CS-Studio Display Builder

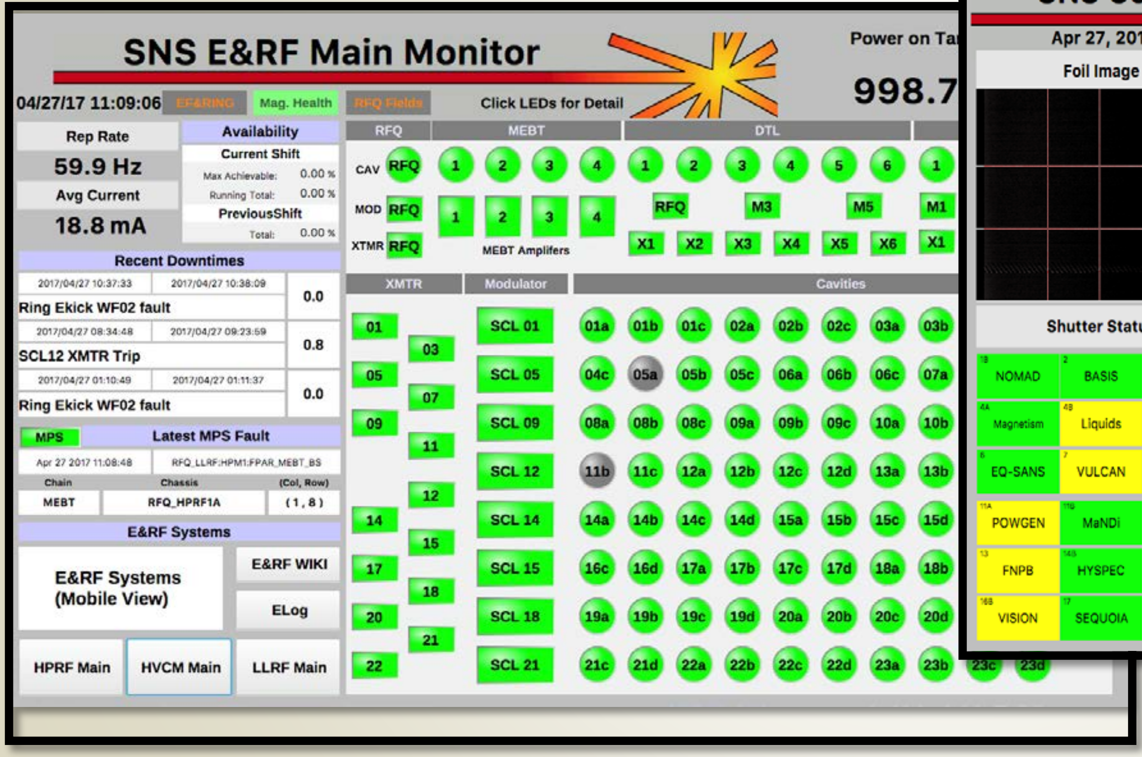
Tutorial presented:
Spring 2017 EPICS Collaboration Meeting at KURRI, Osaka, Japan

Megan Grodowitz,
Kay Kasemir (kasemir@ornl.gov)

Overview

- Display Builder replaces OPI Builder (BOY) in CS-Studio
  - Built on Eclipse plugin framework as both a user interface and IDE
  - Editor to to create user interfaces
  - Runtime execute user interfaces
  - Backward compatible, can import most *.opi files
  - 30+ widget types at this time
    - Simple *data read/write widgets* like gauges and input fields
    - Complex *interactive widgets* like custom graphs or a web browser
    - *Structural and design widgets* like text labels, images, and groups
    - Customize widgets for your site using *widget style classes*
  - Scripting support for custom behaviors
  - Integrated with other CS-Studio tools, e.g. Data Browser

Examples: SNS Accelerator



SNS E&RF Main Monitor

04/27/17 11:09:06

Power on Ta 998.7

Rep Rate	Availability	
59.9 Hz	Current Shift	
Avg Current	Max Achievable:	0.00 %
18.8 mA	Running Total:	0.00 %
	PreviousShift	
	Total:	0.00 %

Recent Downtimes

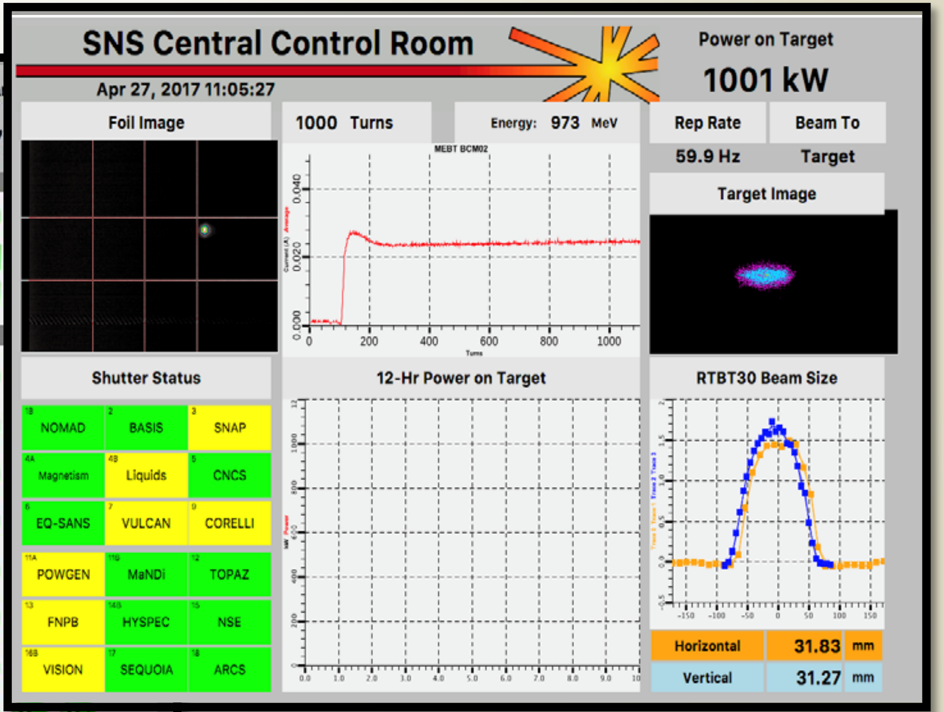
2017/04/27 10:37:33	2017/04/27 10:38:09	0.0
Ring Ekick WF02 fault		
2017/04/27 08:34:48	2017/04/27 09:23:59	0.8
SCL12 XMTR Trip		
2017/04/27 01:10:49	2017/04/27 01:11:37	0.0
Ring Ekick WF02 fault		

Latest MPS Fault

Apr 27 2017 11:08:48	RFQ_LLRF:HPM1:FPAR_MEBT_BS	
Chain	Chassis	(Col, Row)
MEBT	RFQ_HPRF1A	(1 , 8)

E&RF Systems

E&RF Systems (Mobile View)



SNS Central Control Room

Apr 27, 2017 11:05:27

Power on Target 1001 kW

1000 Turns

Energy: 973 MeV

Rep Rate	Beam To
59.9 Hz	Target

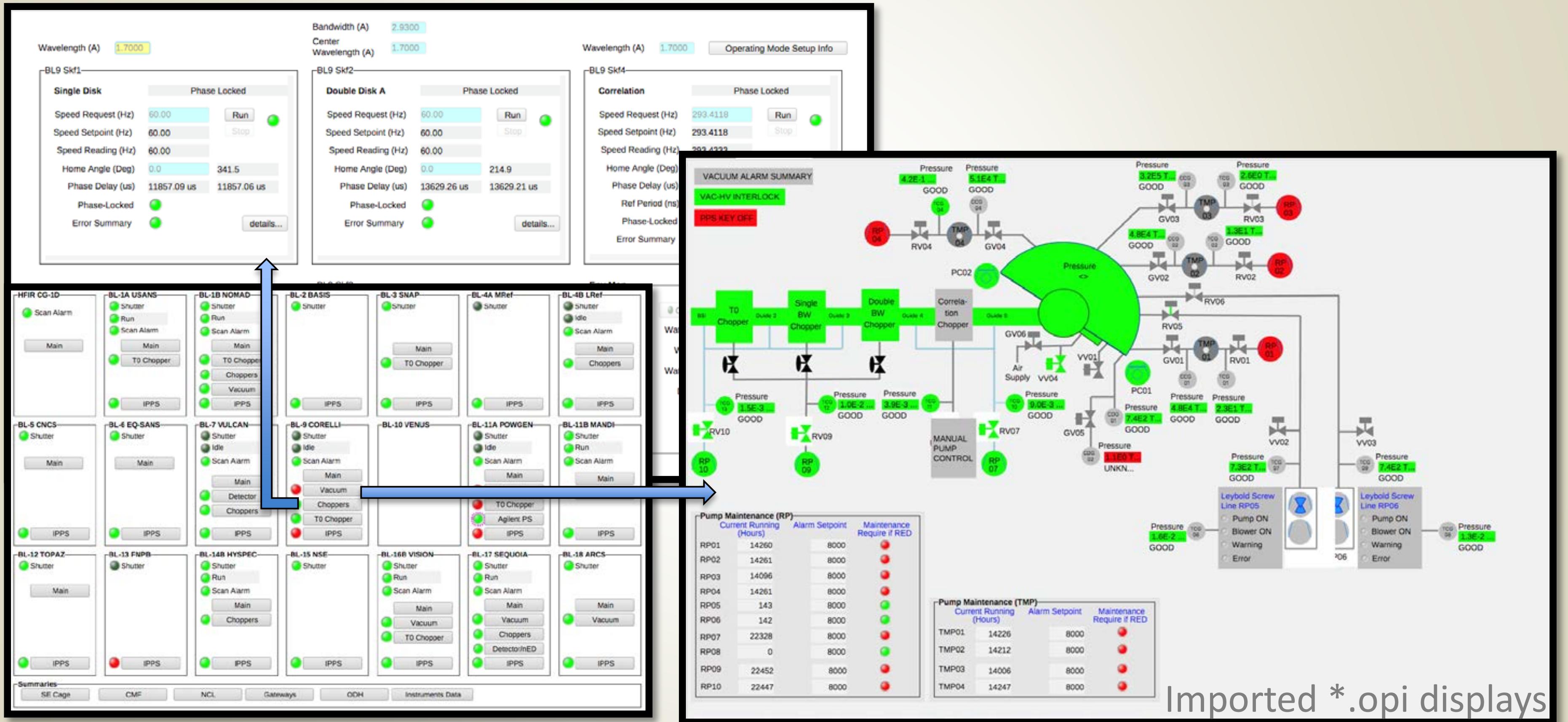
Shutter Status

NOMAD	BASIS	SNAP
Magnetism	Liquids	CNCS
EQ-SANS	VULCAN	CORELLI
POWGEN	MaNDi	TOPAZ
FNPB	HYSPEC	NSE
VISION	SEQUOIA	ARCS

Horizontal	31.83	mm
Vertical	31.27	mm

Imported *.opi displays

Examples: SNS Instruments



Wavelength (A) 1.7000
Bandwidth (A) 2.9300
Center Wavelength (A) 1.7000
Wavelength (A) 1.7000
Operating Mode Setup Info

BL9 Skf1 — Single Disk — Phase Locked

Speed Request (Hz)	60.00	
Speed Setpoint (Hz)	60.00	
Speed Reading (Hz)	60.00	
Home Angle (Deg)	0.0	341.5
Phase Delay (us)	11857.09 us	11857.06 us
Phase-Locked	●	
Error Summary	●	

BL9 Skf2 — Double Disk A — Phase Locked

Speed Request (Hz)	60.00	
Speed Setpoint (Hz)	60.00	
Speed Reading (Hz)	60.00	
Home Angle (Deg)	0.0	214.9
Phase Delay (us)	13629.26 us	13629.21 us
Phase-Locked	●	
Error Summary	●	

BL9 Skf4 — Correlation — Phase Locked

Speed Request (Hz)	293.4118
Speed Setpoint (Hz)	293.4118
Speed Reading (Hz)	293.4233
Home Angle (Deg)	
Phase Delay (us)	
Ref Period (ns)	
Phase-Locked	
Error Summary	

Pump Maintenance (RP)

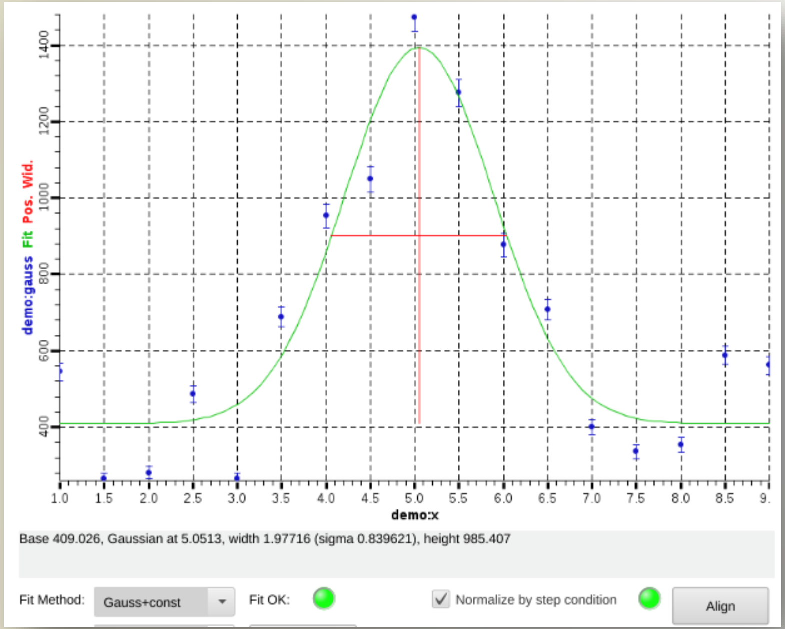
	Current Running (Hours)	Alarm Setpoint	Maintenance Require if RED
RP01	14260	8000	🔴
RP02	14261	8000	🔴
RP03	14096	8000	🔴
RP04	14261	8000	🔴
RP05	143	8000	🟢
RP06	142	8000	🟢
RP07	22328	8000	🔴
RP08	0	8000	🟢
RP09	22452	8000	🔴
RP10	22447	8000	🔴

