Sine calculation | deg C | 4.0377466503013 | 9.0 | MAJOR | 8.0 | MINOR | -9.0 | MAJOR | -8.0 | MINOR Signal | CWS-TCPH-Fxxx:SIG-GENERATED | Generated Signal | deg C | 0.1563842554518 | 9.0 | MAJOR | 8.0 | MINOR | -9.0 | MAJOR | -8.0 | MINOR	
	### 3.4.4 DGC01 – Diagnostic Tool: EPICS PV Tree	
Prerequisite	1. IOC running 2. CSS started	
Test Cases	1. Positive execution of EPICS PV Tree	
Procedure	In CSS, start the Diagnostic Tool EPICS PV Tree:	

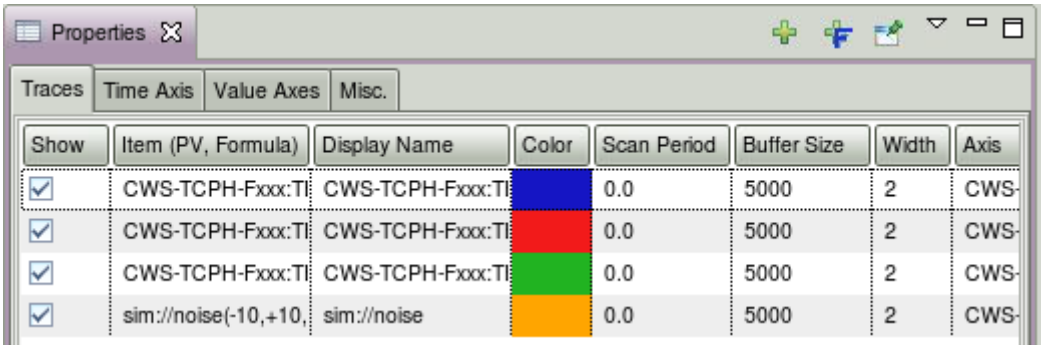
	1. Menu CSS -> Diagnostic Tools -> EPICS PV Tree 2. In the PV field, start to enter *TEM. Click on the matched PV Name `CWS-TCPH-Fxxx:TEMP` and press Enter to validate the name. Press again Enter to retrieve the PV Tree 3. In the PV field, start to enter *CA. Click on the matched PV Name `CWS-TCPH-Fxxx:SINE-CALC` and press Enter to validate the name. Press again Enter to retrieve the PV Tree 4. Use the tool bar buttons to collapse or expend the sub-sections of the tree:  5. Close EPICS PV Tree using the cross 	
Pass Criteria	2. EPICS PV Tree should display information regarding the Temperature Analogue Input (ai):  3. The second hierarchy should show a more interesting hierarchy: a calcout record reads its input value INPA from another calc record:  	
	3.4.5 DGC02 – Diagnostic Tool: PV Fields Viewer	
Prerequisite	1. IOC running 2. CSS started	
Test Cases	1. Positive execution of PV Fields Viewer	
Procedure	In CSS, start the Diagnostic Tool PV Fields Viewer: 1. Menu CSS -> Diagnostic Tools -> PV Fields Viewer 2. In the PV Name field, start to enter *TEM, select the matched PV Name `CWS-TCPH-Fxxx:TEMP` and press Enter twice to view the fields of this PV 3. Close PV Fields Viewer 	

Pass Criteria	2. The PV Fields viewer should display information about the Process Variable:  	
	3.4.6 DGC03 – Diagnostic Tool: Probe	
Prerequisite	1. IOC running 2. CSS started	
Test Cases	1. Positive execution of Probe Diagnostic Tool	
Procedure	The Probe tool allows basic reading and writing of Process Variables. In CSS, start the Diagnostic Tool Probe: 1. Menu CSS -> Diagnostic Tools -> Probe 2. In the PV Name field, start to enter *TEM, select the matched PV Name `CWS-TCPH-Fxxx:TEMP` and press Enter twice to view the value of this PV 3. Detach the view with a right-click on the tab -> Detached. Then resize the view as it fits 4. Close Probe  	
Pass Criteria	3. Probe should display the current value of the given PV together with time stamp and status:	

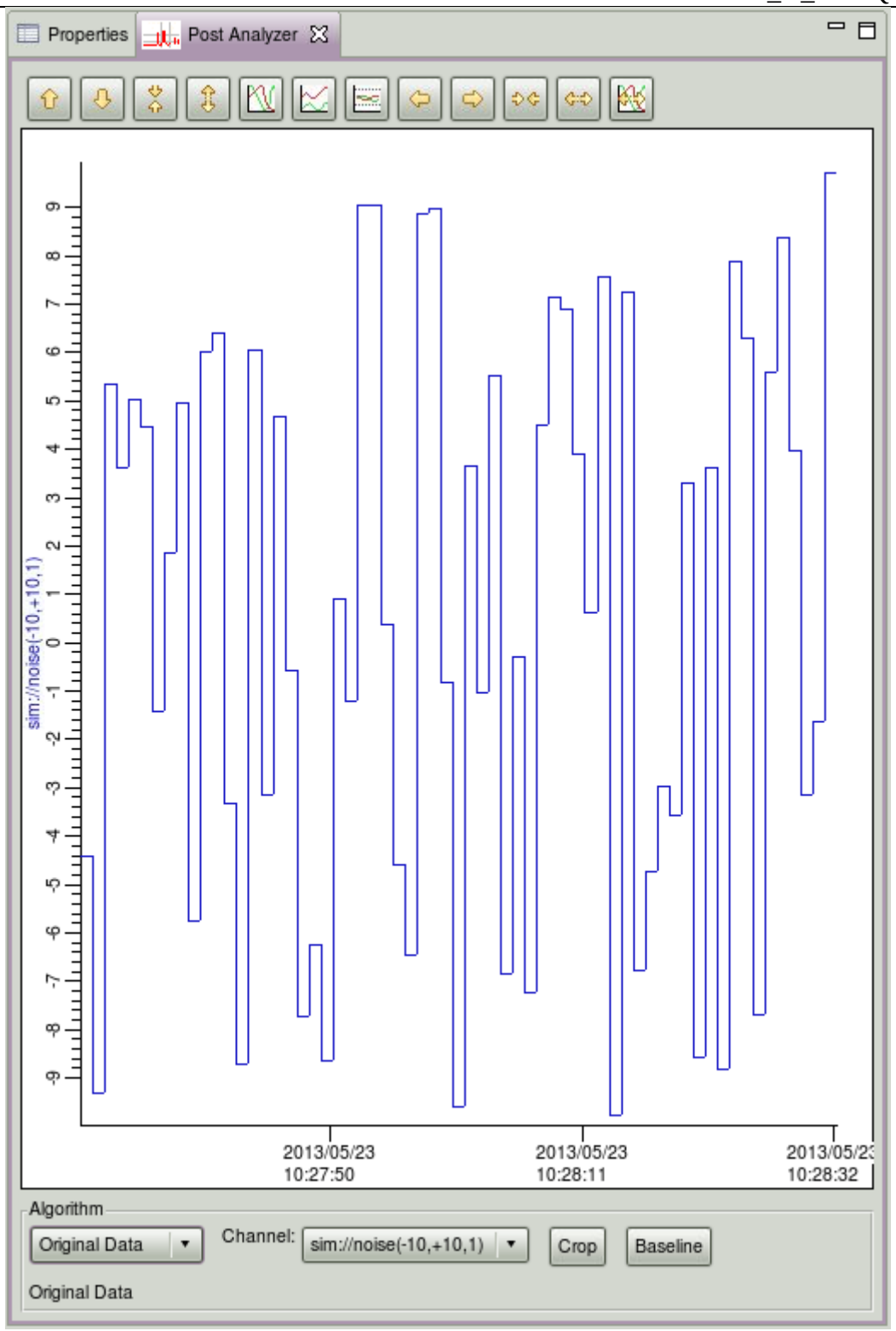
		
	3.4.7 DGC04 – Diagnostic Tool: Post Analyzer	
Prerequisite	1. IOC running 2. CSS started	
Test Cases	1. Positive execution of Post Analyzer Diagnostic Tool	
Procedure	Post Analyzer tool allows the analysis of time series data, for example from the Data Browser, in various ways. In CSS, open first a plot: 1. From the Navigator View, select m-TEST-CSS -> src -> main -> databrowser and then double-click on temp.plt 2. In the plot area, make a right-click and select  option 3. In the Properties View, make a right-click on the simulated PV sim://`noise()` and select Export Samples ->  4. Change the Algorithm from Original Data to FFT 4. Close the Post Analyser  , the plot  and the Properties View.	
Pass Criteria	1. The plot should display a noise signal between -10 to +10:	



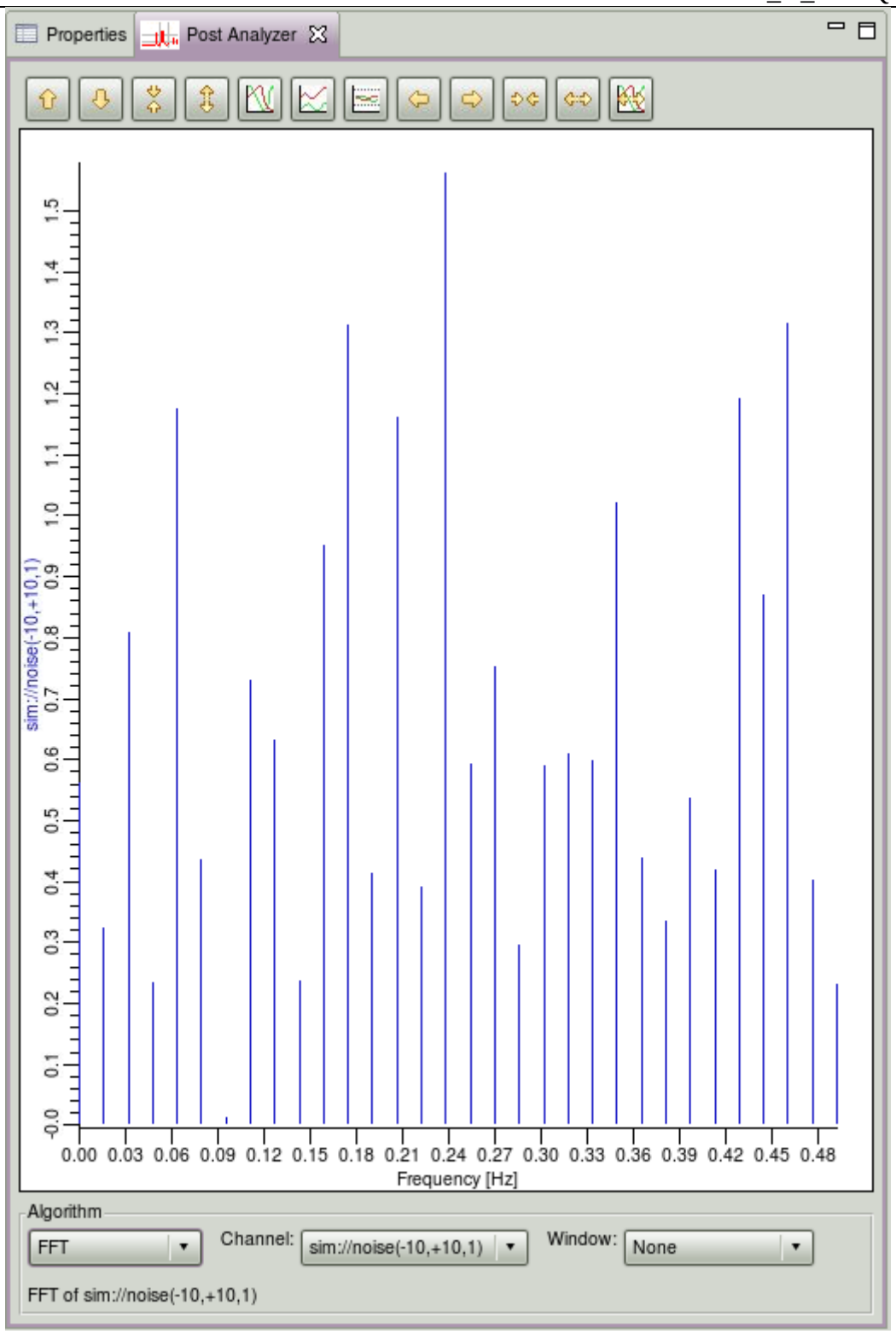
2. The Properties View should be opened:



