



Yamcs Studio User Guide

October, 6th 2015

www.yamcs.org



Yamcs Studio User Guide

YAMCSSTCORE-SA-MA-001

This version was published October, 6th 2015

A later version of this document may be available at www.yamcs.org

© Copyright 2015 – Space Applications Services, NV

Table of Contents

1. Introduction	2
Overview	3
Installation	4
First Steps	5
Understanding the User Interface	7
Connecting to Yamcs	10
2. Running Displays	13
OPI Runtime Perspective	14
Processed Variables	16
Widgets	18
Inspector Windows	20
3. Editing Displays	23
OPI Editor Perspective	24
Resource Management	27
Widget Properties	28
Palette	29
Rules & Scripts	48
Theming	49
4. Views	50
Archive	51
Event Log	55
Alarms	56
Command Stack	57
Command History	61
Command Queues	62
Yamcs Clients	63
Data Links	64
5. Troubleshooting	65
Capturing Log Output	66

Chapter 1. Introduction

1.1. Overview

Brief History of Yamcs

Yamcs started as a server software first and foremost. While it initially started as a swiss-army knife to fill gaps in existing traditional mission control systems, it gradually grew to cover the whole spectrum of TM processing and TC commanding. Missions can have very specific software requirements, and often include a varied stack of software. Over the years Yamcs was extended in various ways to play nice with different kinds of TM and TC software.

Along the way, standalone client GUIs were developed as the need arose. This includes Archive Browser, Event Viewer, Packet Viewer and Yamcs Monitor. These tools are being used in many missions.

For many years, however, the main thing that was felt missing from the Yamcs software stack, was a display solution. And this is where Yamcs Studio comes into play.

Yamcs Studio

Yamcs Studio is a desktop frontend to Yamcs. Its main attraction is its support for operator displays, but it also includes other facets that cover TC commanding and insight into various runtime aspects of Yamcs. Most of our legacy client GUIs have also been ported into Yamcs Studio (with the exception of the Packet Viewer), for an integrated solution.

Yamcs can be made to integrate with display software other than Yamcs Studio (and in fact, this is often the case in long-running missions where Yamcs was added in the mix after the project's initial conception), but there are advantages to working with Yamcs Studio:

- Increased semantical coherence
- Single point of contact
- Opportunities for customisation that covers both server and client
- Integrated views operational views

Technology

Yamcs Studio is an Eclipse RCP application, and builds upon Open Source software libraries like CS-Studio, Netty, Protobuf and of course our own Yamcs API.

The main programming language is Java 8.

License

Yamcs Studio follows a similar licensing scheme as Yamcs Server. The core of Yamcs Studio is open-source and licensed under the Eclipse Public License. Mission-specific extensions can be developed on a case-by-case basis and under custom licenses.

We believe that having an open-source core, is not only fun and exciting, but that this increases the quality of our products and benefits all of our customers equally.

1.2. Installation

Install Java 8

You will need Oracle Java 8 installed. We currently recommend the latest Oracle JDK 8.

Download Yamcs Studio

Download the latest Yamcs Studio release for your platform. Extract to your preferred location and just launch it. When it asks you to choose a workspace, choose a new directory inside for example your home directory. Workspaces contain displays, scripts and user preferences. By default your workspace will be populated with a few sample projects. These projects contain displays that show simulated parameters as produced by a default-configured Yamcs Server.

Troubleshooting

Most problems related to starting Yamcs Studio, have to do with Java not being correctly detected, or by trying to launch Yamcs Studio with an old version of Java. Both of these issues are usually resolved by installing Oracle JDK 8.