Pump Maintenance (TMP)

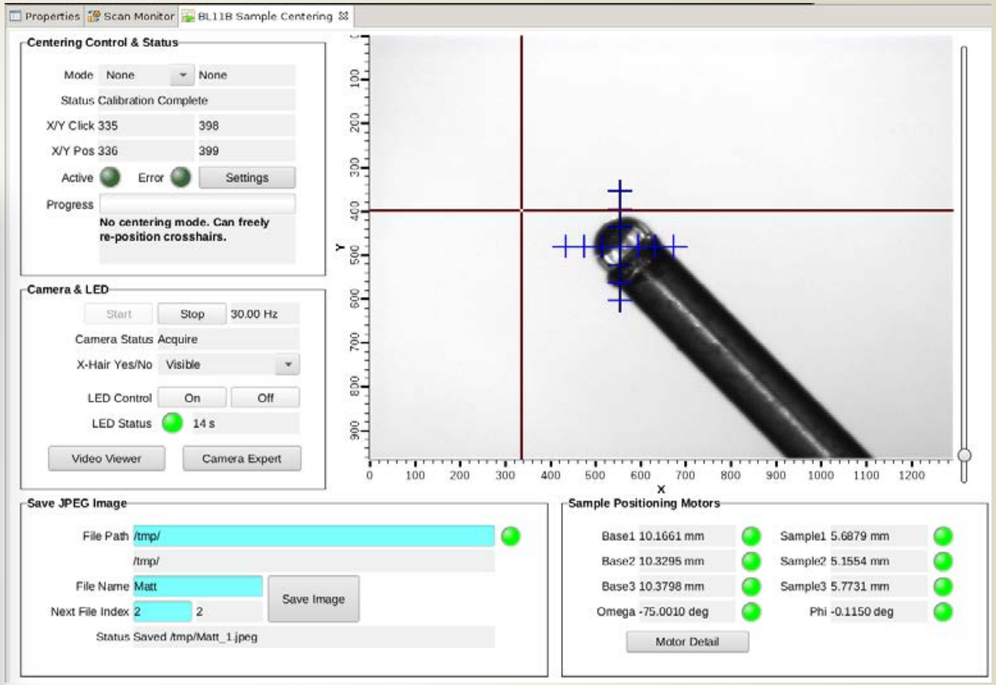
	Current Running (Hours)	Alarm Setpoint	Maintenance Require if RED
TMP01	14226	8000	🔴
TMP02	14212	8000	🔴
TMP03	14006	8000	🔴
TMP04	14247	8000	🔴

Imported *.opi displays

More Examples



Base 409.026, Gaussian at 5.0513, width 1.97716 (sigma 0.839621), height 985.407



Mode	None	None
Status	Calibration Complete	
X/Y Click	335	398
X/Y Pos	336	399
Active	Error	
Progress	No centering mode. Can freely re-position crosshairs.	

Frequency	30.00 Hz
Camera Status	Acquire
X-Hair Yes/No	Visible
LED Control	On / Off
LED Status	14 s

File Path	/tmp/
	/tmp/
File Name	Matt
Next File Index	2 / 2
Status	Saved /tmp/Matt_1.jpeg

Base1	10.1661 mm	Sample1	5.6879 mm
Base2	10.3295 mm	Sample2	5.1554 mm
Base3	10.3798 mm	Sample3	5.7731 mm
Omega	-75.0010 deg	Phi	-0.1150 deg

- Scan UI
- Interactive Alignment

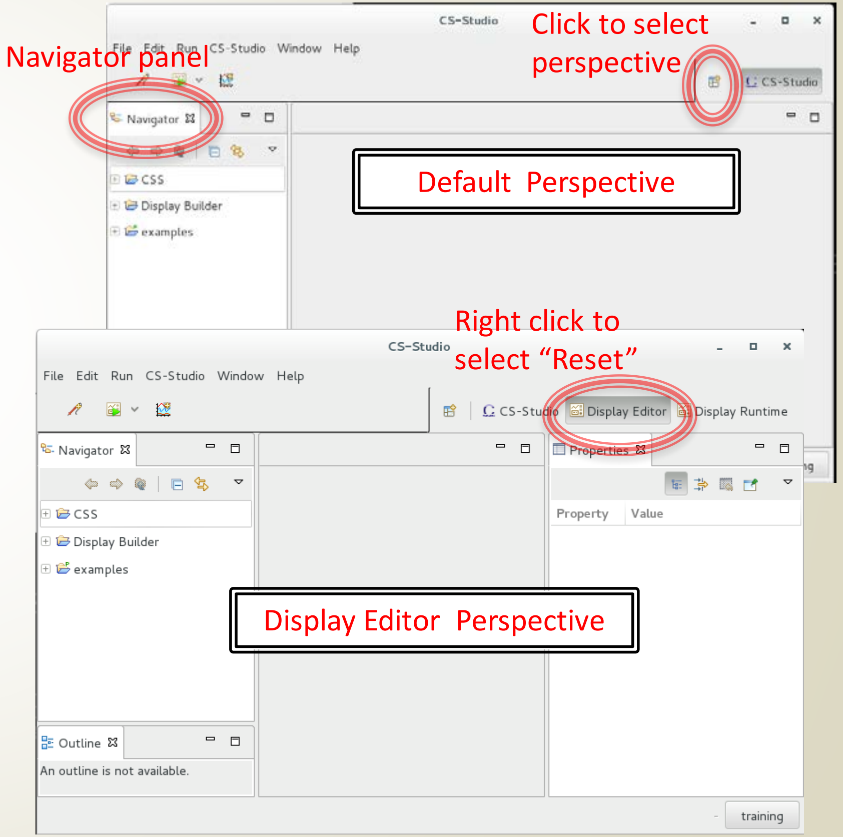
New *.bob displays

CS-Studio

- Display Builder is a component of Control System Studio
  - http://controlsystemstudio.org/
- CS-studio can be downloaded from the location above, or through one of the site specific download areas
- CS-Studio will run on any Windows/Linux/OSX with a supported version of Java
- This tutorial is based on a Linux virtual machine image running the SNS version of CS-Studio

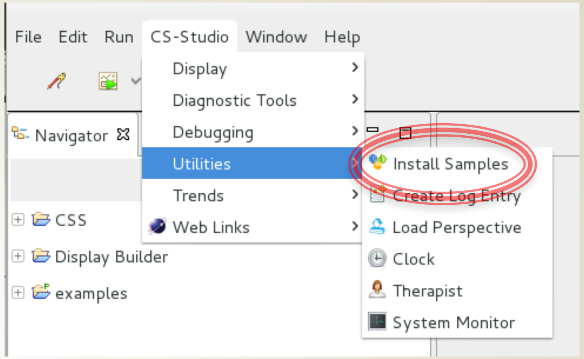
Getting Around in CS-Studio

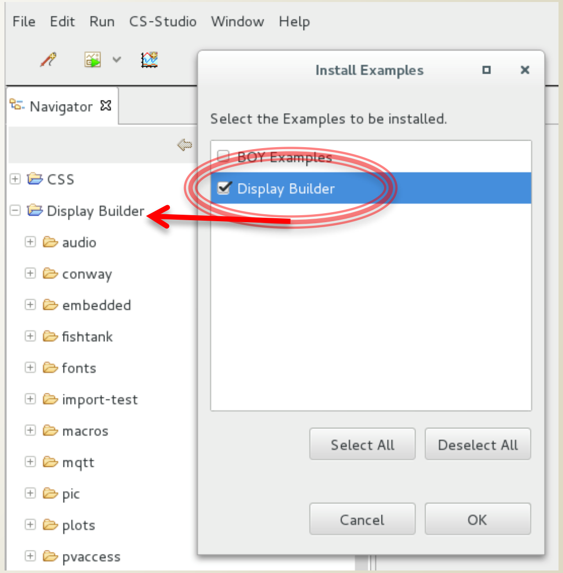
- On Linux, launch CS-Studio with the "css" command
  - A dialog will pop up and ask which workspace to use. For now, leave it as the default selection
- CSS should open up in the default "CS-Studio" Perspective
  - If you see a welcome screen, close it using the small 'X' in the corner of the panel
- All of the sub-windows in Eclipse are called Panels
  - This screen has one open panel: The Navigator
  - Panels can be dragged around and rearranged as you work on a project
- Perspectives define a set layout of panels and controls for a given type of work
  - Right click on perspective name to select "Reset"
  - Display Editor Perspective
    - Open Display Editor perspective with the perspective selection button
    - We will be in this perspective for the entire tutorial
    - Sets up editor area with navigator, outline, and properties panels arranged around it.
  - Display Runtime Perspective
    - We will not need this perspective during the tutorial
    - Does not include file navigator and other tools
    - Meant to be open in production once "Top Displays" have been defined



Display Editor Setup

- Open the “Display Editor” perspective
  - We will be in this perspective during the tutorial
- Install display builder examples
  - Use the top menus to select CS-Studio -> Utilities -> Install Samples
  - Select “Display Builder” and hit OK
  - This will download a number of examples to look at for ideas and techniques
  - This has already been done in the tutorial VM





Display Editor: Basics

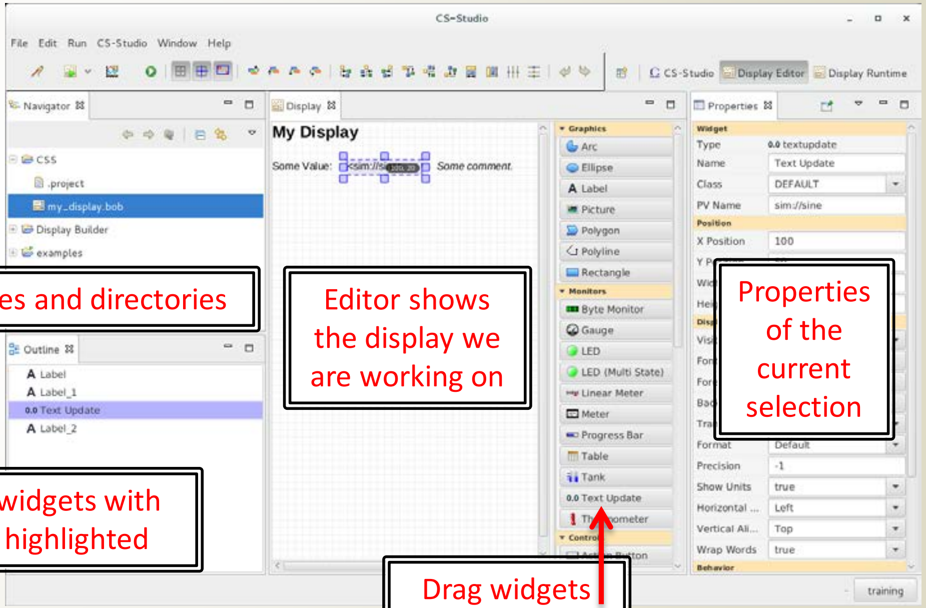
- Focus on function: Edit and Run displays connected to PVs
- Common control system elements should be as easy as 1-2-3:
    1. Drag and drop a "Text Update" widget onto your display in the editor
    2. Assign an EPICS process variable (PV) to your widget in the editor
    3. Hit "Run" and see your display in the runtime

What you will get:

✓ PV *value* with *units* as text
✓ PV *severity* reflected in border color
✓ PV *name* and *value* shown in tool-tip
✓ Indication of 'disconnected' state via a pink border

Walkthrough: My first display

- Create new Display
  - Select File -> New
  - This will open a “Select a Wizard” dialog
  - Choose Display Editor -> New Display
  - Click Next
  - Name the file “my_display.bob” and choose the /CSS container
  - Click Finish



Navigator:
- CSS
  - .project
  - my_display.bob
- Display Builder
- examples

Display: My Display — Some Value: <sim://sine> Some comment.