3. The Post Analyzer tool should display the Original Data:



4. The results of the execution of FFT should be displayed:

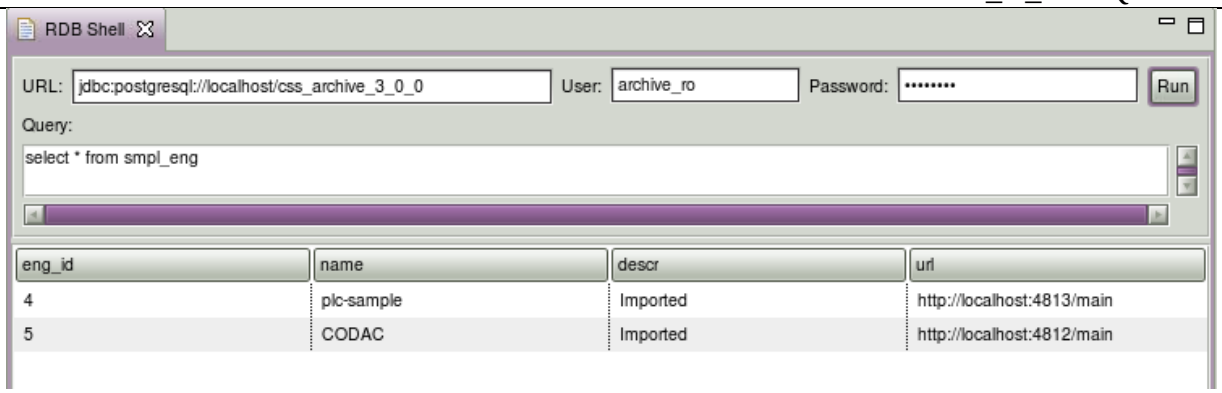


3.4.8 DBG01 – Debugging Tool: RDB Shell

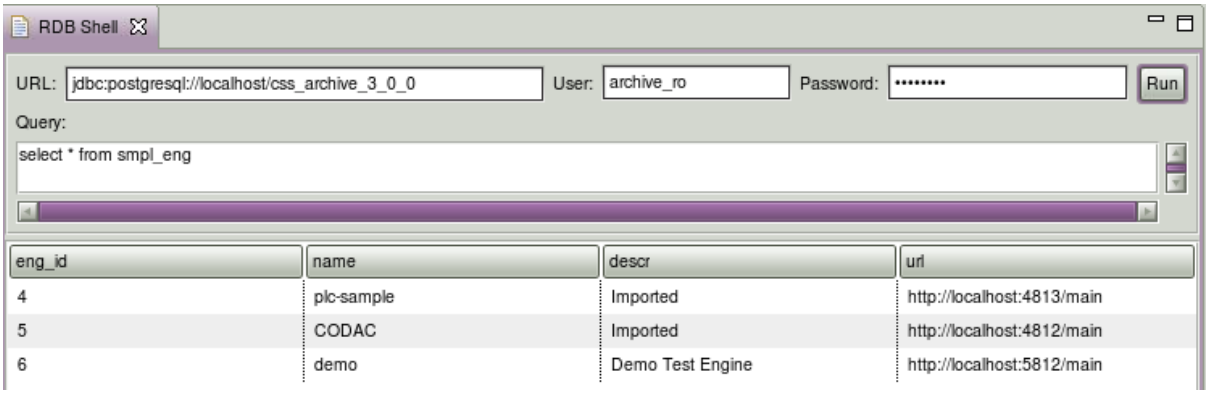
Prerequisite	1. IOC running 2. CSS started
Test Cases	1. Positive execution of RDB Shell Debugging Tool

Procedure	The RDB Shell is a utility for sending SQL statements to the RDB and viewing the result. It is a debug tool for developers. In CSS, start RDB Shell Debugging Tool:
	1. CSS -> Debugging -> RDB Shell
	2. Enter the password for the archive RDB read-only account and click on run to execute the pre-defined query “select * from smpl_eng”
	Import the archive demo configuration from the previous Linux console and start the engine (Note: make sure that you have done `$ cd m-TEST-CSS/`):
	3. $ archive-configtool -engine demo -description 'Demo Test Engine' -port 5812 -import -config src/main/beauty/CWS-TCPH-beauty.xml -replace_engine
	`Importing : src/main/beauty/CWS-TCPH-beauty.xml`
	`Engine : demo`
	`Description : Demo Test Engine`
	`URL : http://localhost:5812/main`
	`Replace engine: true`
	`Steal channels: false`
	$ archive-engine -port 5812 -engine demo&
	4. Go back in CSS and run again the query
	5. Change the query to “select * from channel where name like '%Fxxx%';” and run it. Keep in mind the channel_id of `CWS-TCPH-Fxxx:TEMP`
	6. Change the query and replace “??” by the correct channel_id: “select channel_id,smpl_time,float_val from sample where channel_id=??;” and run it
	7. First retrieve the channel_id of the system monitoring channels with the following query: “select * from channel where name like '%CWS-TCPH-Fxxx%';”. Then, keep in mind the range of the channel_ids. Try to run the following request: ”delete from sample where channel_id>?? and channel_id<??”, replacing “??” by the previous range
	8. Close the RDB Shell  View.	

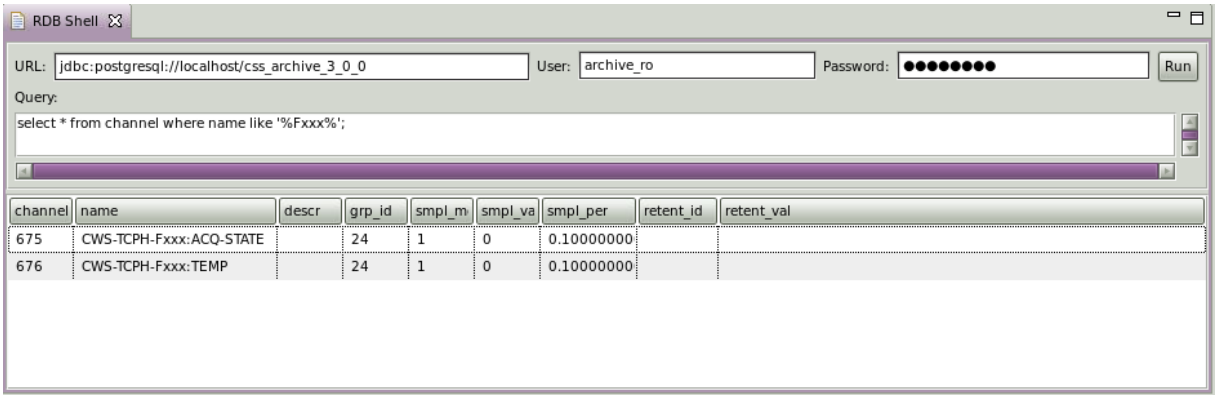
Pass Criteria	1. RDB Shell View is opened with a default RDB URL and query:  2. The list of declared archive engines should be displayed:	



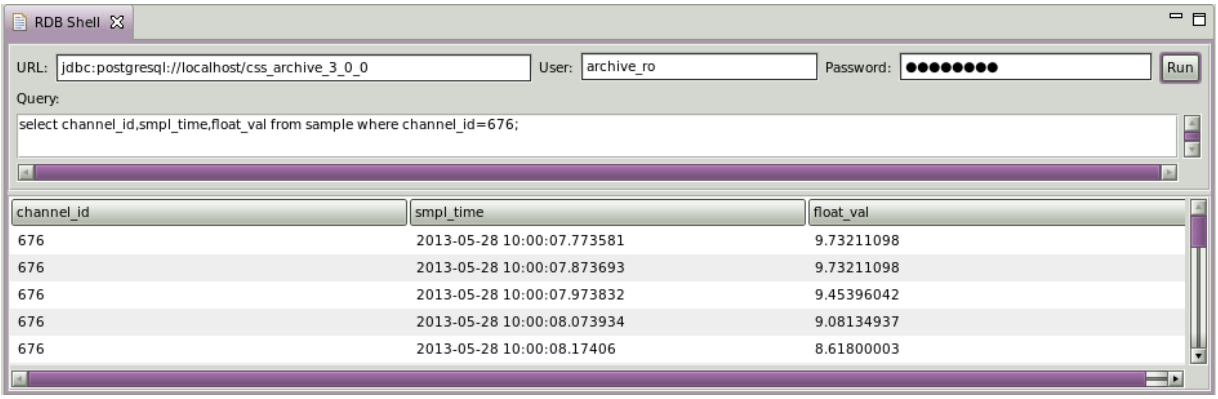
4. The new declared demo archive engine should be listed:



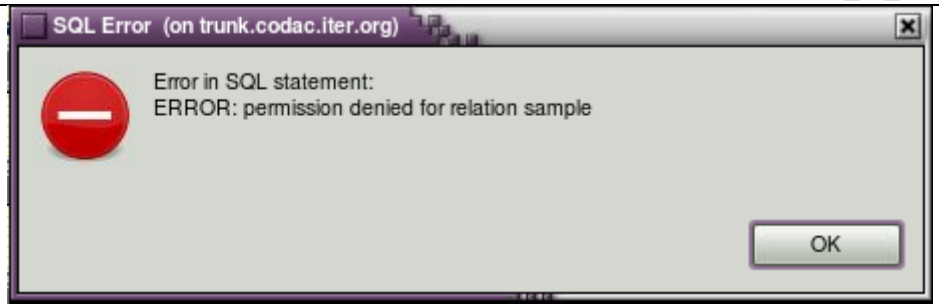
5. The list of channels to be archived for CWS system should be displayed:



6. The list of samples archived for `CWS-TCPH-Fxxx:TEMP` should be displayed:



7. It should be impossible to delete RDB rows with the read-only account:



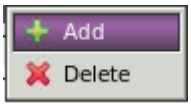
3.4.9 DSP03 - Display Tool: RDB Table Editor

Prerequisite	1. IOC running 2. CSS started
Test Cases	1. Positive execution of the Operator Tool RDB Table Editor
Procedure	Use the already configured RDB file: 1. From the Navigator View, select m-TEST-CSS -> src -> main -> boy then double click on archive.rdb file. Enter the archive read-write account and password:

2. Modifying Cells: in the table editor, modify selected cells by clicking the cell and entering a new value – for instance select a PV to be archived and change the Description field to “PV To Be Archived”. Press Enter.