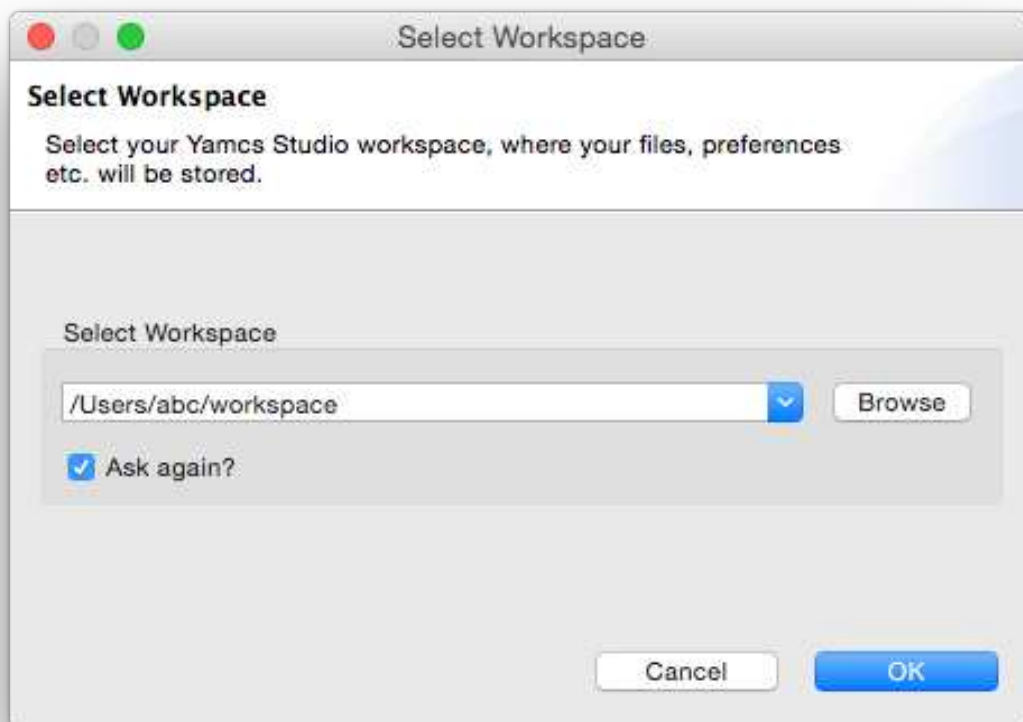
In case that didn't help, Try defining the `-vm` property in the root `yamcs-studio.ini` file. Refer to the instructions available at <https://wiki.eclipse.org/Eclipse.ini>.

1.3. First Steps

Launching Yamcs Studio

When you launch Yamcs Studio for the first time it will ask you to choose a workspace. A **workspace** is where you store your resources (e.g. a display file).

With Yamcs Studio, you are always working on one workspace at a time. Usually workspaces are fairly static, and you can often do with just one of them. If you untick the **Ask again?** option you will no longer see this message at startup.