Outline:
- Label
- Label_1
- 0.0 Text Update
- Label_2

Properties:

Widget	
Type	0.0 textupdate
Name	Text Update
Class	DEFAULT
PV Name	sim://sine
Position	
X Position	100
Format	Default
Precision	-1
Show Units	true
Horizontal ...	Left
Vertical Ali...	Top
Wrap Words	true
Behavior	

Navigator shows files and directories

Editor shows the display we are working on

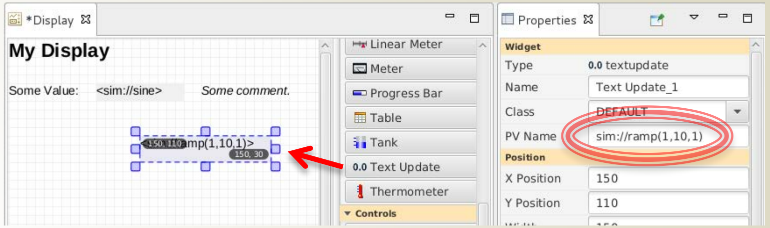
Properties of the current selection

Outline shows all widgets with current selection highlighted

Drag widgets from the palette onto the display

My First Display: Add Some Widgets

- Add a ramp value with text output
  - L-Click and hold to grab and drag a “Text Update” widget from the palette onto the display
  - With the new widget selected, slect the “PV Name” field in the properties and type “sim://ramp(1,10,1)”
  - This means to add a value that counts to 10 at 1 second intervals (more on sim PVs later)
- Add a toggle button to input a value
  - Drag a “Boolean Button” widget onto the display
  - Select “PV Name” and type “loc://test”
  - This makes the boolean button write to a user defined local variable called test. It toggles one bit of the variable between 1 and 0.
- Add an LED to read from the test variable
  - Drag an “LED” widget onto the display
  - Select “PV Name” and type “loc://test”
  - This makes the LED light up when any bit in the test variable is non-zero
- Save File
  - File ->Save
  - OR Ctrl-S

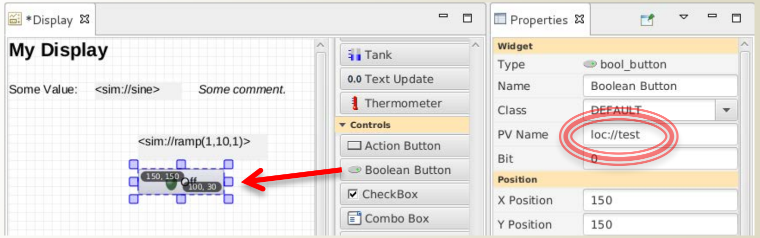


My Display

Some Value: <sim://sine> Some comment.

ramp(1,10,1)>

Widget	
Type	0.0 textupdate
Name	Text Update_1
Class	DEFAULT
PV Name	sim://ramp(1,10,1)
Position	
X Position	150
Y Position	110

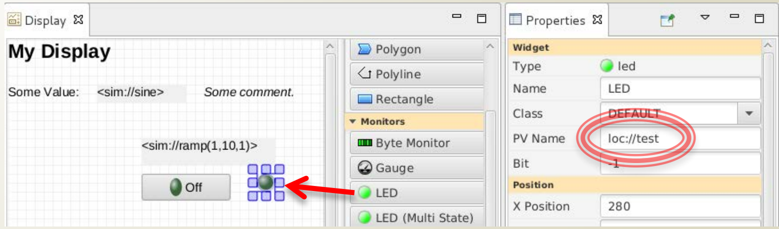


My Display

Some Value: <sim://sine> Some comment.

<sim://ramp(1,10,1)>

Widget	
Type	bool_button
Name	Boolean Button
Class	DEFAULT
PV Name	loc://test
Bit	0
Position	
X Position	150
Y Position	150



My Display

Some Value: <sim://sine> Some comment.

<sim://ramp(1,10,1)>

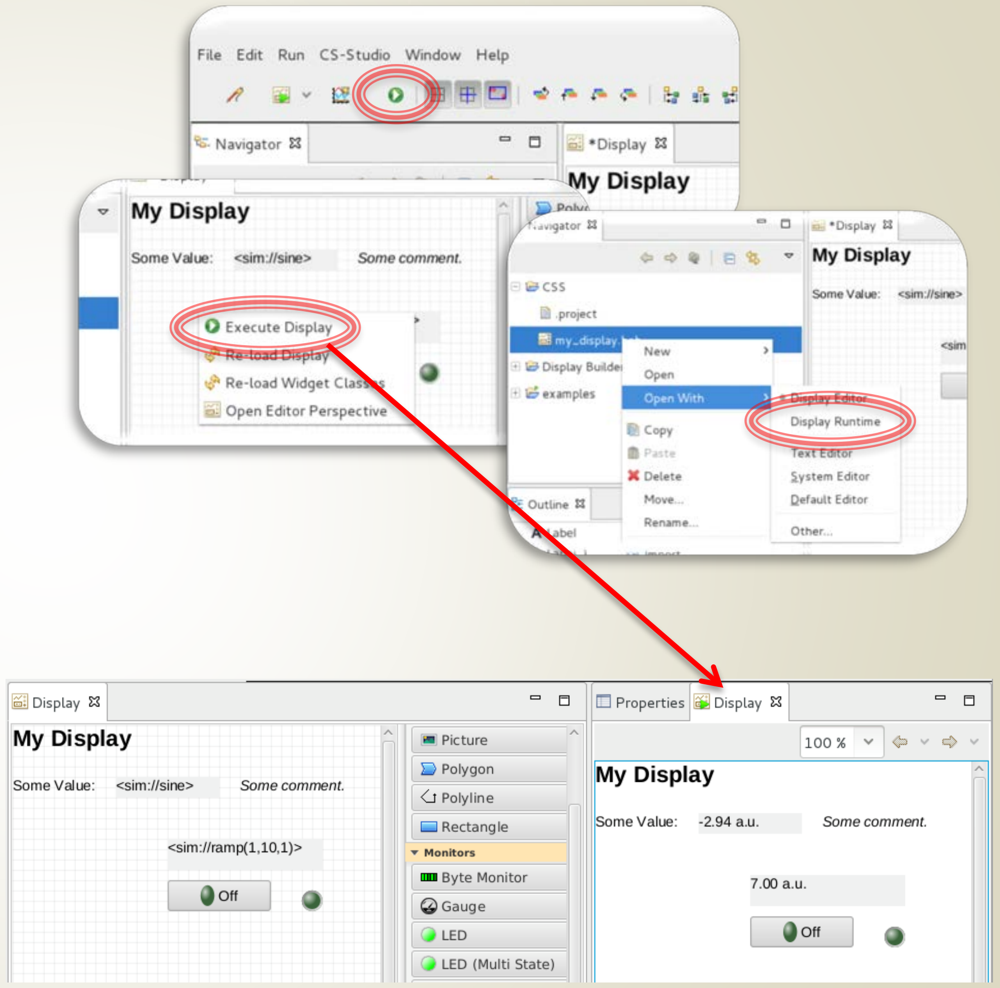
Widget	
Type	led
Name	LED
Class	DEFAULT
PV Name	loc://test
Bit	-1
Position	
X Position	280

My First Display: How to Run

a) Hit the Run button on the control bar
b) OR right click inside the display and “Execute Display” (will also save)
c) OR Right click on my_display.bob -> Open With -> Display Runtime
d) OR execute in standalone runtime, outside of CS-Studio

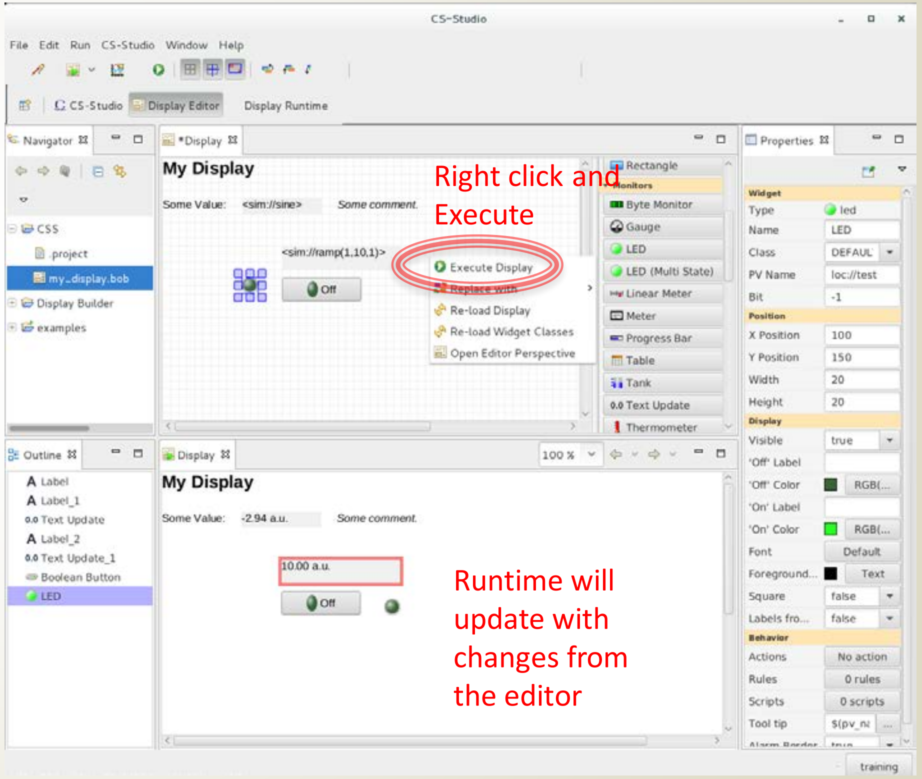
Runtime panel will open to the right side, over the properties panel

- Watch your ramp value count to 10
- The ramp value will show a single solid outline for warning and double outline for severe at the high and low end of its range
- Click the button to turn the LED on and off by changing the value of “test”



Arranging Panels for Display Testing

- By default the runtime panel opens on the right side, but can be re-arranged.
- Drag the runtime down to the bottom below the editor
- Make some change to your display in the editor
- Right click on the display in the editor, select “Execute Display”
  - The display file will be saved
  - The runtime display will update

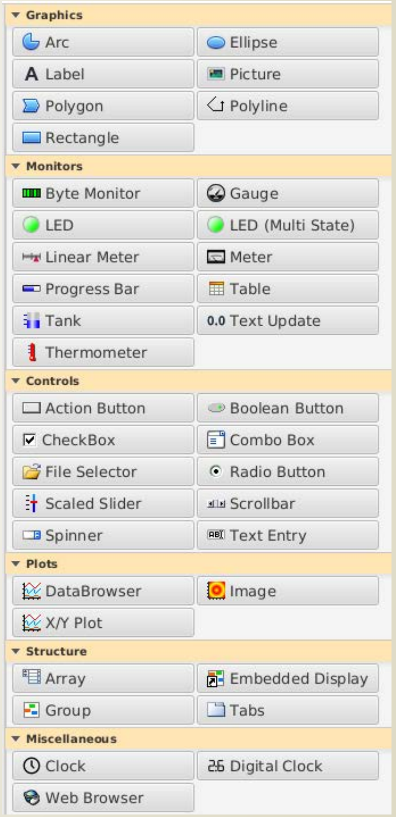


Widget	
Type	led
Name	LED
Class	DEFAUL
PV Name	loc://test
Bit	-1
Position	
X Position	100
Y Position	150
Width	20
Height	20
Display	
Visible	true
'Off' Label	
'Off' Color	RGB(...
'On' Label	
'On' Color	RGB(...
Font	Default
Foreground...	Text
Square	false
Labels fro...	false
Behavior	
Actions	No action
Rules	0 rules
Scripts	0 scripts
Tool tip	$(pv_n

Editor: Using the Widget Palette

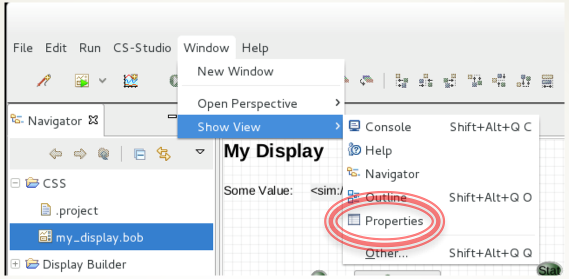
- Add widget to display:
  a) Drag widget from palette into editor
  b) Select widget in palette, then ‘rubberband’ location and size in editor

- Widgets are arranged by section
  – Graphics are static images
  – Monitors change in response to changes in a connected PV (read-only)
  – Controls make changes to PVs, and may also change in response to changes in the PV (read and write)
  – Plots are more complex images based on PV values
  – Structures are used to make more advanced display elements, such as groups of widgets or embedded displays
  – And then some miscellaneous elements…
- Hover the mouse over a widget to see a description tooltip

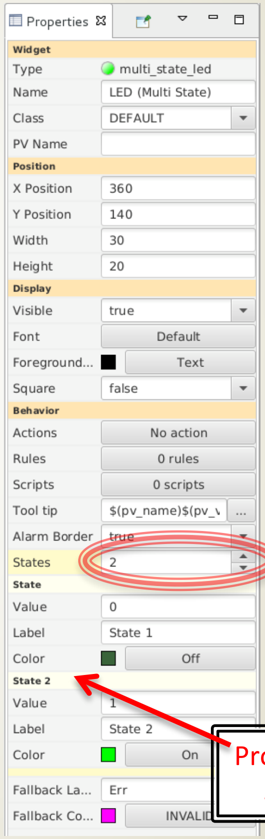


Editor: Customizing Widget Properties

- Properties panel allows editing widget attributes
  - Opens automatically in Editor Perspective
  - If it is closed, to open it:
    - Reset Editor Perspective
    - OR Click Window -> Show View -> Properties

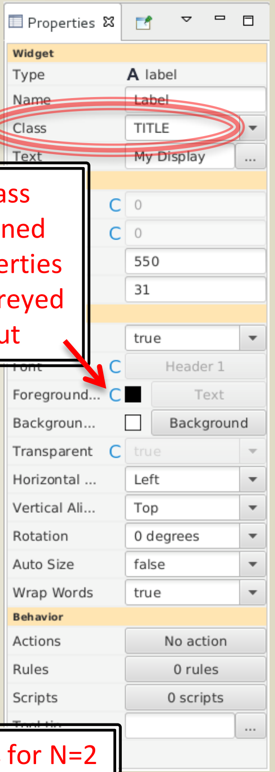


- Properties change depending on which widget is selected
  - Click on the display file name to see file properties
  - Click on the background of the display to see display properties
- Widget class defines property defaults (more on classes later)
- Properties panel is dynamic. For example, changing number of states in multistate LED will add more fields to edit state attributes



Properties

Widget	
Type	multi_state_led
Name	LED (Multi State)
Class	DEFAULT
PV Name	
Position	
X Position	360
Y Position	140
Width	30
Height	20
Display	
Visible	true
Font	Default
Foreground...	■ Text
Square	false
Behavior	
Actions	No action
Rules	0 rules
Scripts	0 scripts
Tool tip	(pv_name)(pv_v
Alarm Border	true
States	2
State	
Value	0
Label	State 1
Color	■ Off
State 2	
Value	1
Label	State 2
Color	🟩 On
Fallback La...	Err
Fallback Co...	■ INVALID