Note that the first, leftmost column contains a “key” that’s used to identify the row in the RDB. In the screenshot it’s the “ID” column. You can only change that “key” for newly added rows (see below), not for existing rows

3. Adding Rows: add a new row by opening the context menu (right-click in table) and selecting "Add":



The new row will appear at the bottom of the table, where you can now enter the individual cell values, including the "key" which must usually be unique, i.e. different from other rows:

<ID>	<CHANNEL>	<DESCR>	<GRP_ID	<SMPL_MODE_ID>	<SMPL_VAL>	<SMPL_PER>

Enter the following line:

10000	MyPVToBeArchived	Another PV	100	1	0	5

4. Deleting Rows: select the newly entered row, then mark it for deletion via the context menu (right-click in table) and selecting "Delete". Rows marked for deletion will have a grey

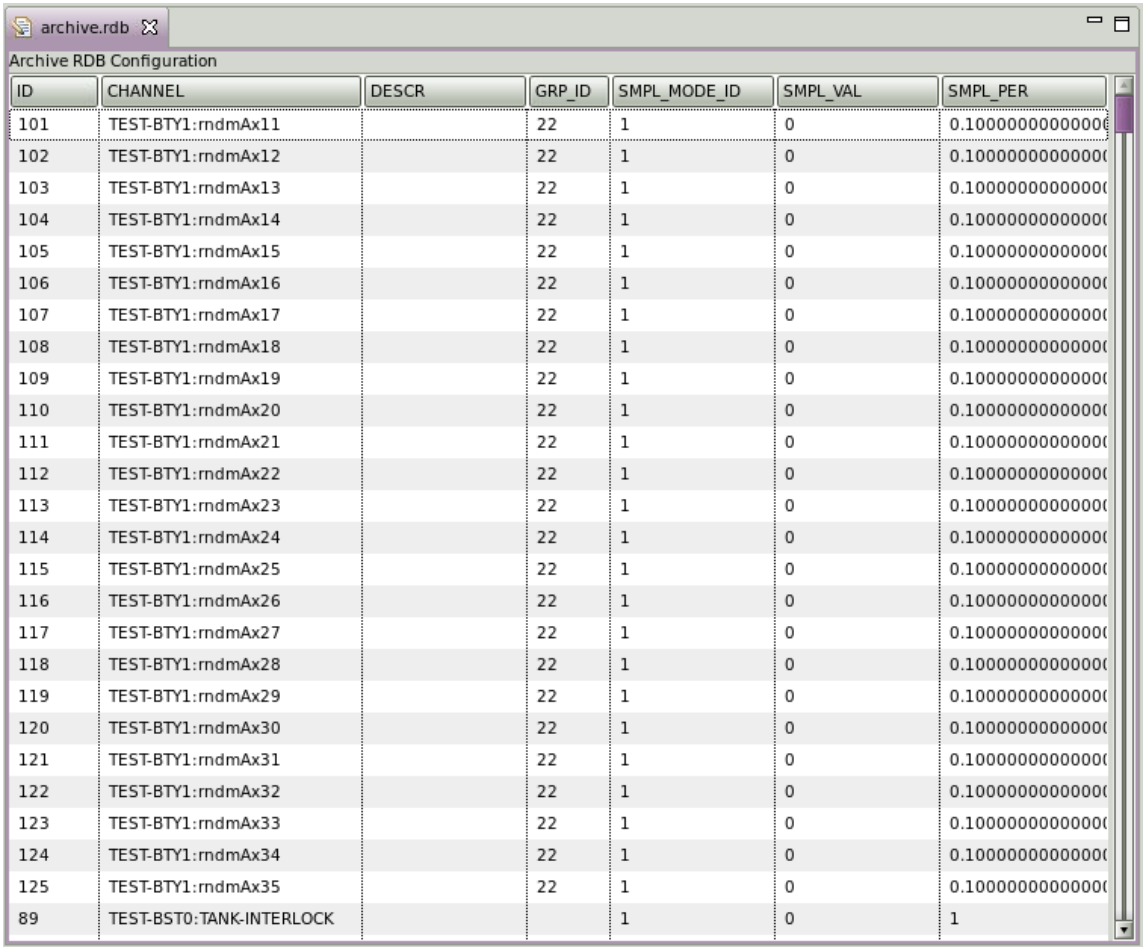
background, and a tool-tip will also indicate that the row was marked for deletion

5. Saving Changes to the RDB: to write new rows, cell modifications or row deletions to the RDB, "save" the changes via the File/Save menu

8. Close RDB Table Editor using the cross 

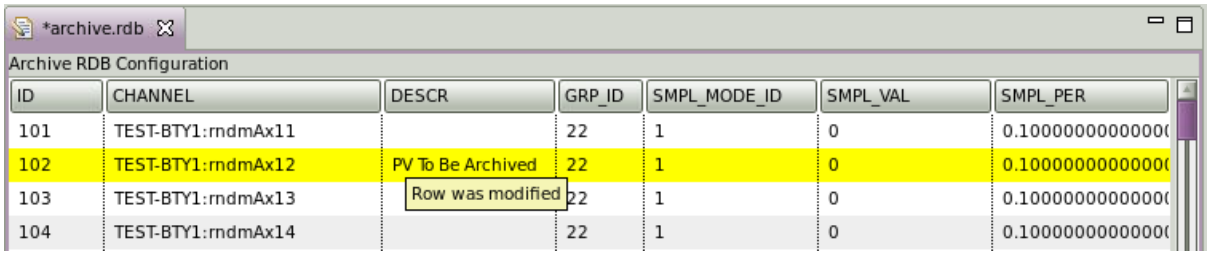
Pass Criteria

1. The archive channel table should be displayed:



ID	CHANNEL	DESCR	GRP_ID	SMPL_MODE_ID	SMPL_VAL	SMPL_PER
101	TEST-BTY1:rndmAx11		22	1	0	0.10000000000000
102	TEST-BTY1:rndmAx12		22	1	0	0.10000000000000
103	TEST-BTY1:rndmAx13		22	1	0	0.10000000000000
104	TEST-BTY1:rndmAx14		22	1	0	0.10000000000000
105	TEST-BTY1:rndmAx15		22	1	0	0.10000000000000
106	TEST-BTY1:rndmAx16		22	1	0	0.10000000000000
107	TEST-BTY1:rndmAx17		22	1	0	0.10000000000000
108	TEST-BTY1:rndmAx18		22	1	0	0.10000000000000
109	TEST-BTY1:rndmAx19		22	1	0	0.10000000000000
110	TEST-BTY1:rndmAx20		22	1	0	0.10000000000000
111	TEST-BTY1:rndmAx21		22	1	0	0.10000000000000
112	TEST-BTY1:rndmAx22		22	1	0	0.10000000000000
113	TEST-BTY1:rndmAx23		22	1	0	0.10000000000000
114	TEST-BTY1:rndmAx24		22	1	0	0.10000000000000
115	TEST-BTY1:rndmAx25		22	1	0	0.10000000000000
116	TEST-BTY1:rndmAx26		22	1	0	0.10000000000000
117	TEST-BTY1:rndmAx27		22	1	0	0.10000000000000
118	TEST-BTY1:rndmAx28		22	1	0	0.10000000000000
119	TEST-BTY1:rndmAx29		22	1	0	0.10000000000000
120	TEST-BTY1:rndmAx30		22	1	0	0.10000000000000
121	TEST-BTY1:rndmAx31		22	1	0	0.10000000000000
122	TEST-BTY1:rndmAx32		22	1	0	0.10000000000000
123	TEST-BTY1:rndmAx33		22	1	0	0.10000000000000
124	TEST-BTY1:rndmAx34		22	1	0	0.10000000000000
125	TEST-BTY1:rndmAx35		22	1	0	0.10000000000000
89	TEST-BST0:TANK-INTERLOCK			1	0	1

2. Rows with modified cells will be highlighted in yellow, and a tool-tip will also indicate that the row was changed:



ID	CHANNEL	DESCR	GRP_ID	SMPL_MODE_ID	SMPL_VAL	SMPL_PER
101	TEST-BTY1:rndmAx11		22	1	0	0.10000000000000
102	TEST-BTY1:rndmAx12	PV To Be Archived	22	1	0	0.10000000000000
103	TEST-BTY1:rndmAx13		22	1	0	0.10000000000000
104	TEST-BTY1:rndmAx14		22	1	0	0.10000000000000

3. The new line is highlighted:

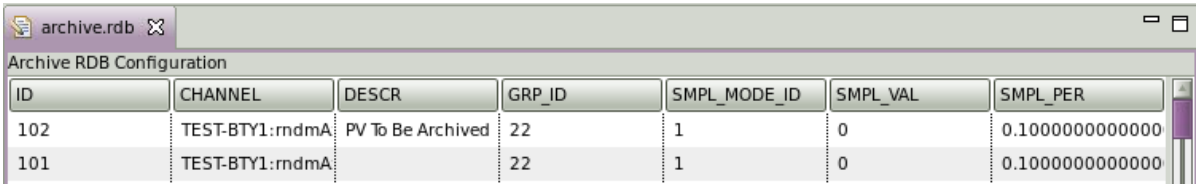
10000	MyPVToBeArchived	Another PV	100	1	0	5

4. The row to be deleted is in grey:

10000	MyPVToBeArchived	Another PV	100	1	0	5

5. Close 

and open again archive.rdb configuration file (m-TEST-CSS -> src -> main -> boy -> archive.rdb) and check that there is no more modified rows in yellow and that the Description made in step 2 has been taken into account



ID	CHANNEL	DESCR	GRP_ID	SMPL_MODE_ID	SMPL_VAL	SMPL_PER
102	TEST-BTY1:rndmA	PV To Be Archived	22	1	0	0.1000000000000
101	TEST-BTY1:rndmA		22	1	0	0.1000000000000

3.4.10 DBG02 – Debugging Tool: JMS Monitor

Prerequisite

1. IOC running
2. CSS started
3. In the previous Linux console:

$ alarm-configtool -root demo -import -file src/main/beast/CWS-TCPH-beast.xml

```
Alarm Config Tool 1.0.0.codac_core_xxx
2013-05-24 12:16:12.254 INFO [Thread 21]
org.apache.activemq.transport.failover.FailoverTransport (doReconnect) - Successfully connected
to tcp://localhost:61616
Reading RDB configuration of 'demo'
Deleting existing RDB configuration for 'demo'
Importing configuration 'demo' from src/main/beast/CWS-TCPH-beast.xml
Loading /demo/CWS-TCPH
Loading /demo/CWS-TCPH/CWS-TCPH
Loading /demo/CWS-TCPH/CWS-TCPH/CWS-TCPH-Fxxx
Loading /demo/CWS-TCPH/CWS-TCPH/CWS-TCPH-Fxxx/CWS-TCPH-Fxxx:ACQ-STATE
Loading /demo/CWS-TCPH/CWS-TCPH/CWS-TCPH-Fxxx/CWS-TCPH-Fxxx:TEMP
Loading /demo/CWS-TCPH/CWS-TCPH/CWS-TCPH-SYSM
Loading /demo/CWS-TCPH/CWS-TCPH/CWS-TCPH-SYSM/CWS-TCPH-SYSM:H0-SYSHLTS
Loading /demo/CWS-TCPH/CWS-TCPH/CWS-TCPH-SYSM/CWS-TCPH-SYSM:H0CORE-IOCHLTS
Loading /demo/CWS-TCPH/CWS-TCPH/CWS-TCPH-SYSM/CWS-TCPH-SYSM:H0SYSM-IOCHLTS
```

$ alarm-server -root demo&

```
2013-05-24 12:18:33.598 INFO [Thread 1] org.csstudio.alarm.beast.server.Application (start) -
Alarm Server 1.0.0.codac_core_xxx started for 'demo' configuration
Alarm Server 1.0.0.codac_core_xxx
Configuration Root: demo
Database URL:       jdbc:postgresql://localhost/css_alarm_3_0_0
JMS URL:            failover:(tcp://localhost:61616)?randomize=false
JMS Server Topic:   demo_SERVER
JMS Client Topic:   demo_CLIENT
JMS Talk Topic:     demo_TALK
JMS Global Topic:   GLOBAL_SERVER
```