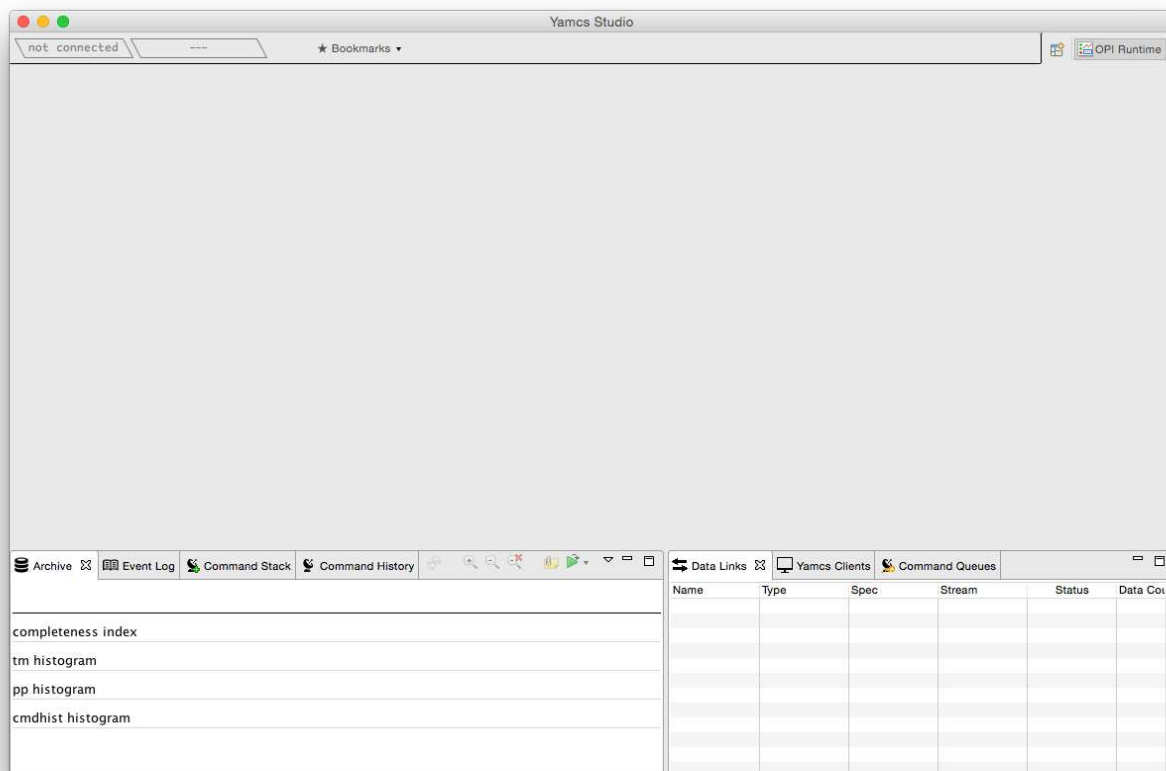
Choose your preferred location, and click **OK**.



If you unticked the **Ask again?** option, but you want to switch workspace at a later moment, open **File > Switch Workspace...** from the window menu to choose a different directory.

Empty Workspace

Yamcs Studio is now launched and you should see an empty workspace with the default window arrangement:






The empty area in the middle is where displays will open. In the middle of the screen we have a

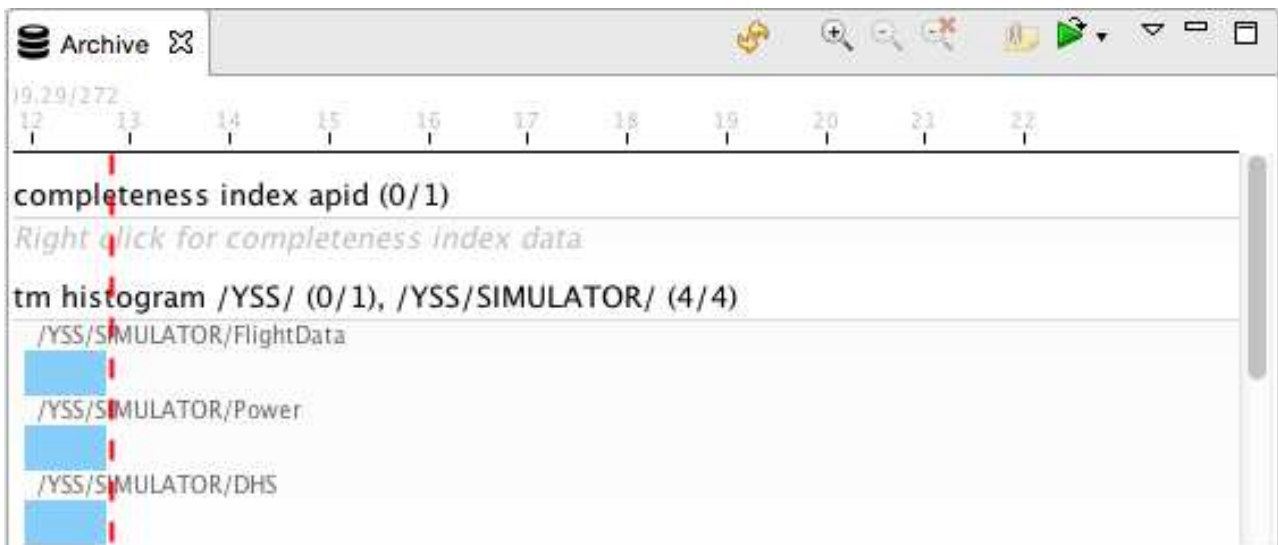
Yamcs Studio has two different modes (called *perspectives*). OPI Editor and OPI Runtime. When Yamcs Studio is launched for the first time the user will be welcomed with the default OPI Runtime perspective, which is used during realtime operations, or for testing out displays with live telemetry.