Properties

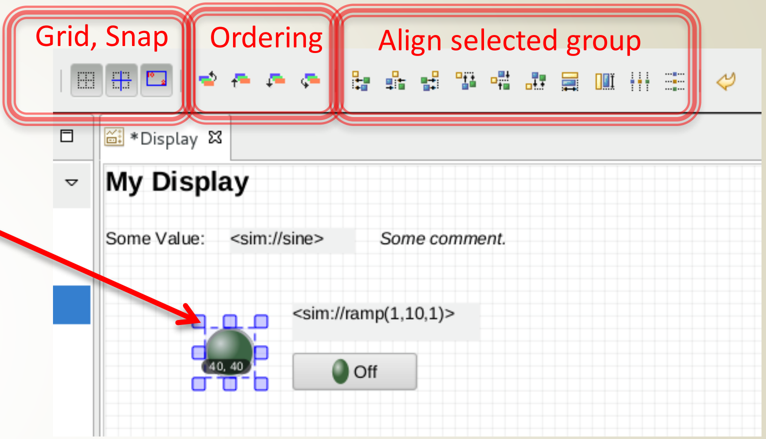
Widget	
Type	label
Name	Label
Class	TITLE
Text	My Display
	C 0
	C 0
	550
	31
	true
Font	C Header 1
Foreground...	C ■ Text
Backgroun...	□ Background
Transparent	C true
Horizontal ...	Left
Vertical Ali...	Top
Rotation	0 degrees
Auto Size	false
Wrap Words	true
Behavior	
Actions	No action
Rules	0 rules
Scripts	0 scripts
Tool tip	

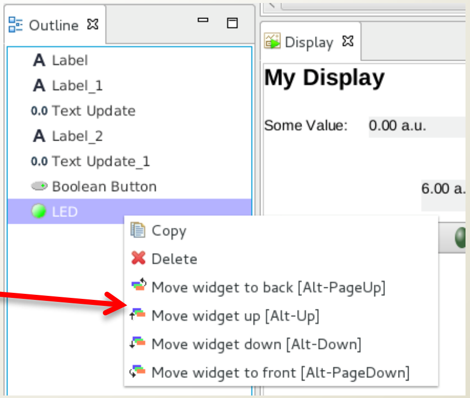
Class defined properties are greyed out

Properties for N=2 states appear

Editor: Manipulating Widgets

- Select and in editor using left click and drag
- Resize in editor by left click and drag edges of outline box
- Use alignment tools to help with arrangements
  - Grid
  - Snap to Grid
  - Snap to Geometry (sides and sizes of other widgets)
  - Arrange group of widgets
- Arrange front to back
  - Use alignment tools for widget ordering
  - OR select widget in outline
    - Right click for context menu and hotkey combos
    - alt+up or alt-down to move forward or back



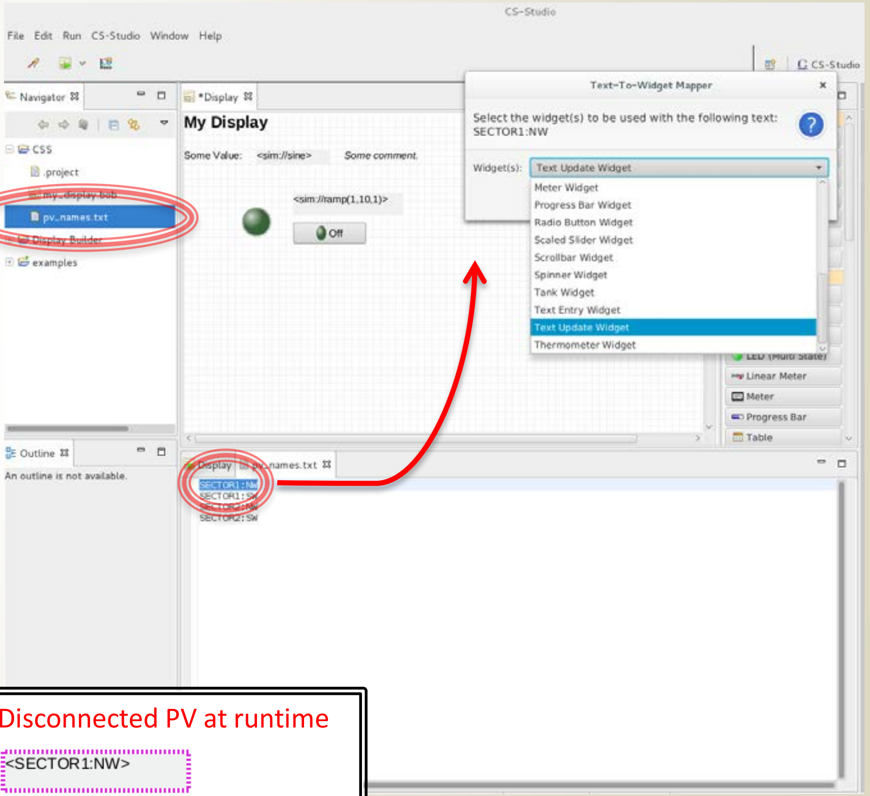


PV Names and Examples

- `ca://some_pv_name`
  - EPICS Channel Access PV
- `some_pv_name`
  - Typically same, since “ca://” is the default
- `sim://sine`
  - Simulated PV. Read online help for details
- `loc://x(4)`
  - Local PV. Read online help for details
- `pva://x`
  - EPICS V4 pvAccess
- `mqtt://x`
  - MQTT PV

Editor: Drag and Drop Text

- CS-Studio includes drag and drop functionality
  - Text can be dropped into display
  - Prompt will ask about Widget type
- Exercise: A file full of process variable names
  - Make a new text file in Eclipse
    - File -> New then select General -> File
    - Name the file pv_names.txt and select /CSS parent folder
  - Arrange text file below Display Editor panel
  - Type some lines of text to be PV names
  - Drag text from file into display editor, it will give prompt for widget type
  - Make a text update widget
- What happens when we run with this text update widget assigned to our unqualified PV name?
  - The default PV type is channel access (ca://)
  - This txt update will show the PV disconnected, unless you typed a PV name that is reachable from your tutorial machine using channel access



Text-To-Widget Mapper

Select the widget(s) to be used with the following text: SECTOR1:NW

Widget(s): Text Update Widget

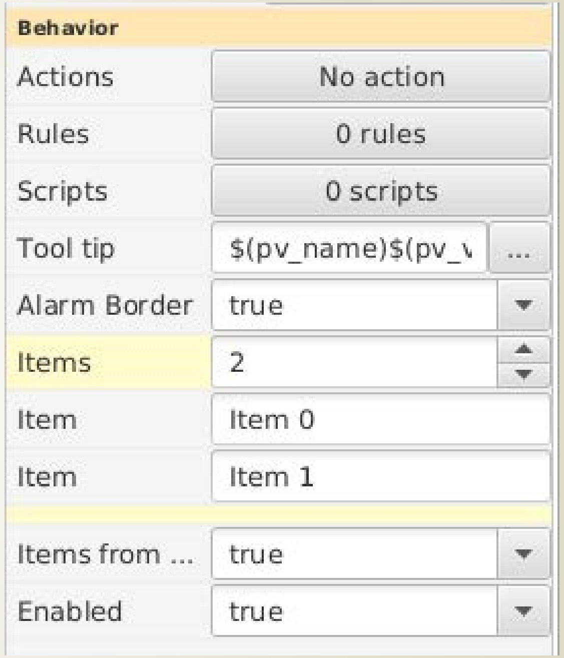
- Meter Widget
- Progress Bar Widget
- Radio Button Widget
- Scaled Slider Widget
- Scrollbar Widget
- Spinner Widget
- Tank Widget
- Text Entry Widget
- Text Update Widget
- Thermometer Widget

Disconnected PV at runtime

<SECTOR1:NW>

Editor: Widget Behaviors

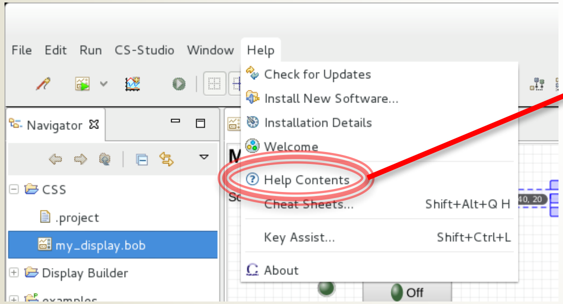
- Widget behaviors allow control over more complex functionality
  - Later exercises will cover these in more detail
- Actions
  - Widgets can have assigned actions
  - Usually for button press
- Rules and Scripts
  - Widgets can have arbitrary scripts attached to them that alter widget properties Rules are a safe way to generate common script actions
- Tooltips
  - Set the text that appears when hovering over a widget
- Alarm Border
  - Only for widgets that can have a PV assigned to them
  - Disable or enable border for alarm states
- Items, Items from…
  - This is used for multi-choice widgets like a combo box
  - To get items from an enum PV type, assign the PV to the widget and set "Item from.." to true
- Enabled
  - Only for widgets with mouse press functionality (buttons, checkbox, etc)
- Other Behaviors…
  - Various widgets have behaviors regarding limits, min/max values, etc
  - Behaviors with "from…" indicate whether to get these values from the connected PV or not

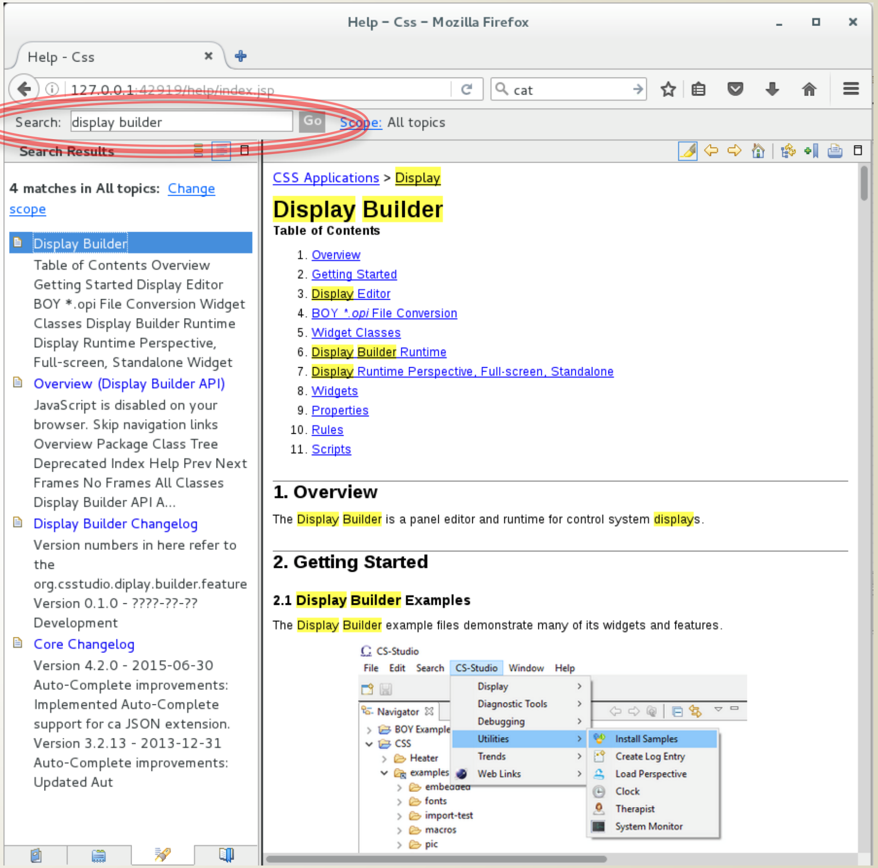


Behavior	
Actions	No action
Rules	0 rules
Scripts	0 scripts
Tool tip	(pv_name)(pv_v …
Alarm Border	true
Items	2
Item	Item 0
Item	Item 1
Items from …	true
Enabled	true

Access Online Help

- Searchable online help is included
- Help -> Help Contents





4 matches in All topics: Change scope

Display Builder
Table of Contents Overview Getting Started Display Editor BOY *.opi File Conversion Widget Classes Display Builder Runtime Display Runtime Perspective, Full-screen, Standalone Widget

Overview (Display Builder API)
JavaScript is disabled on your browser. Skip navigation links Overview Package Class Tree Deprecated Index Help Prev Next Frames No Frames All Classes Display Builder API A...

Display Builder Changelog
Version numbers in here refer to the org.csstudio.diplay.builder.feature Version 0.1.0 - ????-??-?? Development

Core Changelog
Version 4.2.0 - 2015-06-30 Auto-Complete improvements: Implemented Auto-Complete support for ca JSON extension. Version 3.2.13 - 2013-12-31 Auto-Complete improvements: Updated Aut

CSS Applications > Display

Display Builder

Table of Contents

1. Overview
2. Getting Started
3. Display Editor
4. BOY *.opi File Conversion
5. Widget Classes
6. Display Builder Runtime
7. Display Runtime Perspective, Full-screen, Standalone
8. Widgets
9. Properties
10. Rules
11. Scripts

1. Overview

The Display Builder is a panel editor and runtime for control system displays.