Test Cases

1. Positive execution of JMS Monitor Debugging Tool

Procedure

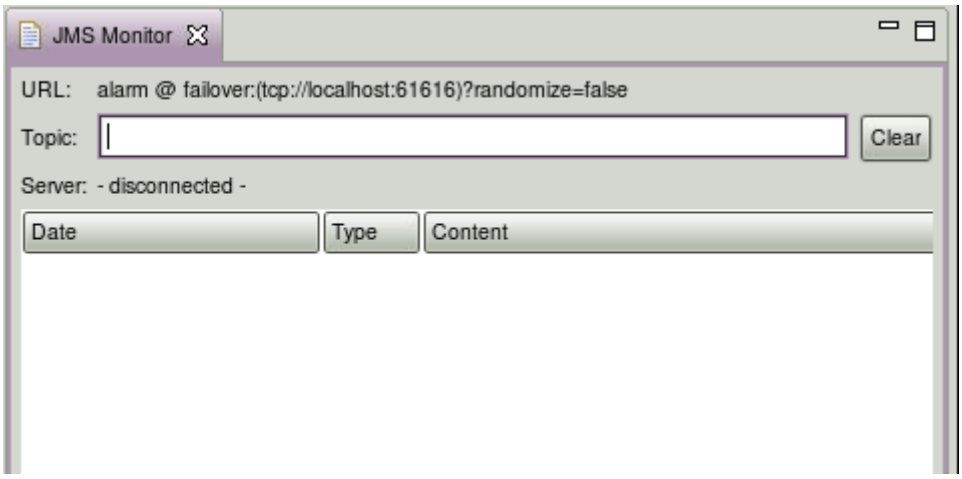
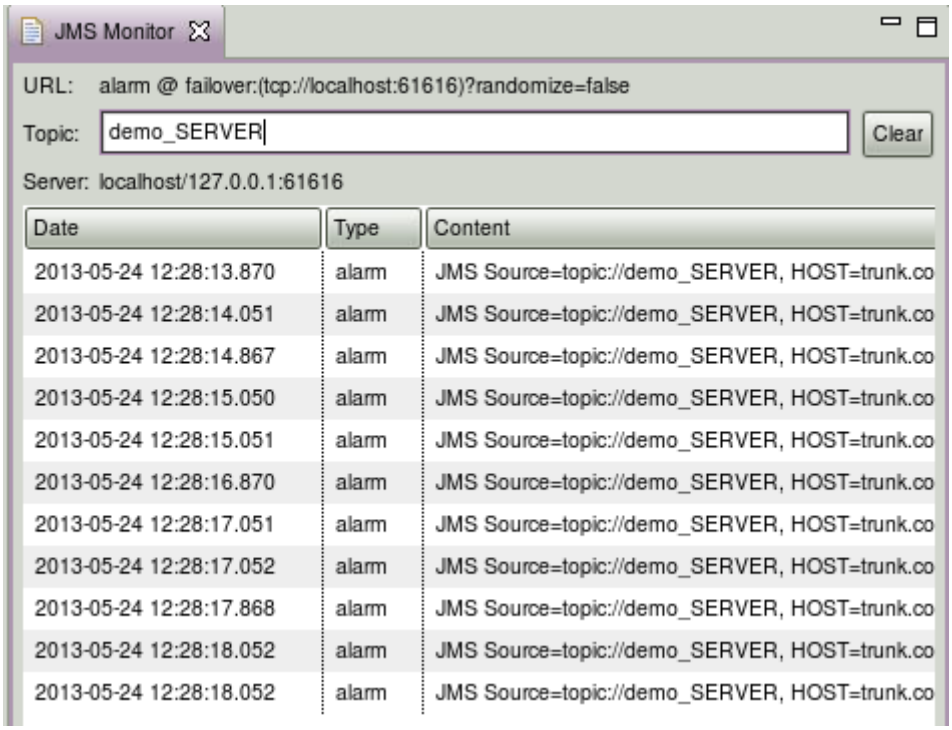
The JMS Monitor is a utility for monitoring raw JMS messages. It is a debug tool for developers. In CSS, start JMS Monitor Debugging Tool:

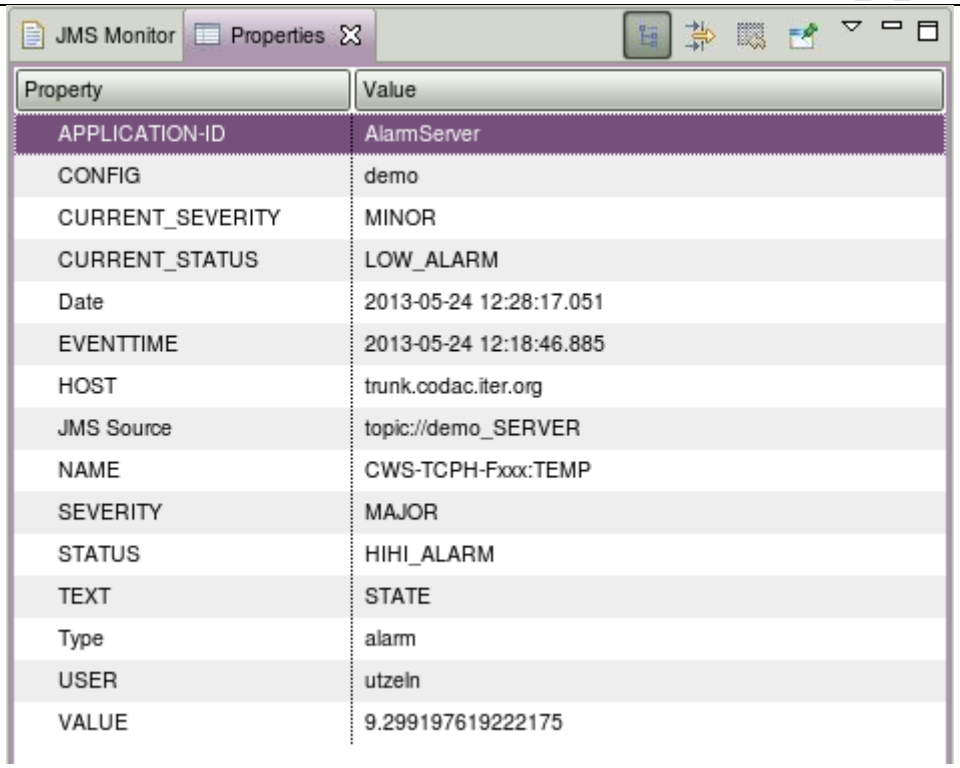
1. CSS -> Debugging -> JMS Monitor

2. Enter the topic “demo_SERVER” and press Enter

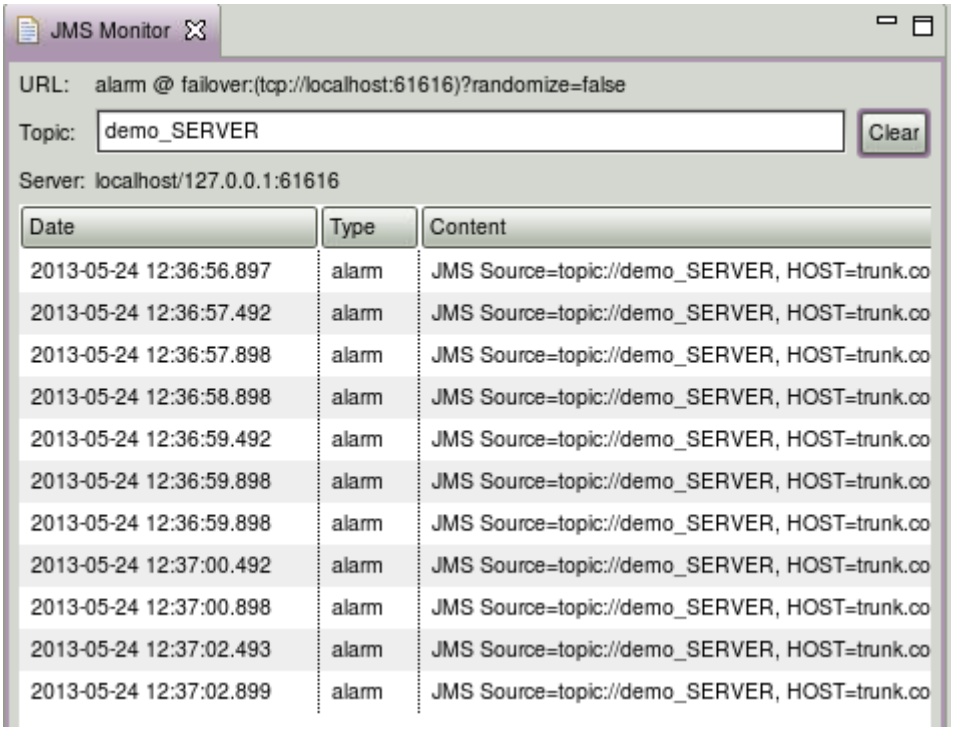
3. Double click on one JMS message to show the details of the alarm. Check the content and close the Properties View

4. By default, JMS Monitor only keeps the last 500 messages in a ring buffer. Wait enough time to see that the first messages are overwritten by the next ones

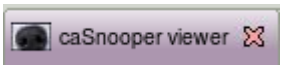
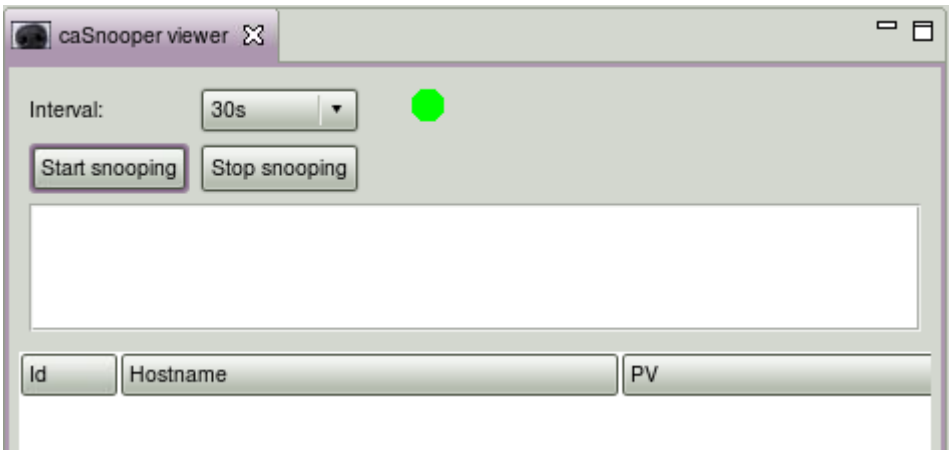
	5. Click on the button Clear to remove received messages in the list 6. Close the JMS Monitor  View.	
Pass Criteria	1. JMS Monitor should start:  2. Alarm notifications should arrive and be displayed:  3. A Properties View is opened with the details of the selected alarm:	