1.4. Understanding the User Interface

Yamcs Studio is composed out of multiple views that are arranged together in a perspective. The user has great flexibility in modifying the default arrangement to his liking.

Views

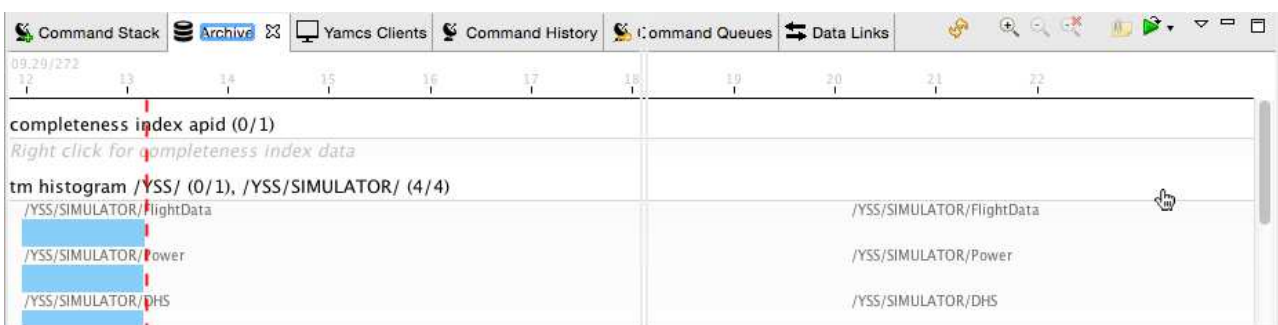
Views all share the same user interface organization. On the left you see a tab with the view icon, followed by a title, and then a close icon. On the outer right there are actions to  **Minimize** or  **Maximize** the view. Some views (such as the one in the screenshot) also have a third pull-down icon  with view-specific actions in it. Most views, though, add dedicated colored icons next to the standard icons. The pull-down menu is used to hide less-often used actions.



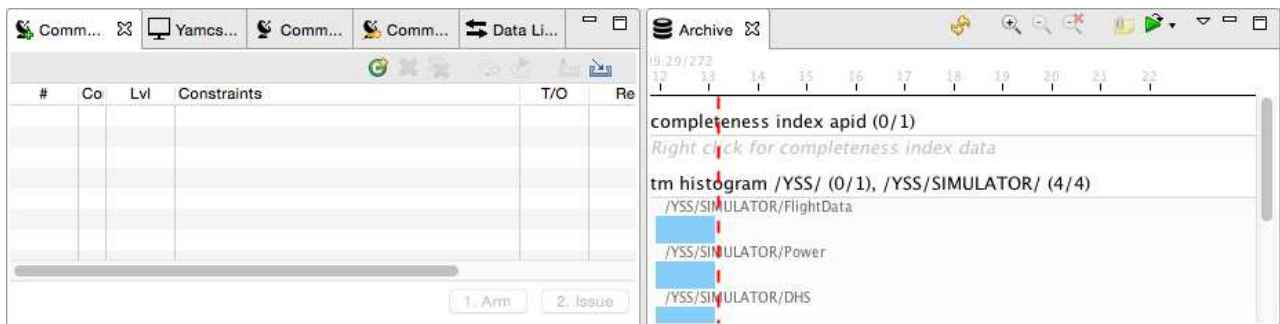
To reopen a view which you closed earlier, or to open another view choose **Window > Show View**

Views can be resized, moved and stacked. This allows you to customize your workspace to your own personal preference.