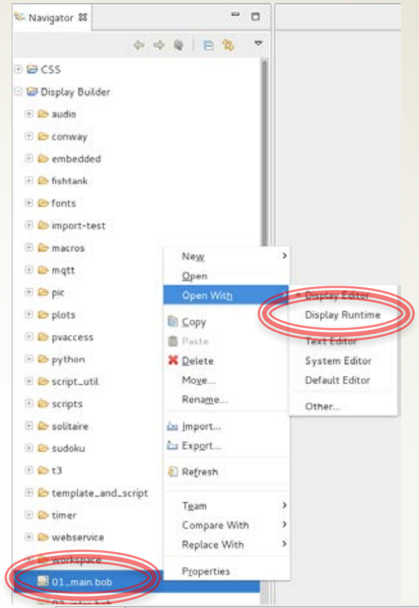
2. Getting Started

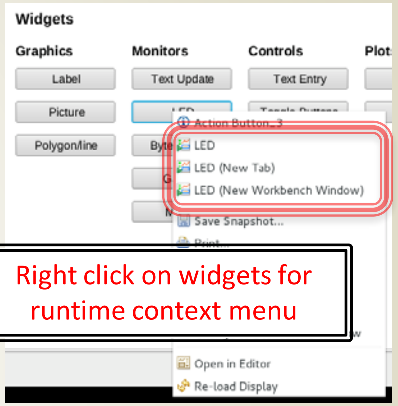
2.1 Display Builder Examples

The Display Builder example files demonstrate many of its widgets and features.

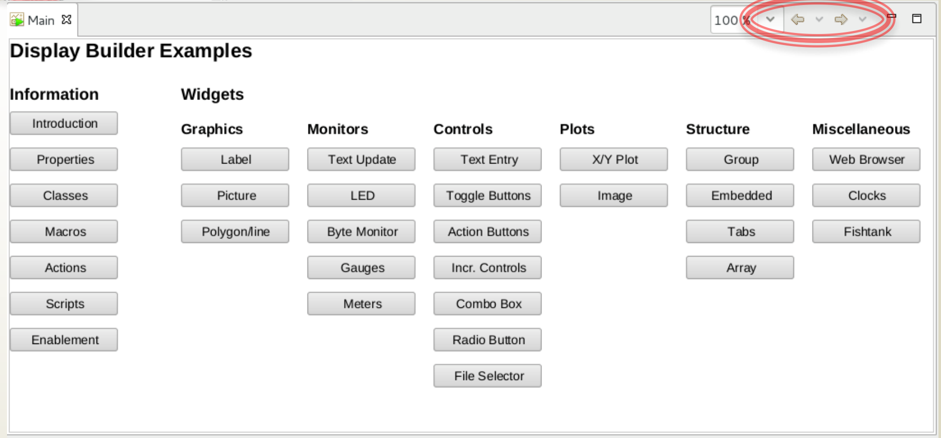
Runtime Panel

- The runtime panel opens when we execute a display
  - With the Display Builder examples installed, navigate to Display Builder -> 01_main.bob
  - Right click and select Open With -> Display Runtime
  - Resize the main runtime panel so that the scroll bars disappear and all the buttons are visible
- Navigation
  - Click buttons to open new displays with examples
  - Use the forward and back arrows to navigate between screens
  - Right click on a button to choose to open in (New tab) or (New workbench window)



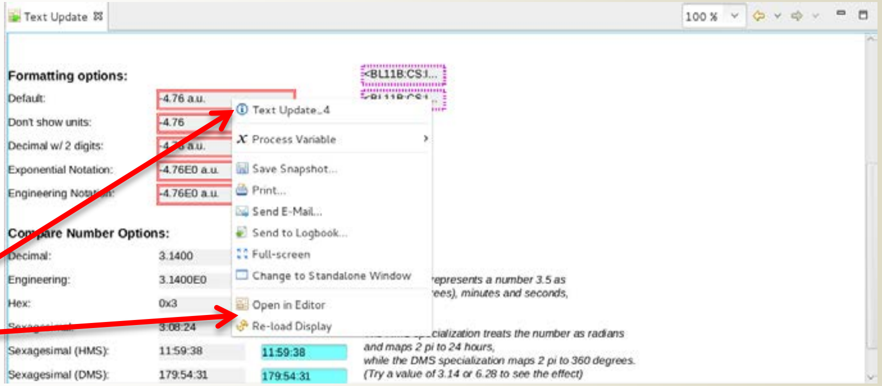


Right click on widgets for runtime context menu



Runtime: Tooltip & Context Menu

- From the main example screen, click “Text Update” button
- Tooltip
  - Hover over any of the text update fields to see a tooltip for the widget
  - PV info for widgets with connected PV
  - General info for other widgets
- Context Menu (right click on any widget)
  - Get info for a given widget (i) -> debugging help
  - Reload Display
  - Open Display in editor
  - Send email (if configured for your site)
  - Send to Logbook (if configured for your site)
  - Possibly other functions…



Formatting options:

Default:	-4.76 a.u.
Dont show units:	-4.76
Decimal w/ 2 digits:	-4.[illegible] a.u.
Exponential Notation:	-4.76E0 a.u.
Engineering Nota[illegible]:	-4.76E0 a.u.

Com[illegible]are Number Options:

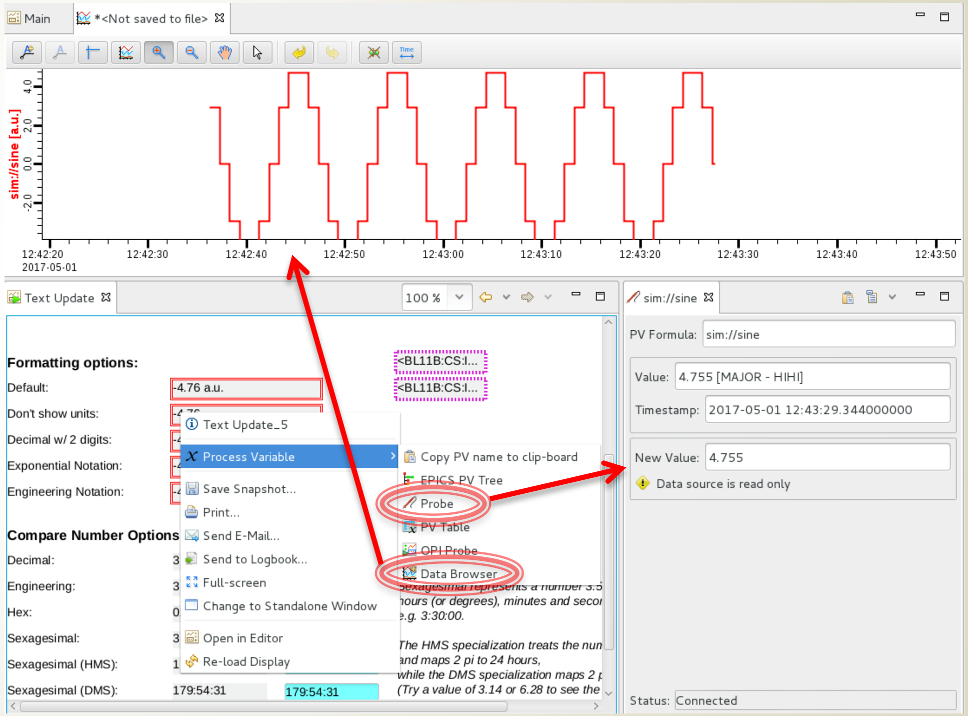
[illegible]ecimal:	3.1400	
Engineering:	3.1400E0	
Hex:	0x3	
[illegible]	3:08:24	
Sexagesimal (HMS):	11:59:38	11:59:38
Sexagesimal (DMS):	179:54:31	179:54:31

[illegible]represents a number 3.5 as [illegible]ees), minutes and seconds,

[illegible]cialization treats the number as radians and maps 2 pi to 24 hours, while the DMS specialization maps 2 pi to 360 degrees. (Try a value of 3.14 or 6.28 to see the effect)

Runtime: Send PV to other CSS Tools

- Display Builder is integrated with other CS-Studio tools…
- Probe tool
  - Right-click, select Process Variable -> Probe
- Databrowser
  - Right click, select Process Variable -> Data Browser



Formatting options:

Default:	-4.76 a.u.
Don't show units:	-4.76
Decimal w/ 2 digits:	-4
Exponential Notation:	-4
Engineering Notation:	-4

<BL11B:CS:I...

<BL11B:CS:I...

Compare Number Options

Decimal:	3	
Engineering:	3	
Hex:	0	
Sexagesimal:	3	
Sexagesimal (HMS):	1	
Sexagesimal (DMS):	179:54:31	179:54:31

sexagesimal represents a number 3.5 hours (or degrees), minutes and secon e.g. 3:30:00.

The HMS specialization treats the num and maps 2 pi to 24 hours, while the DMS specialization maps 2 p (Try a value of 3.14 or 6.28 to see the

PV Formula: sim://sine

Value: 4.755 [MAJOR - HIHI]

Timestamp: 2017-05-01 12:43:29.344000000

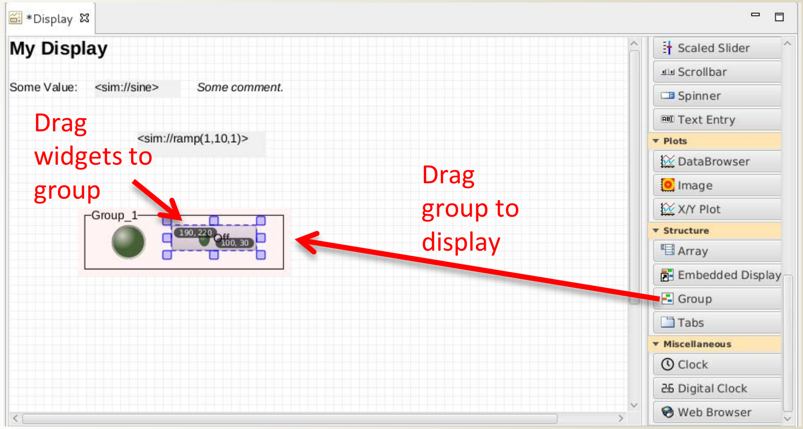
New Value: 4.755

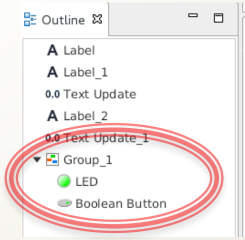
Data source is read only

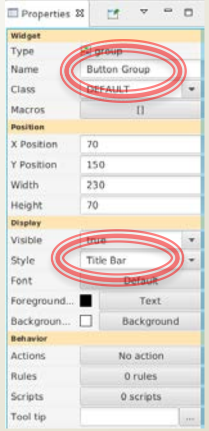
Status: Connected

Exercise: Grouping Container

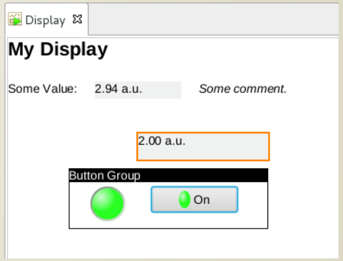
- Using the my_display.bob that we started earlier, group the LED and boolean button so that they can be manipulated as a unit.
  - Under “Structure” in the widget palette, drag a “Group” widget onto the display
  - Drag the LED into the group
  - Drag the boolean button into the group
  - The group will highlight when widgets are added, and the outline will reflect the new structure
  - Select the group and use the properties change the name to “Button Group” and the style to “Title Bar”
  - Right click and “Execute Display” to see the group in the runtime and click the button






Properties

Widget		
Type	group	
Name	Button Group	
Class	DEFAULT	
Macros	[]	
Position		
X Position	70	
Y Position	150	
Width	230	
Height	70	
Display		
Visible	true	
Style	Title Bar	
Font	Default	
Foreground...	■	Text
Backgroun...	□	Background
Behavior		
Actions	No action	
Rules	0 rules	
Scripts	0 scripts	
Tool tip		...



Exercise: Macros and Containers

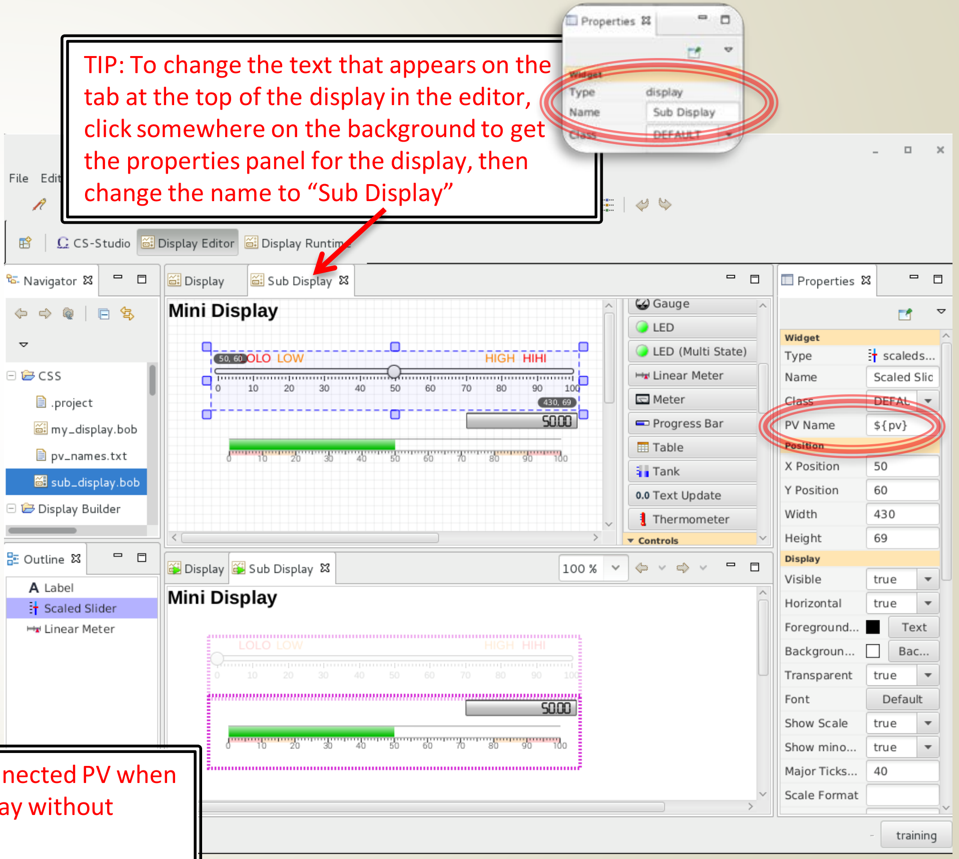
- Display Builder supports macros
  - Notate as $(macro) or ${macro}
  - Can be used for any widget property
  - Often used as PV name or partial PV name
    - $(pv)
    - $(pvprefix)_setpoint
  - Can also be used for numeric or boolean properties as long as value is number resp. “true”/”false”
- Utility of this comes from building a display using macros, then making several instances of this display with different macro definitions
- Macros can be defined at various levels, and are resolved in the following precedence:
  - Preferences
  - OpenDisplayAction
  - EmbeddedWidget
  - DisplayModel
  - GroupWidget
- For the next exercise:
  - Make a new display (“sub_display.bob”) using macros for widget properties
  - Add two embedded displays to “my_display.bob” with different macro definitions for sub display widgets