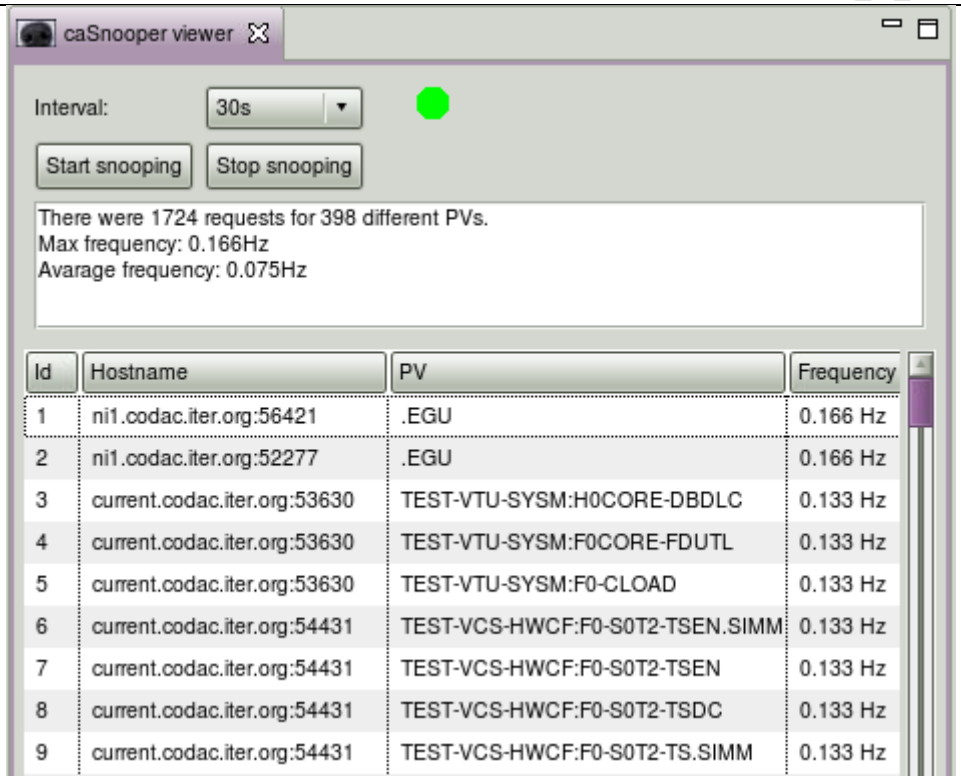


6. The JMS Message ring buffer should be flushed and start to be filled again:



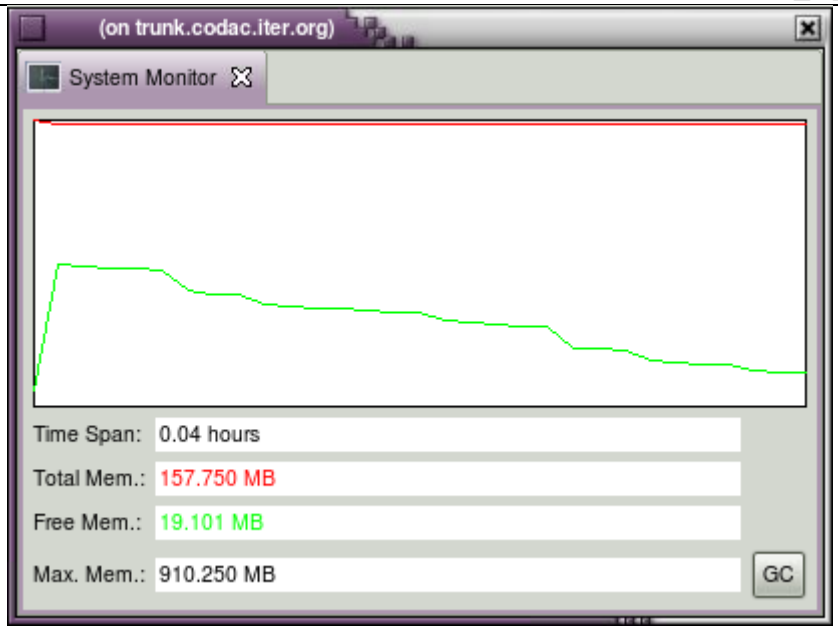
	3.4.11 UTY01 – Utilities: CA Snooper	
Prerequisite	1. IOC running 2. CSS started	
Test Cases	1. Positive execution of Channel Access CA Snooper Utility	

Procedure	CSS CA Snooper is a simple program to display statistics of broadcasts on the network. In CSS, start CA Snooper utility: 1. CSS -> Utilities -> CA Snooper 2. Choose the Interval using the combo box and select 30s. Then click on the button “Start snooping” 3. After 30s, some results and statistics should be displayed and refreshed at the given interval 4. Try to sort the results differently by clicking on each column 5. Stop snooping 6. Close the CA Snooper  View.	
Pass Criteria	1. csSnooper viewer should start:  2. The status of CA Snooper should be green:  3. Some results and statistics should be displayed:	

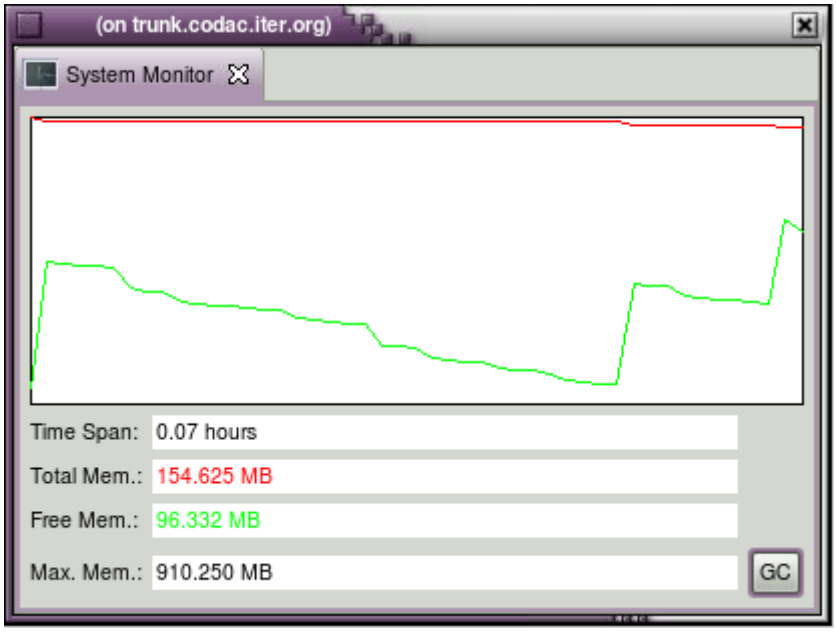


3.4.12 UTY02 – Utilities: System Monitor

Prerequisite	1. IOC running 2. CSS started
Test Cases	1. Positive execution of System Monitor Utility
Procedure	System Monitor is a simple utility for tracking memory usage. In CSS, start System Monitor utility: 1. CSS -> Utilities -> System Monitor 2. Press the "GC" button to trigger a garbage collection 3. Close the System Monitor  View.
Pass Criteria	1. System Monitor should start and after some time give the "max", "total" and "free" memory obtained from the Java Runtime API:

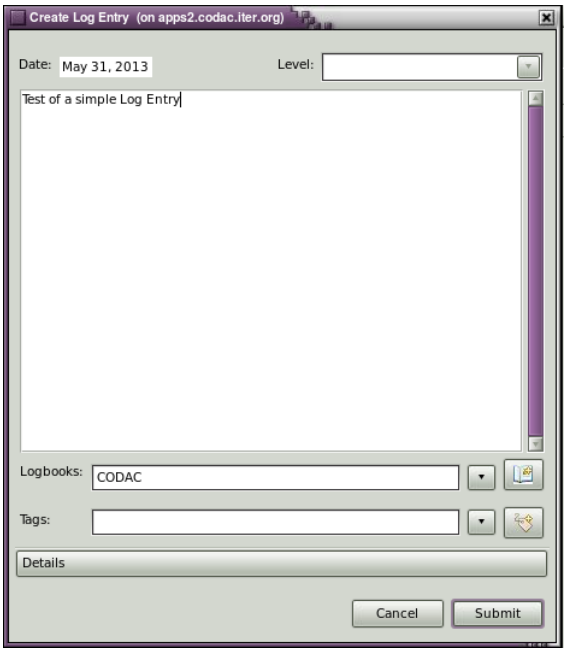
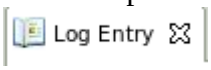


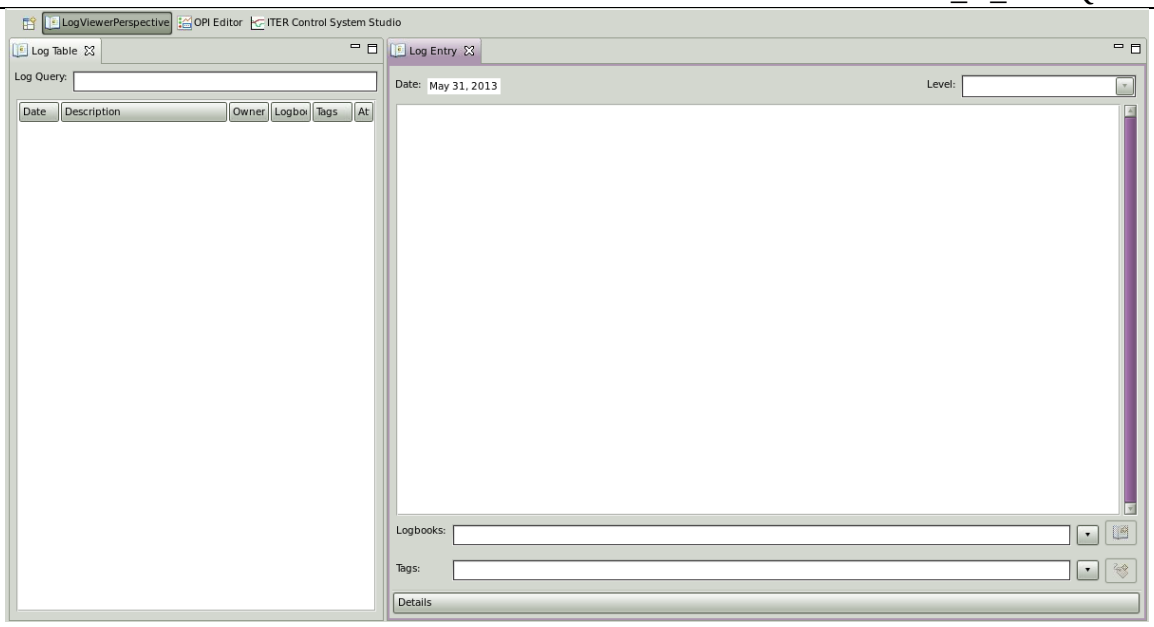
2. After the garbage collection, the memory that the JVM considers as "free" is increased (green trace):



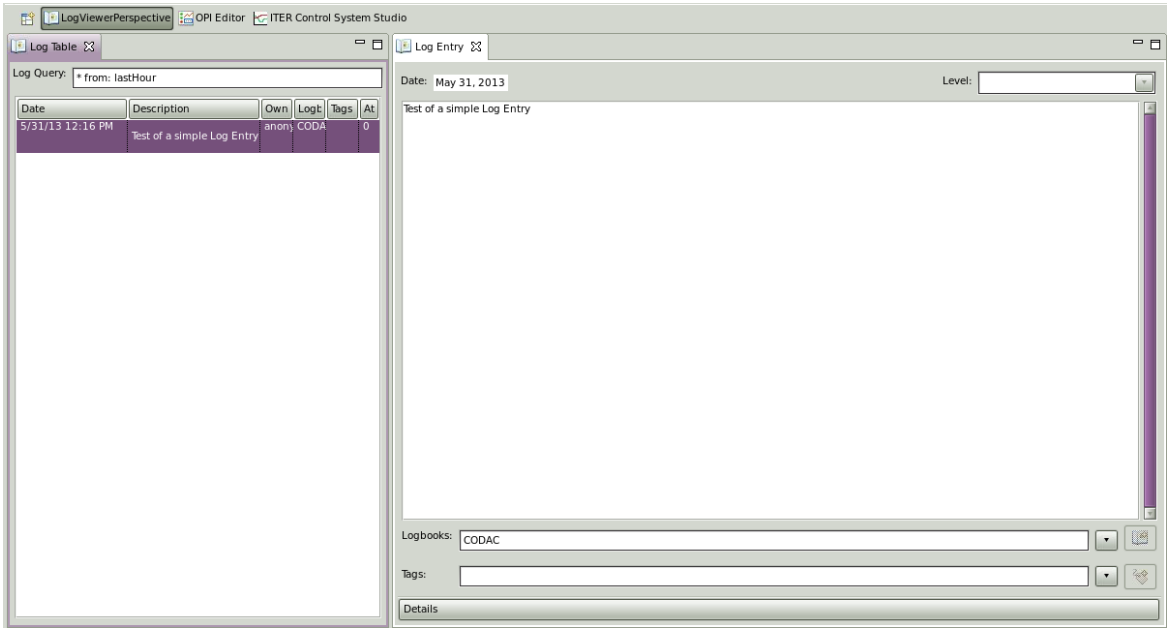
3.4.13 UTY03 – Utilities: Create Log Entry

Prerequisite	1. IOC running 2. CSS started
Test Cases	1. Positive execution of the Create Log Entry Utility
Procedure	This tool allows entering messages into the logbook. In CSS, start Create Log Entry utility: 1. CSS -> Utilities -> Create Log Entry 2. In the text field below the Date, enter “Test of a simple Log Entry.” And click on Submit

Pass Criteria	1-2. The Create Log Entry windows should open to allow entering the detail of the entry: 	
	3.4.14 UTY04 – Utilities: Search Logbook	
Prerequisite	1. IOC running 2. CSS started 3. A simple Log Entry created	
Test Cases	1. Positive execution of the Search Logbook Utility	
Procedure	This tool lists the log entries. In CSS, start the Search Logbook utility: 1. CSS -> Utilities -> Search Logbook 2. In the Log Query field, enter "* from: lastHour" to list the last recent entries and press Enter. Select "Test of a simple Log Entry" 3. Close the LogViewer Perspective  via a right-click -> Close. Close also the Log Entry 	
Pass Criteria	1. A new perspective for the Log Viewer should open:	

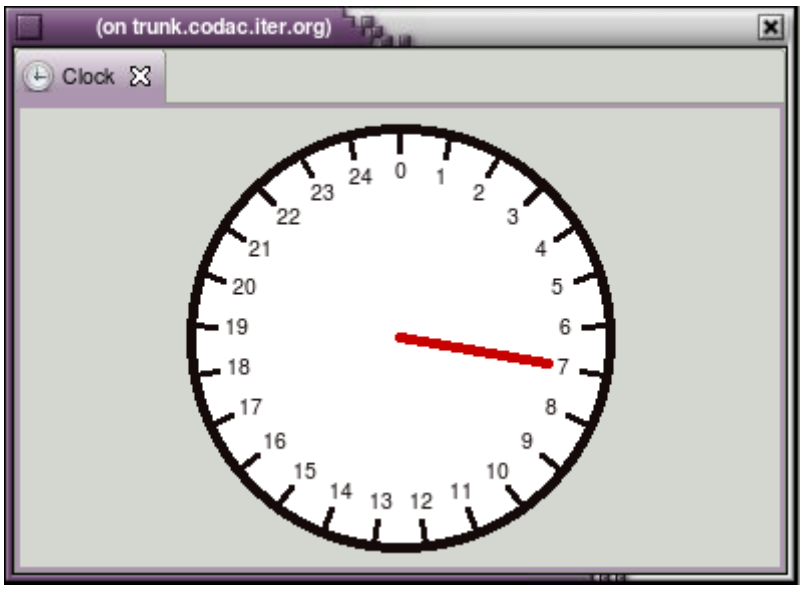
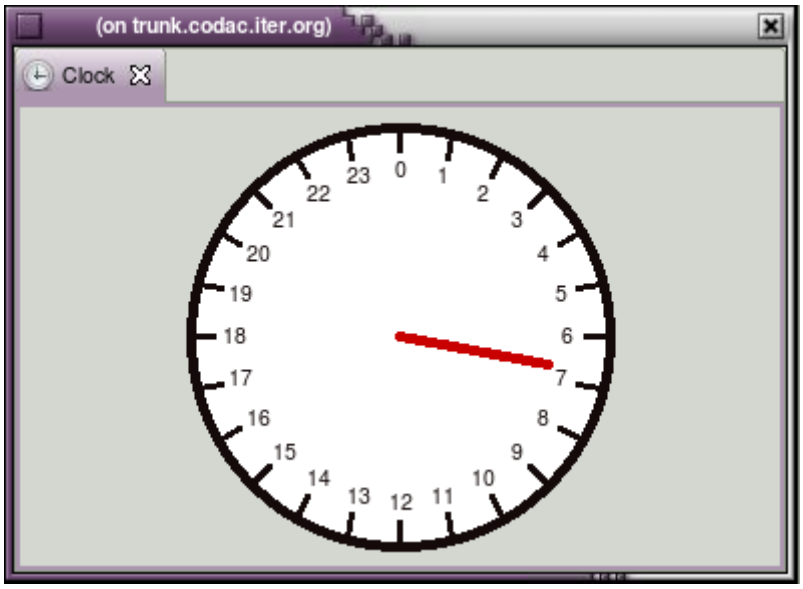


2. The simple Log Entry created should be displayed:

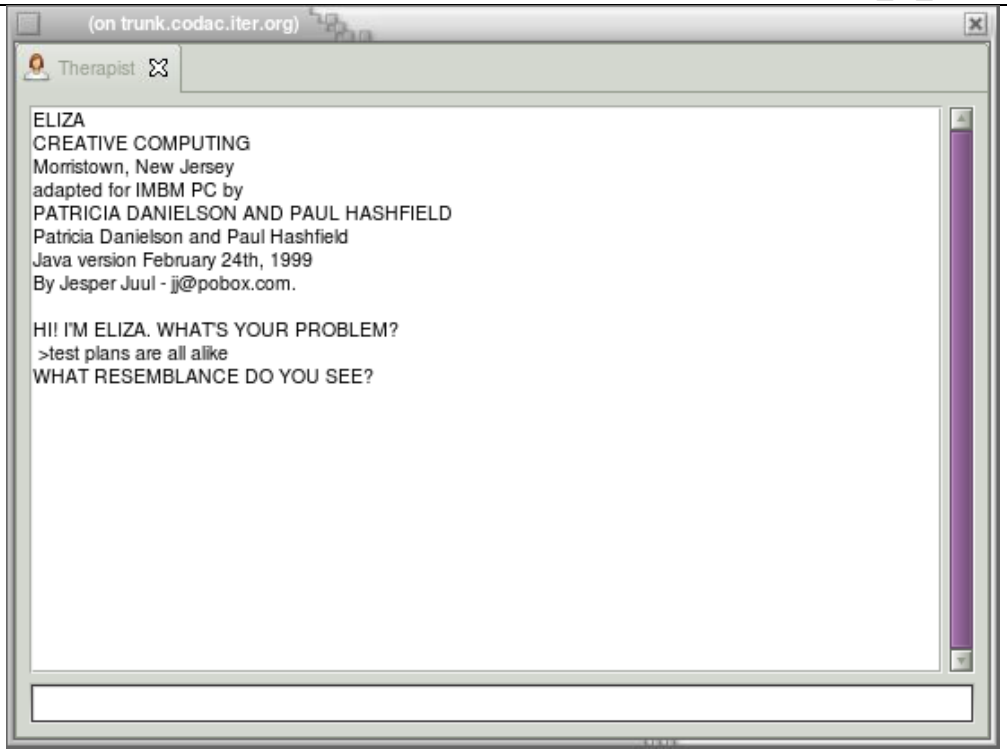


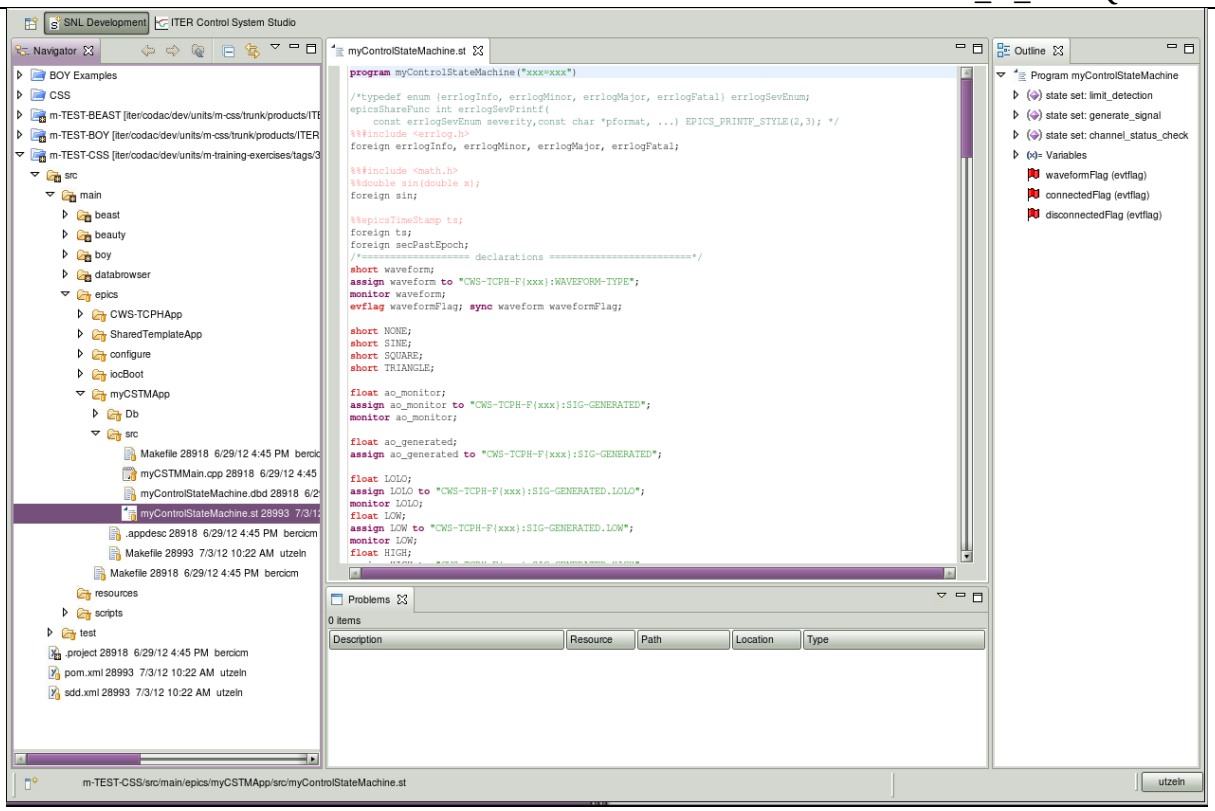
3.4.15 UTY05 – Utilities: Clock

Prerequisite	1. IOC running 2. CSS started
Test Cases	1. Positive execution of the Clock Utility
Procedure	The CSS clock provides the people who interact with the control system with a highly accurate clock, fully integrated into the Control System Studio workbench. In CSS, start the Clock utility: 1. CSS -> Utilities -> Clock 2. Change the Clock preferences via the menu Edit -> Preferences… -> CSS Applications ->

	Utilities -> Clock. Change the number of Hours to 24. Click on Apply 3. To see the changes, you need first to close the clock  and reopen it CSS -> Utilities -> Clock 4. Close the Clock Utility 	
Pass Criteria	1. The clock should open an in contrast to a layman's 24 hour clock, the CSS clock displays a 25 hour clock face, thereby providing CSS users with an extra hour each day at no additional charge!  3. A clock of 24 hours:  	
	3.4.16 UTY06 – Utilities: Therapist	
Prerequisite	1. IOC running	