Exercise: Macros and Containers

- Create a new display file
  - File -> New, then Display Builder -> New Display
  - Name it “sub_display.bob” in folder /CSS
  - Delete all widgets except for the title label, change it from “My Display” to “Mini Display”
    - Delete widgets using backspace (ctrl-delete on OSX)
    - Set label text in the “Text” field of properties
- Add a scaled slider from the controls section of the palette
  - set “PV Name”=$(pv)
  - Under behaviors, set:
    - “Limits from…”=false
- Add a linear meter from the monitors section of the palette
  - Set “PV Name”=$(pv)
  - Under behaviors, set:
    - “Limits from…”=false
    - “Unit from PV”=false
- Right click and “Execute Display” to save the file and see the sub display in the runtime

TIP: To change the text that appears on the tab at the top of the display in the editor, click somewhere on the background to get the properties panel for the display, then change the name to “Sub Display”

Widgets will show disconnected PV when we execute the sub display without macros defined

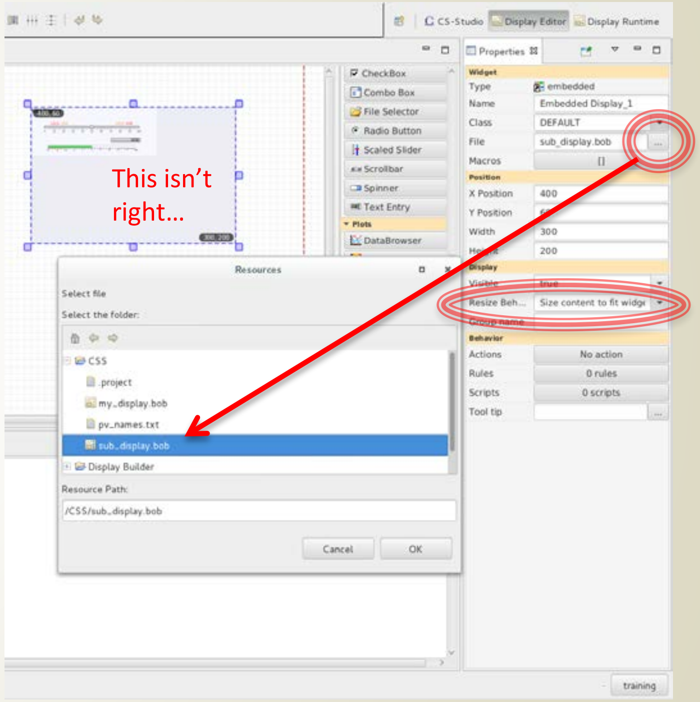


Widget	
Type	display
Name	Sub Display
Class	DEFAULT

Widget	
Type	scaleds...
Name	Scaled Slic
Class	DEFAU
PV Name	$(pv)
Position	
X Position	50
Y Position	60
Width	430
Height	69
Display	
Visible	true
Horizontal	true
Foreground...	Text
Backgroun...	Bac...
Transparent	true
Font	Default
Show Scale	true
Show mino...	true
Major Ticks...	40
Scale Format	

Exercise: Macros and Containers

- Open “my_display.bob” in the editor
- Add the Sub Display:
  - Drag an “Embedded Display” from the palette
  - With the embedded display widget selected:
    - Set the “Resize Behavior” to “Size content to fit widget”
    - Click the “…” button next to the File property
    - Navigate to /CSS/sub_display.bob in the file selection dialog
    - Click OK
- Oh no! The sub display is very small inside the embedded display widget
  - Each display has a width and height in pixels
  - We will need to set the sub display size boundaries more tightly around the widgets

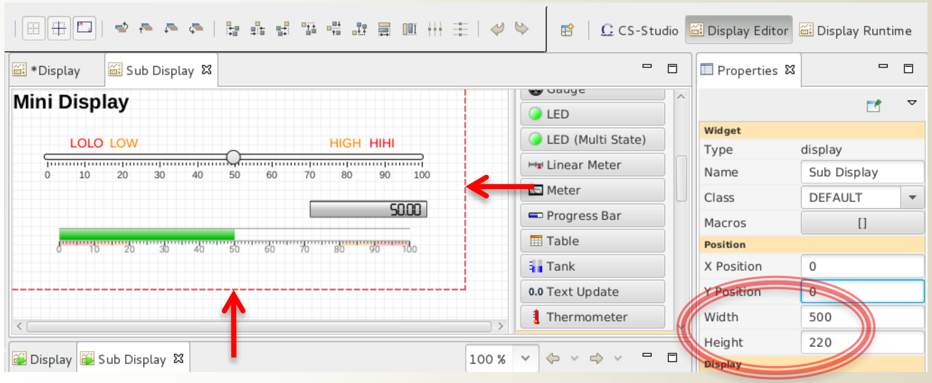


/CSS/sub_display.bob

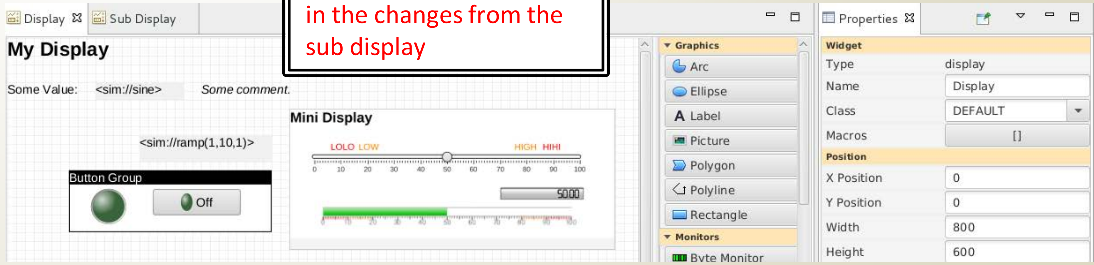
Widget	
Type	embedded
Name	Embedded Display_1
Class	DEFAULT
File	sub_display.bob
Macros	[]
Position	
X Position	400
Y Position	60
Width	300
Height	200
Display	
Visible	true
Resize Beh…	Size content to fit widge
Group name	
Behavior	
Actions	No action
Rules	0 rules
Scripts	0 scripts
Tool tip	

Exercise: Macros and Containers

- Open "sub_display.bob" in the editor
- Click on the background to get the display properties panel
- Change width and height until the red outline is tight around the widgets
- Return to "my_display.bob"
  - Right click -> "Reload Display"



Widget	
Type	display
Name	Sub Display
Class	DEFAULT
Macros	[]
Position	
X Position	0
Y Position	0
Width	500
Height	220
Display	

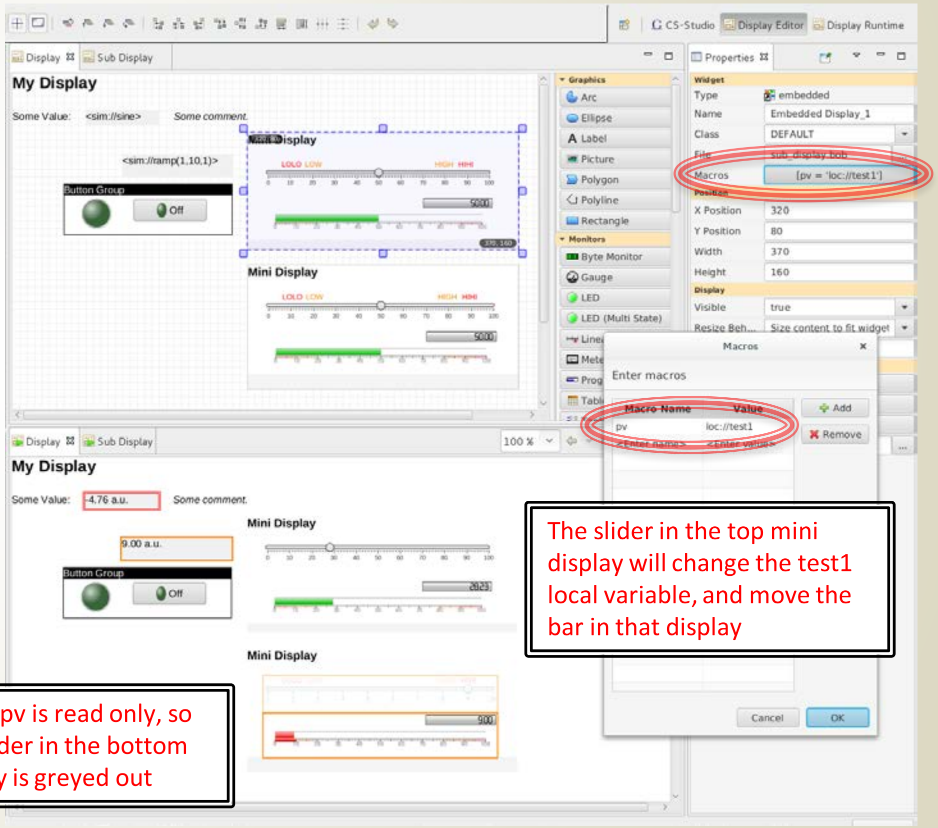


Widget	
Type	display
Name	Display
Class	DEFAULT
Macros	[]
Position	
X Position	0
Y Position	0
Width	800
Height	600

Reloading display will pull in the changes from the sub display

Exercise: Macros and Containers

- Add a second embedded display widget below the first
  - Set the file to “sub_display.bob”
  - Set resize behavior to “Size content to fit widget”
- Select the top embedded display:
  - Click the “Macros” property button
  - Enter a macro with name “pv” and value “loc://test1
- Select the bottom embedded display
  - Click “Macros”
  - Enter a macro with name “pv” and value “sim://ramp(1,10,1)”
- Right click and “Execute Display”



Properties

Widget	
Type	embedded
Name	Embedded Display_1
Class	DEFAULT
File	sub_display.bob
Macros	[pv = 'loc://test1']
Position	
X Position	320
Y Position	80
Width	370
Height	160
Display	
Visible	true
Resize Beh...	Size content to fit widget

Macros

Enter macros

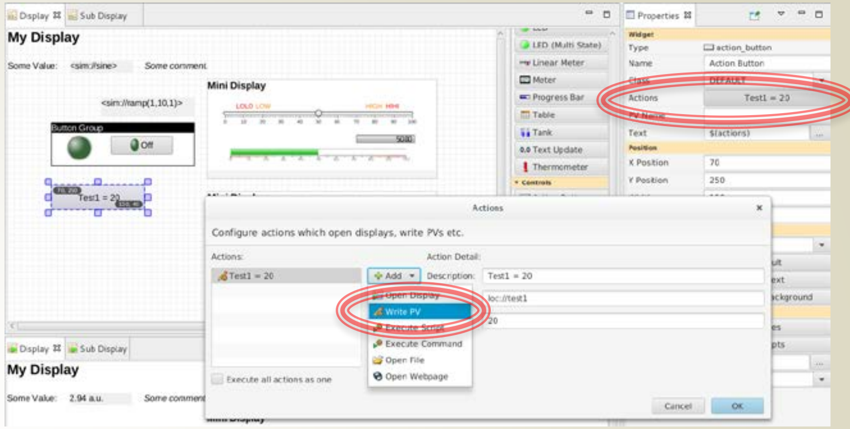
Macro Name	Value
pv	loc://test1
<Enter name>	<Enter value>

The slider in the top mini display will change the test1 local variable, and move the bar in that display

Ramp pv is read only, so the slider in the bottom display is greyed out

Exercise: Action and Menu Buttons

- Add an action button to change the value of the test1 variable
  - Drag an action button from the controls widget palette to the display
  - Click the Actions property button to open Actions dialog
  - Use the “Add” dropdown to select “Write PV”
  - Enter:
    - Description: Test1 = 20
    - PV Name: loc://test1
    - Value: 20
- Menu buttons are action buttons with multiple actions
  - Drag another action button to the display
  - Click the Actions property button to open Actions dialog
  - Use “Add” to add two actions, as above, where one action sets test1 to 50, and one sets test1 to 75
  - Set the “Text” widget property to “Set Test1 Value”
- Right click and “Execute Display”
  - Use the buttons to set the value in the top mini display



Actions

Configure actions which open displays, write PVs etc.

Actions: Test1 = 20

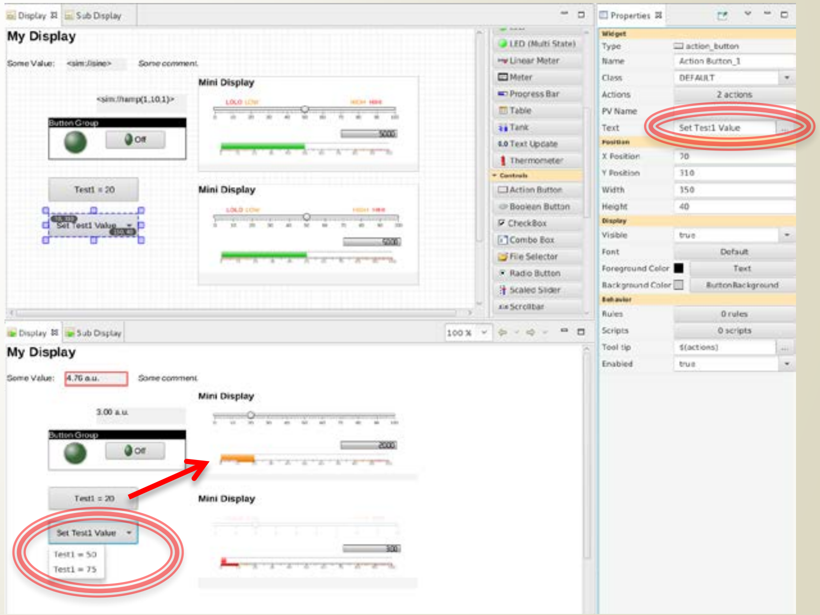
Action Detail: Description: Test1 = 20

loc://test1

20

Execute all actions as one

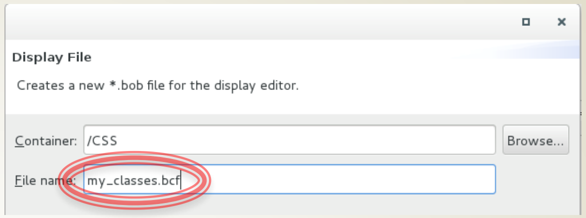
Type	action_button
Name	Action Button
Class	DEFAULT
Actions	Test1 = 20
Text	$(actions)
X Position	70
Y Position	250



Type	action_button
Name	Action Button_1
Class	DEFAULT
Actions	2 actions
PV Name	
Text	Set Test1 Value
X Position	70
Y Position	310
Width	150
Height	40
Visible	true
Font	Default
Foreground Color	Text
Background Color	ButtonBackground
Rules	0 rules
Scripts	0 scripts
Tool tip	$(actions)
Enabled	true

Exercise: Make a widget class file

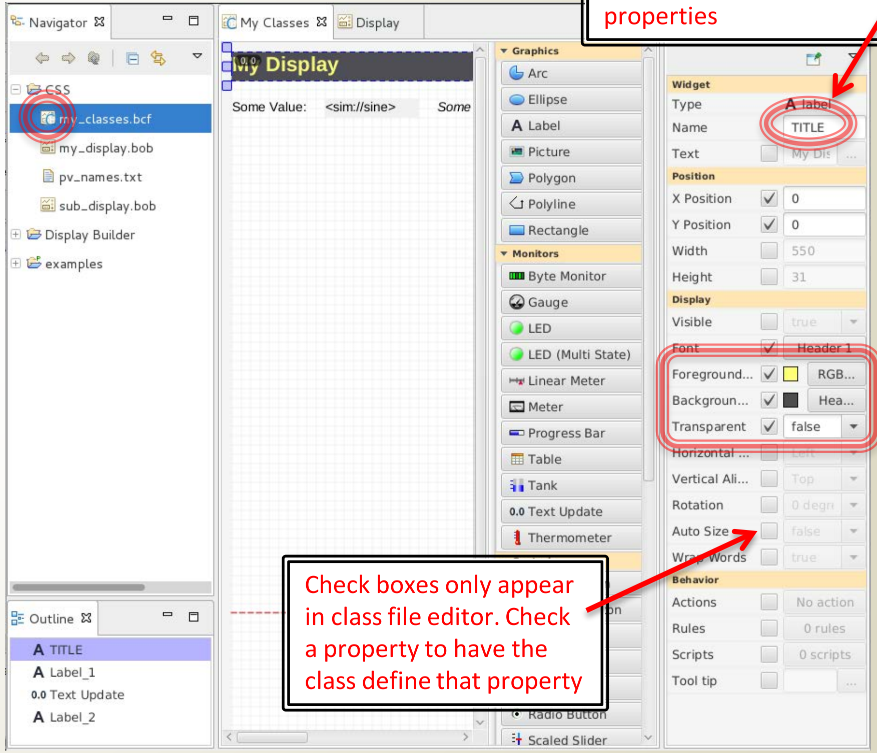
- Widget classes provide support for creating a standardized style for displays
- Design the widget properties once, in a class definition file, then use them many times
- Define a new class file:
  - Create a new display file, File -> New, then Display Editor -> New Display
  - For the file name, enter “my_classes.bcf”
  - The .bcf extension indicates a class file
  - The file icon in the navigator will reflect that this is a class file type
- Override the default TITLE class
  - Select the title “My Display”
  - The properties panel in the .bcf file editor works differently than in the .bob file editor
  - Checkboxes indicate which properties are set by the class
  - Set the widget name to “TITLE”
  - Change the “Foreground” property to a light yellow color
  - Change the “Transparent” property to false
  - Check the box next to the “Background” property, then set it to a dark grey color
- Save the file “my_classes.bcf”


Display File

Creates a new *.bob file for the display editor.

Container: /CSS Browse...

File name: my_classes.bcf



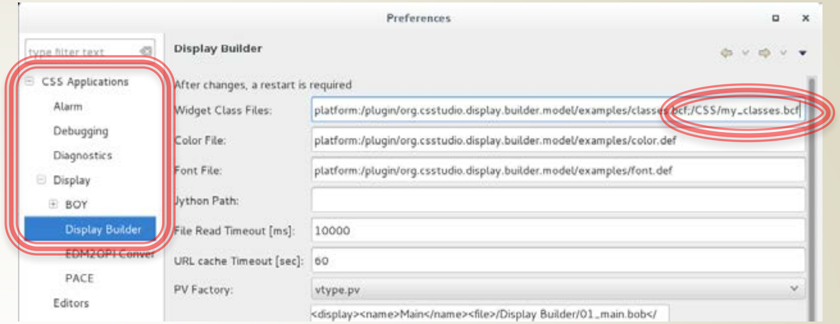
Widget		
Type		Label
Name		TITLE
Text	☐	My Dis
Position		
X Position	✓	0
Y Position	✓	0
Width	☐	550
Height	☐	31
Display		
Visible	☐	true
Font	✓	Header 1
Foreground...	✓	RGB...
Backgroun...	✓	Hea...
Transparent	✓	false
Horizontal ...	☐	Left
Vertical Ali...	☐	Top
Rotation	☐	0 degr
Auto Size	☐	false
Wrap Words	☐	true
Behavior		
Actions	☐	No action
Rules	☐	0 rules
Scripts	☐	0 scripts
Tool tip	☐	

Widget name in the class editor is the class name for this set of widget properties

Check boxes only appear in class file editor. Check a property to have the class define that property

Exercise: Make a widget class file

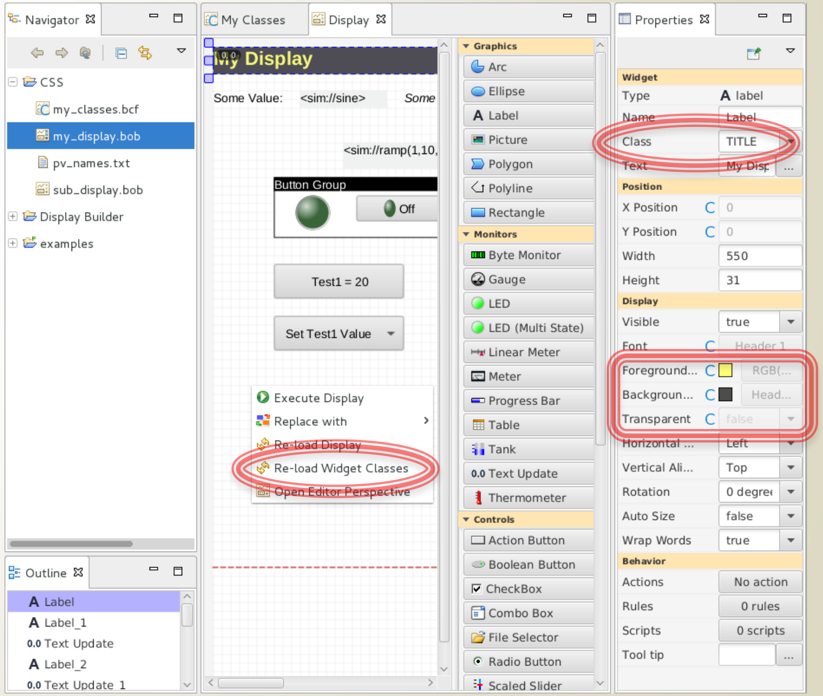
- Tell CS-Studio to use your class file
  - Open preferences: Edit -> Preferences
  - Navigate to CSS Applications -> Display -> Display Builder
  - Find the line “Widget Class Files”
  - Add your class file to the end of the list
    - Separate files with semi-colons
    - Use the path /CSS/my_classes.bcf
- Update your display to see changes
  - Open “mydisplay.bob” in the editor
  - Select the top label that says “My Display”, make sure the class is set to “TITLE”
  - Right click and select “Re-load Widget Classes”
  - The properties of the widget change to match the class style



Display Builder

After changes, a restart is required

Widget Class Files:	platform:/plugin/org.csstudio.display.builder.model/examples/classes.bcf;/CSS/my_classes.bcf
Color File:	platform:/plugin/org.csstudio.display.builder.model/examples/color.def
Font File:	platform:/plugin/org.csstudio.display.builder.model/examples/font.def
Jython Path:	
File Read Timeout [ms]:	10000
URL cache Timeout [sec]:	60
PV Factory:	vtype.pv
	<display><name>Main</name><file>/Display Builder/01_main.bob</



Widget	
Type	label
Name	Label
Class	TITLE
Text	My Disp
Position	
X Position	0
Y Position	0
Width	550
Height	31
Display	
Visible	true
Font	Header 1
Foreground...	RGB(...
Backgroun...	Head...
Transparent	false
Horizontal ...	Left
Vertical Ali...	Top
Rotation	0 degree
Auto Size	false
Wrap Words	true
Behavior	
Actions	No action
Rules	0 rules
Scripts	0 scripts
Tool tip	

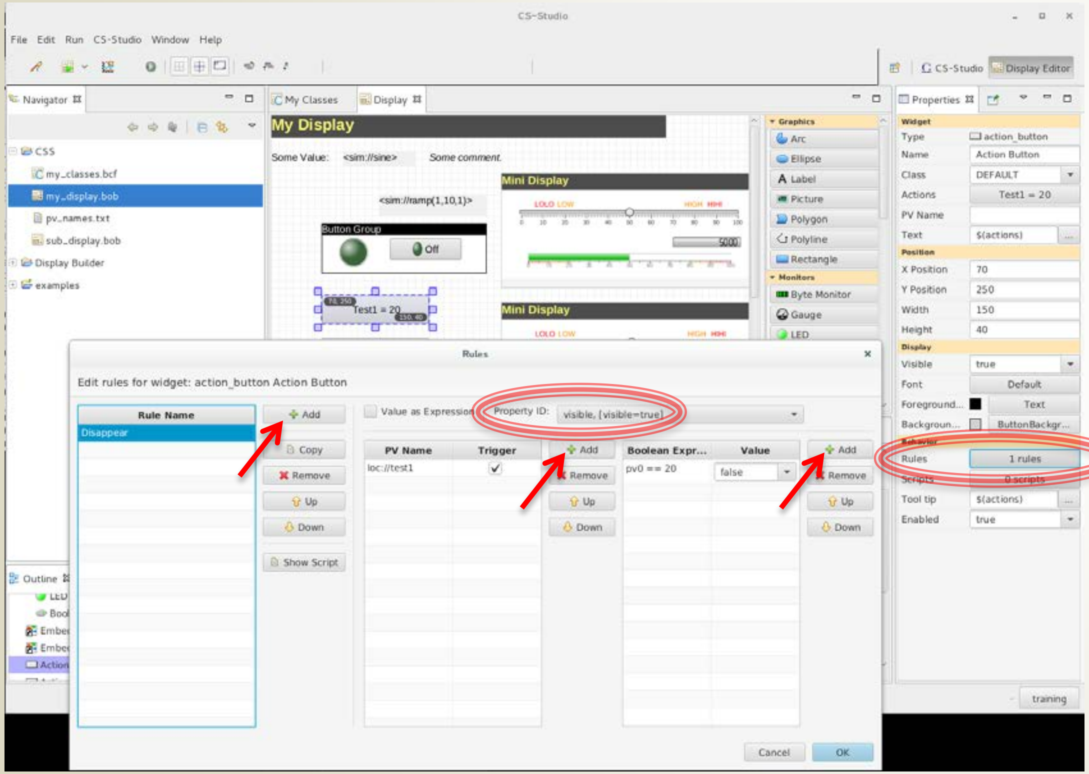
Exercise: Widget Classes

- Class files
  - Can override classes or define entirely new ones
    - One could design a “BLUE LED” in a class file
    - Once the .bcf class file is loaded, “BLUE LED” will appear in the class dropdown for LEDs in the display editor for .bob files
  - Are resolved in the order they are listed
    - You could have a “master.bcf”, then override certain settings in another file, “control_room1.bcf”
- Classes
  - Are applied to widgets when CS-Studio starts up
  - Are applied to widgets when the display is reloaded
  - Are applied to widgets when the widget classes are reloaded