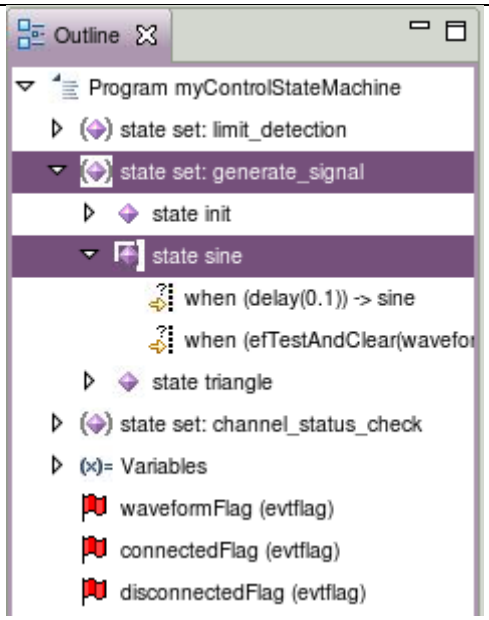
	2. CSS started	
Test Cases	1. Positive execution of the Therapist Utility	
Procedure	The control system environment can present unique stress factors to operators. At times it might be helpful to have a psychiatrist in the control room, but that is often impractical. In the mid twentieth century, Dr. Joseph Weizenbaum created the well-known "ELIZA" program. It acts like a Rogerian therapist, returning statements based on the user's input. In CSS, start the Therapist utility: 1. CSS -> Utilities -> Therapist 2. In the field <Enter your sorrows here>, enter for instance “test plans are all alike” and press Enter. Check if Eliza answer helps to solve the issue 3. Close the Therapist Utility 	
Pass Criteria	1. The Therapist Utility should open:  2. Eliza answer should look like this one:	

		
	3.4.17 SNL01 – SNL: SNL development Perspective	
Prerequisite	1. IOC running 2. CSS started	
Test Cases	1. Positive execution of the SNL Development Perspective	
Procedure	In CSS, open the dedicated SNL Development Perspective: 1. Window -> Open Perspective -> Other… 2. Select SNL Development  and click on OK 3. From the Navigator View, browse m-TEST-CSS -> src -> epics -> myCSTMApp -> src. Double-click on the state machine myControlStateMachine.st	
Pass Criteria	3. The SNL Editor should open:	

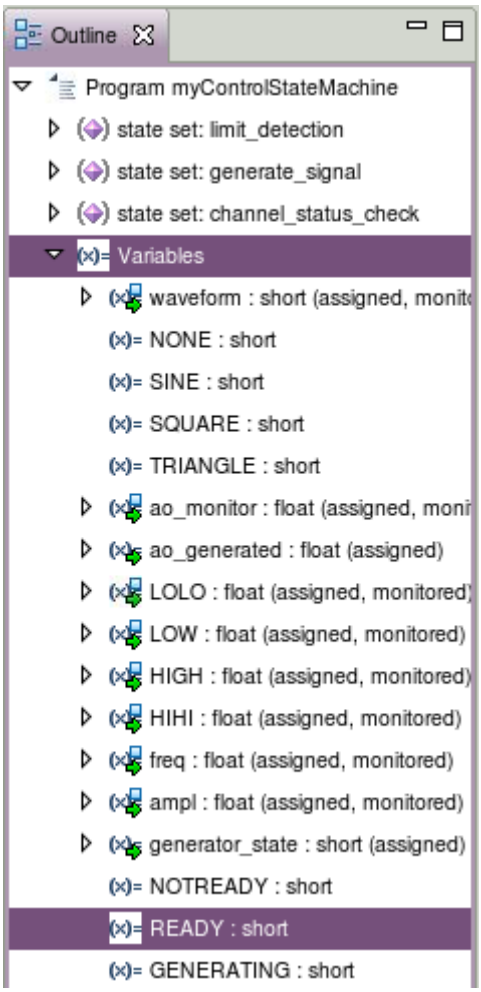


3.4.18 SNL02 – SNL: Outline View

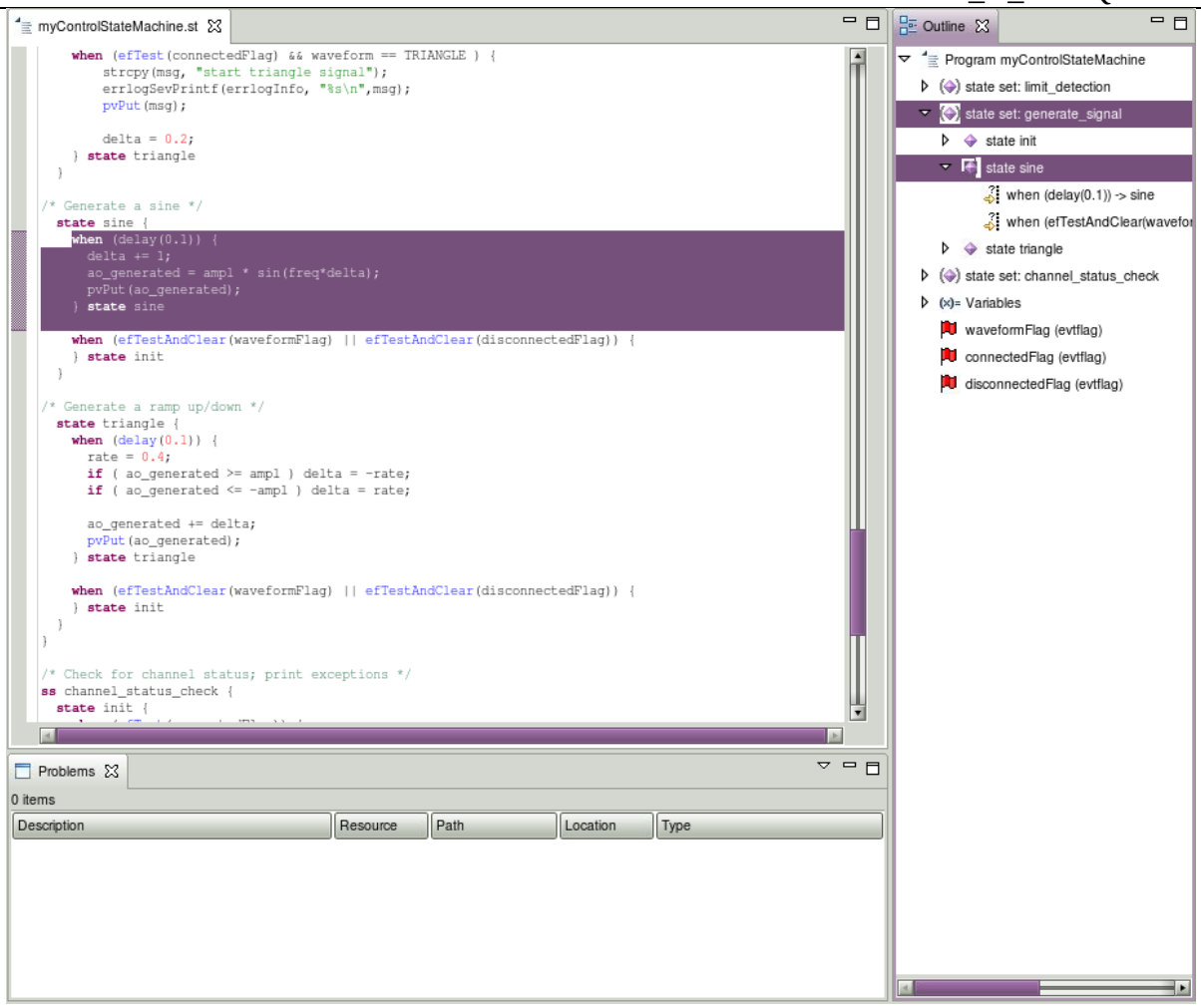
Prerequisite	1. IOC running 2. CSS started 3. SNL Development Perspective and SNL Editor opened
Test Cases	1. Positive execution of the Outline View
Procedure	In CSS, position the cursor in the Outline View and see how it is reflected in the Editor: 1. In the Outline View, browse Program myControlStateMachine -> state set: generate_signal -> state sine -> when (delay(0.1)) -> sine



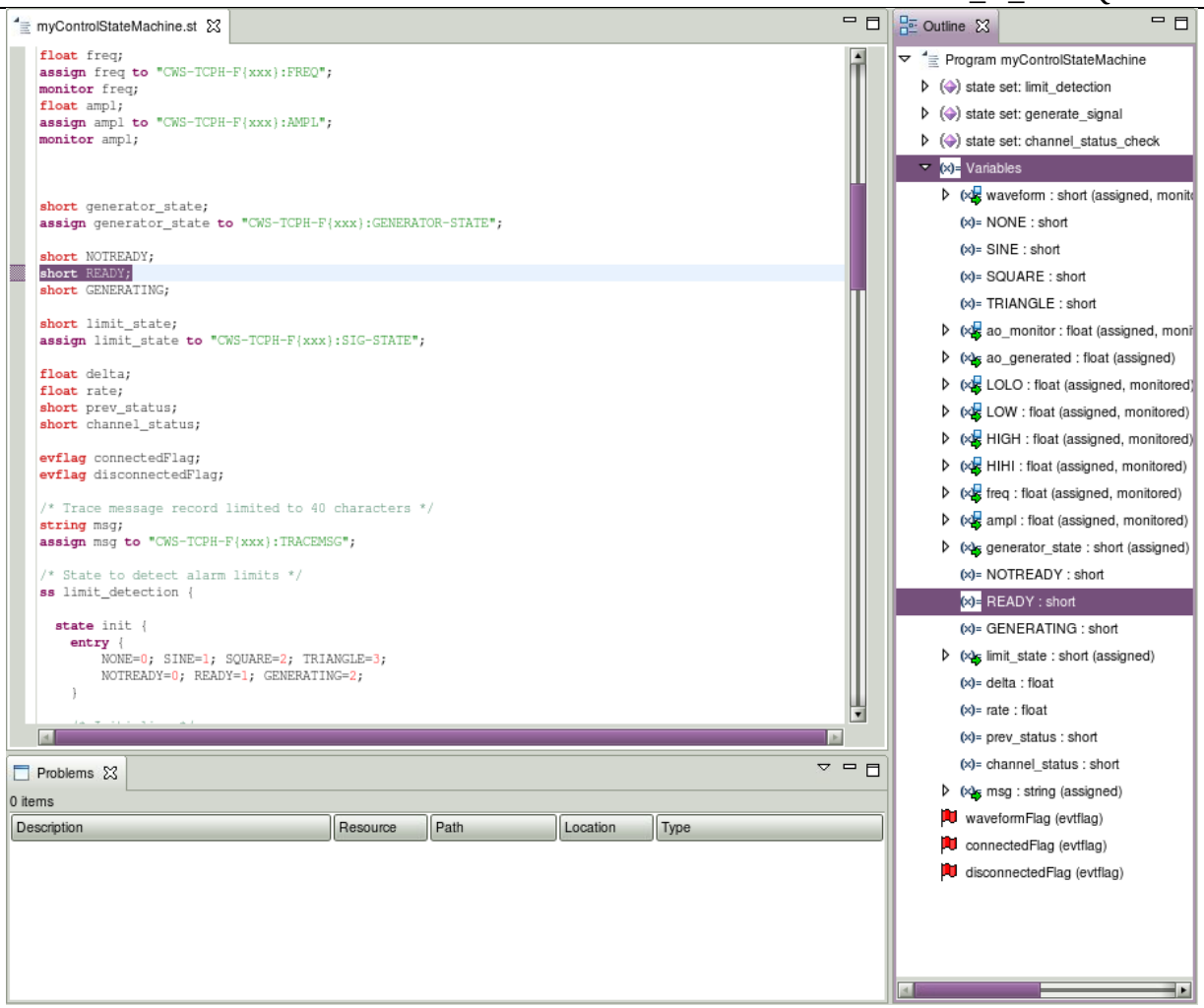
2. From the Outline View, browse (x)= Variables -> (x)= READY : short



Pass Criteria	1. The “when” condition block should be highlighted:	



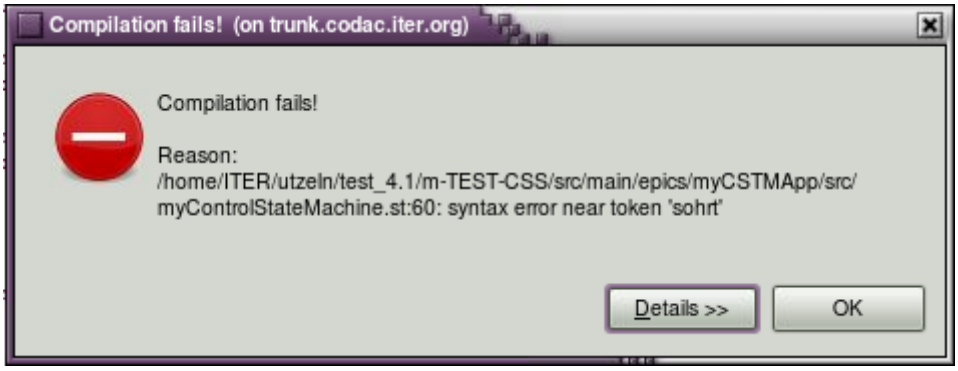
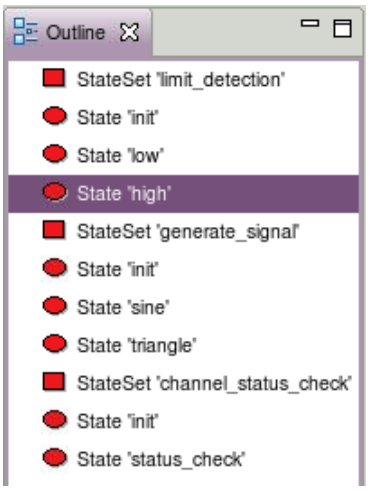
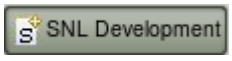
2. In the SNL Editor the cursor should point in the declaration part to the specified constant:



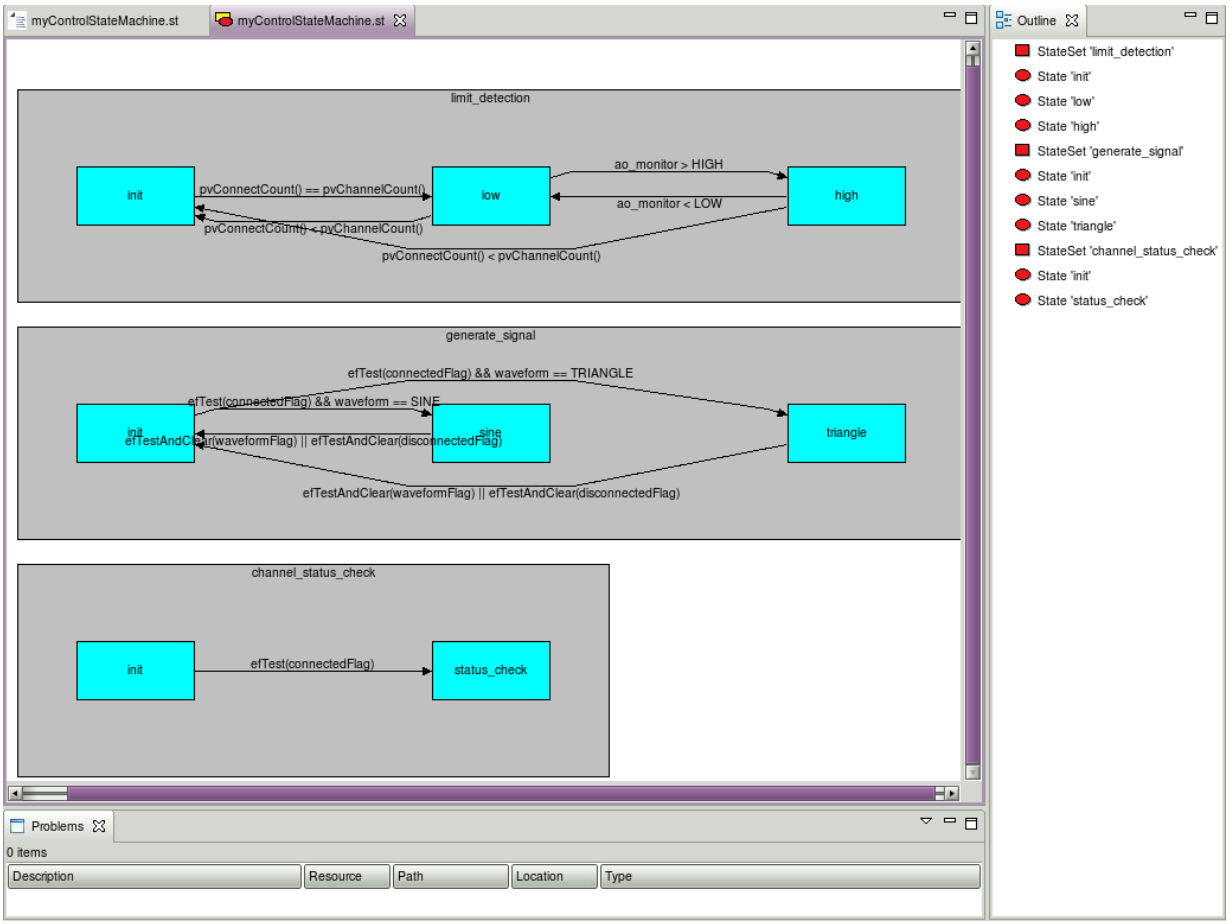
3.4.19 SNL03 – SNL: pre-compilation

Prerequisite	1. IOC running 2. CSS started 3. SNL Development Perspective and SNL Editor opened
Test Cases	1. Positive execution of the CSS SNL pre-compilation
Procedure	In CSS, enter bad syntax in the SNL code after the constant “short READY;”: 1. sohrt UNKNOWN; 2. Save the state machine code by <CTRL>+S 3. Fix the error as follow: short UNKNOWN; 4. Save the code by <CTRL>+S

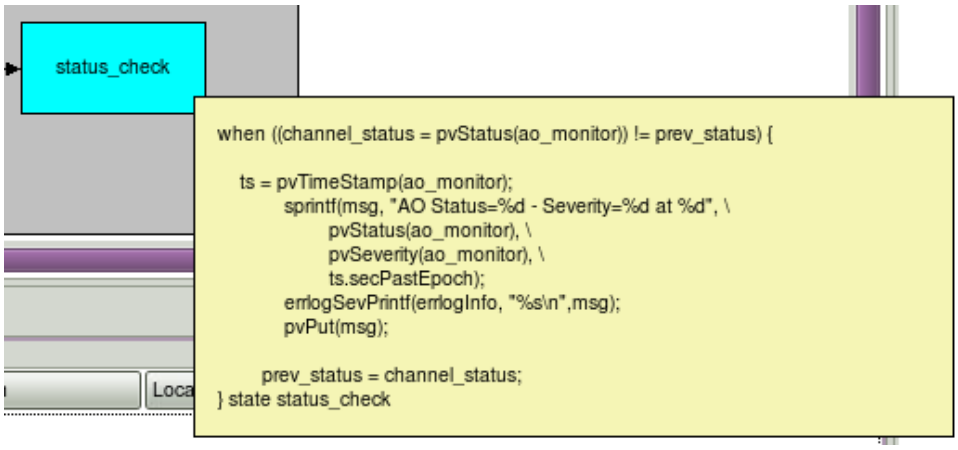
Pass Criteria	1. The absent keyword “sohrt” should NOT be highlighted:
	`short NOTREADY;`
	`short READY;`
	`sohrt UNKNOWN;`

	2. The pre-compilation should raise an exception:  3. The correct keyword is highlighted: `short UNKNOWN;` 4. No exception raised during the pre-compilation	
	3.4.20 SNL04 – SNL: Diagram Editor	
Prerequisite	1. IOC running 2. CSS started 3. SNL Development Perspective and SNL Editor opened	
Test Cases	1. Positive execution of the CSS SNL Diagram Editor	
Procedure	In CSS, start the SNL Diagram Editor: 1. From the Navigator View, right-click on myControlStateMachine.st -> Open With -> SNL Diagram Editor 2. Move the mouse over channel_status_check -> status_check 3. From the Outline View, click on State “High”:  4. Close the SNL Diagram Editor  and the SNL Editor  . Close the Perspective  via a right-click.	

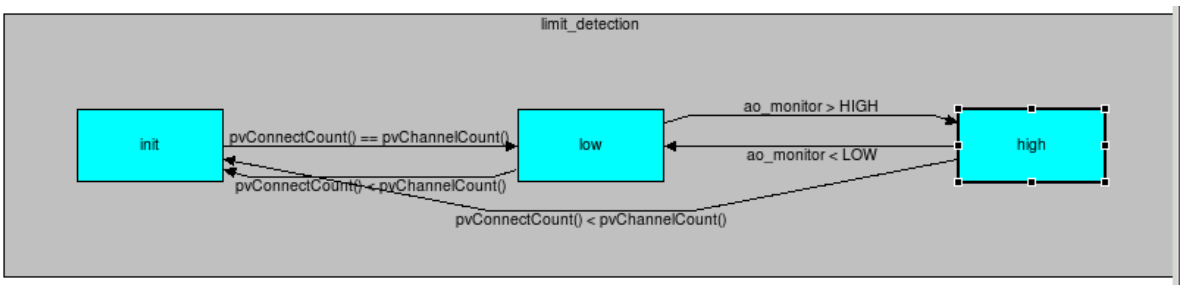
Pass Criteria	1. The SNL Diagram Editor should open:



2. The code of status_check state should be displayed:



3. The state selected in the Outline View is also selected in the SNL Diagram Editor:



	3.4.21 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 Stop the demo archive engine and alarm server: 3. $ ps –ef|grep demo|grep <username? $ kill -9 <PID1> < PID2> <PID3>…		

3.5 Component Test Log

	3.5.1 INT01 - Welcome	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.2 DSP01 - Display Tool: PV Table	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.3 DSP02 - Display Tool: PACE Editor	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.4 DGC01 – Diagnostic Tool: EPICS PV Tree	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.5 DGC02 – Diagnostic Tool: PV Fields Viewer	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.6 DGC03 – Diagnostic Tool: Probe	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		

	3.5.7 DGC04 – Diagnostic Tool: Post Analyzer	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.8 DBG01 – Debugging Tool: RDB Shell	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.9 DSP03 - Display Tool: RDB Table Editor	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.10 DBG02 – Debugging Tool: JMS Monitor	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.11 UTY01 – Utilities: CA Snooper	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.12 UTY02 – Utilities: System Monitor	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		

	3.5.13 UTY03 – Utilities: Create Log Entry	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.14 UTY04 – Utilities: Search Logbook	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.15 UTY05 – Utilities: Clock	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.16 UTY06 – Utilities: Therapist	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.17 SNL01 – SNL: SNL development Perspective	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.18 SNL02 – SNL: Outline View	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		

	3.5.19 SNL03 – SNL: pre-compilation	[PASS / FAIL]
[Bug ID]	[Bug title to briefly describe the anomaly]	
Remarks		
	3.5.20 SNL04 – SNL: Diagram Editor	[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?	