Rules, Scripts

- Ideally, just add Widgets to the display, configure size and PV Name.
  Such 'simple' displays will be easy to maintain, can later be translated to the next display tool
- Scripts, triggered by PV updates, can
  - Modify any property of any PV
  - Add widgets
  - Call any of the internal CS-Studio code
  - Will be harder to understand and maintain

Exercise: Rules



Rules

Edit rules for widget: action_button Action Button

Value as Expression Property ID: visible, (visible=true)

Rule Name
Disappear

Add · Copy · Remove · Up · Down · Show Script

PV Name	Trigger
loc://test1	✓

Boolean Expr...	Value
pv0 == 20	false

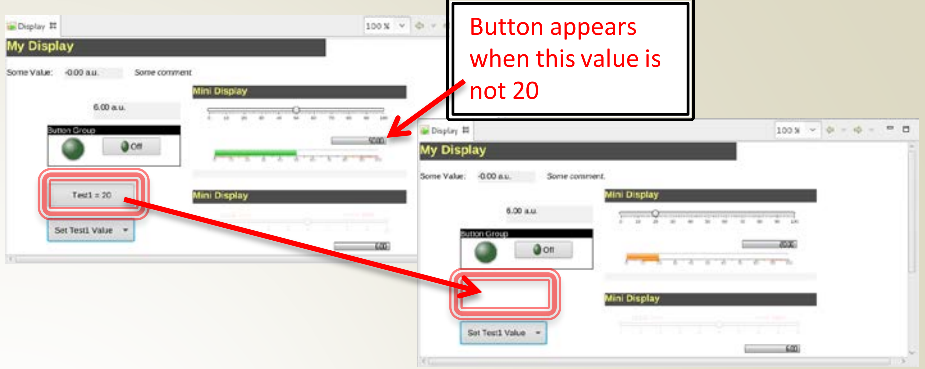
Cancel · OK

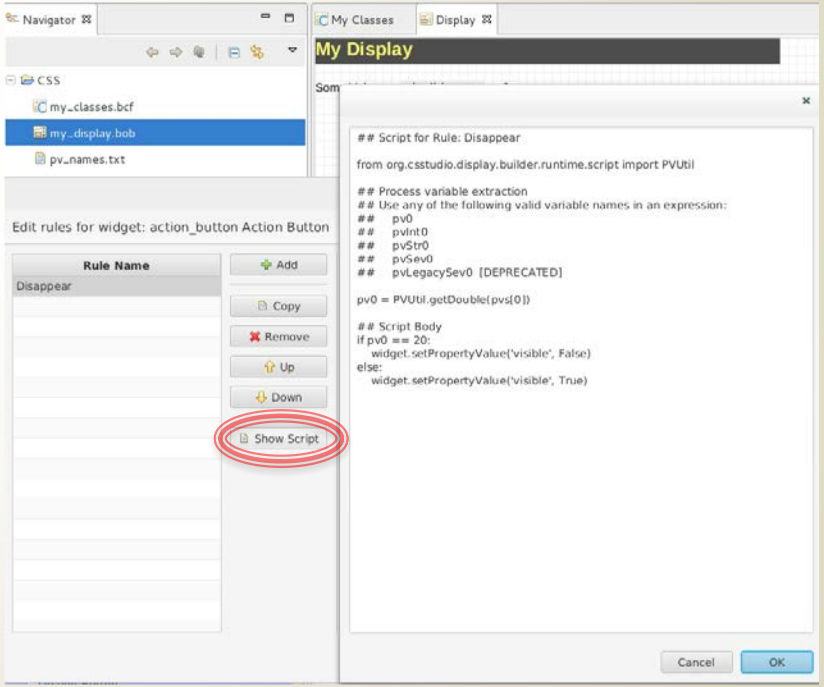
Property	Value
Widget	
Type	action_button
Name	Action Button
Class	DEFAULT
Actions	Test1 = 20
PV Name	
Text	$(actions)
Position	
X Position	70
Y Position	250
Width	150
Height	40
Display	
Visible	true
Font	Default
Foreground...	Text
Backgroun...	ButtonBackgr...
Behavior	
Rules	1 rules
Scripts	0 scripts
Tool tip	$(actions)
Enabled	true

- Rules are a guided way to safely generate scripts tied to widgets
- Scripts generally use some type of internal logic to determine a widget property
- Exercise: Make a button invisible when it is not needed
  - Open the “my_display.bob”
  - Select the action button “Test1=20”
  - Click the “Rules” property button to open the Rules definition dialog
  - Click “Add” next to “Rule Name”
    - Set rule name to “Disappear”
  - Use the Property ID dropdown
    - Select “visible”
  - Click “Add” next to “PV Name”
    - Enter “loc://test1”
    - Leave the trigger checkbox checked
  - Click “Add” next to “Boolean Expression”
    - Enter “pv0 == 20”
    - Set value to “false”
  - Click OK

Exercise: Rules

- To see the rule in action, right click on the display and “Execute Display”
  - Push the “Test1=20” button
  - The button will disappear until you change the value to something other than 20 using the other button or the slider
- What was going on with the pv0 in the boolean expression? What does this rule mean?
  - In the Rules dialog, select “Show Script” to see the python script that runs for this button
  - Each PV that we enter can be accessed using its index, starting at 0
  - If we use the “pv0” accessor, that will return a numeric (float) value for the 0th pv in our pv list for this rule -> loc://test1
  - We could alternatively use “pvStr0” to access this pv’s string value, or “pvSev0” to get its severity status
  - Since we selected the “visible” property from the dropdown, the rule generates the script to change this value
  - Try adding and changing rules and pvs to look at different script outcomes





Edit rules for widget: action_button Action Button

Disappear

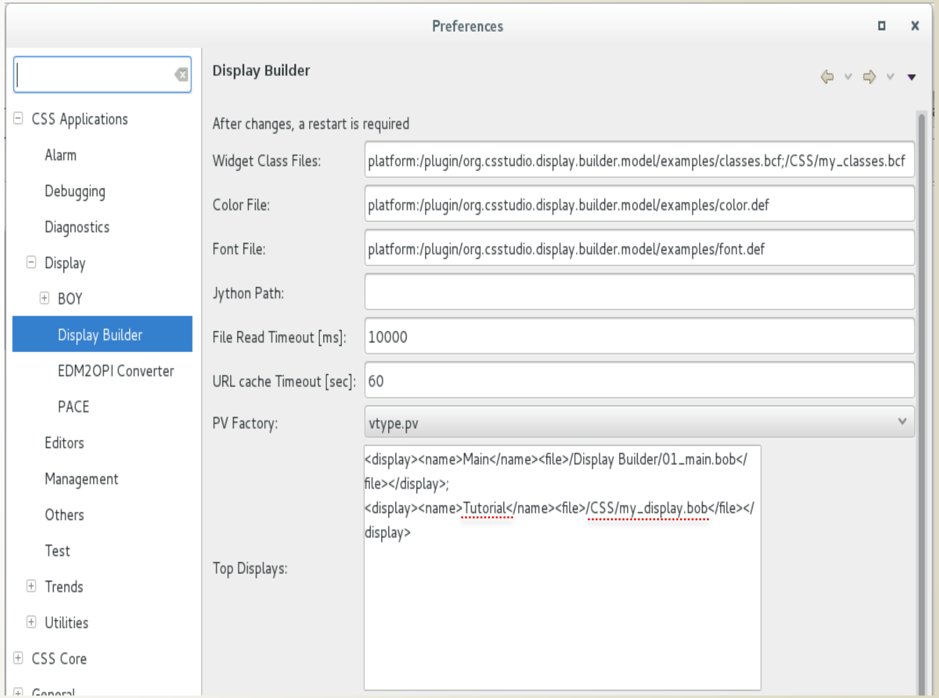
```
## Script for Rule: Disappear

from org.csstudio.display.builder.runtime.script import PVUtil

## Process variable extraction
## Use any of the following valid variable names in an expression:
##     pv0
##     pvInt0
##     pvStr0
##     pvSev0
##     pvLegacySev0  [DEPRECATED]

pv0 = PVUtil.getDouble(pvs[0])

## Script Body
if pv0 == 20:
    widget.setPropertyValue('visible', False)
else:
    widget.setPropertyValue('visible', True)
```

Scripting…

- As seen in rules, widgets can have scripts tied to them
- Scripting behavior is more advanced topic
- See Help and examples to get started scripting
- Use Scripts widget property to attach Python or JavaScript

Runtime Perspective Configuration

- As mentioned earlier, there is a "Display Runtime" perspective that is meant to be used in production
- Configure the "Top Displays" available in runtime perspective
  - Select Edit -> Preferences
  - Navigate to CSS Applications -> Display -> Display Builder
  - Locate the "Top Displays" area
  - Copy/paste the XML for the existing top display
  - Change the "name" to Tutorial
  - Change the "file" to /CSS/my_display.bob



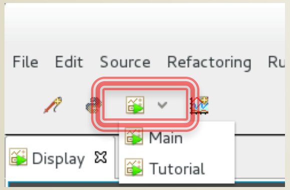
Display Builder

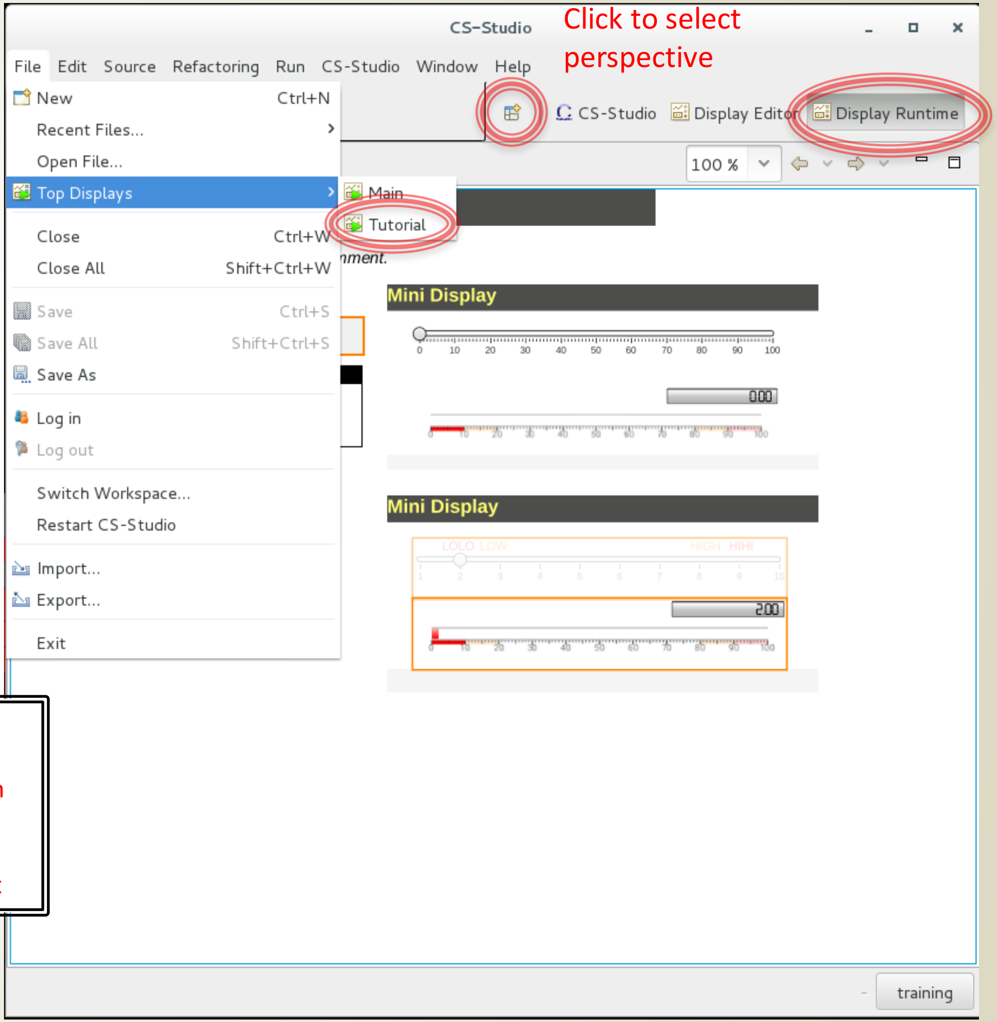
After changes, a restart is required

Widget Class Files:	platform:/plugin/org.csstudio.display.builder.model/examples/classes.bcf;/CSS/my_classes.bcf
Color File:	platform:/plugin/org.csstudio.display.builder.model/examples/color.def
Font File:	platform:/plugin/org.csstudio.display.builder.model/examples/font.def
Jython Path:	
File Read Timeout [ms]:	10000
URL cache Timeout [sec]:	60
PV Factory:	vtype.pv
Top Displays:	<display><name>Main</name><file>/Display Builder/01_main.bob</file></display>;<display><name>Tutorial</name><file>/CSS/my_display.bob</file></display>

Runtime Perspective

- Open the “Display Runtime” perspective in CS-Studio
- Open your display by selecting File -> Top Displays -> Tutorial
- Top Displays can also be accessed with the toolbar





Navigator, editor toolbar items, outline, and properties do not appear in Runtime Perspective.
This arrangement is meant for production deployment

Backward Compatibility: OPI Files

- Display Editor can read *.opi files
  - Right click on .opi and select Open With -> Display Editor
  - Most widgets will work, screens need to be checked
  - Scripts will need to be updated
- When the file is saved from display editor, it will save in new *.bob format
- Can maintain both files
  - Will execute the *.bob file if both *.bob and *.opi exist
  - Scripts can check if they are running in BOY or Display Builder and take different actions depending on runtime environment
    - See examples -> script_util -> portable.py

For more information…

- CS-Studio is an open source collaborative project
- http://controlsystemstudio.org/