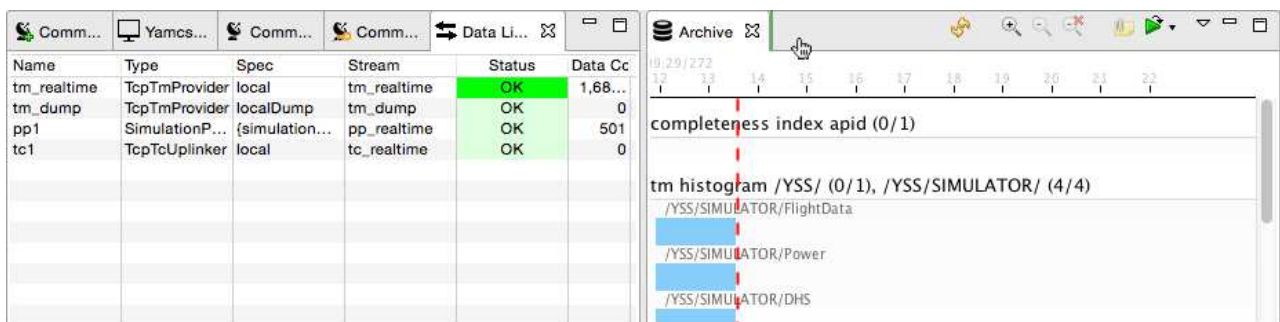
For example, let's say we want to put the Archive view in its own dedicated location. Click on the tab title, and while holding the mouse down, drag towards the right. If you move far enough, you will notice an outline suggesting the view's new position (this may look slightly different on your platform).



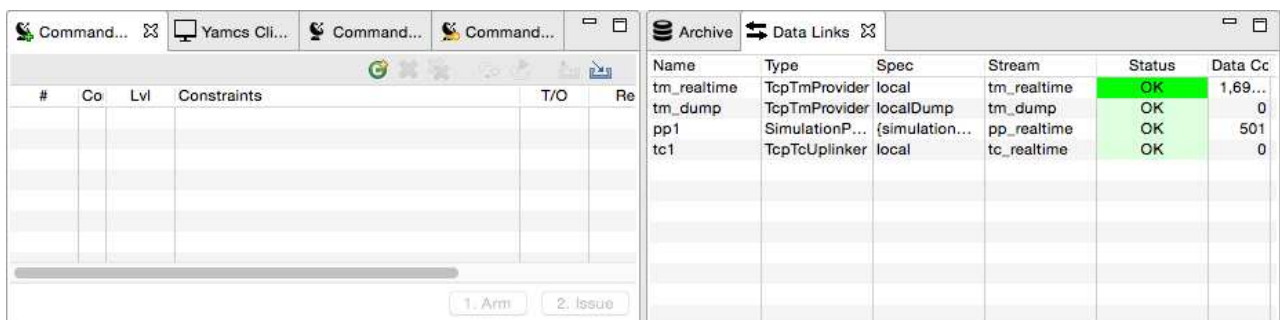
Release the mouse to confirm this view's new position.



Let's say we want to move the Data Links view to the right as well. Again, click its title and drag your mouse next to the Archive tab. You will see a green bar suggesting this tab's new placement.



Release the mouse to confirm this view's new position.



Feel free to experiment some more with the drag feature. As you try dragging to different locations, you will notice that Yamcs Studio has several hot spots where you can attach your views. For example, you can detach windows by dragging them outside of your application window. This provides additional screen space if your workstation supports multiple monitors.


When you close Yamcs Studio and reopen it, it will restore your preferred view and window arrangement.



Yamcs Studio stores the information about your view arrangement in a `.metadata` folder inside your workspace. This is how it knows how to restore this information through restarts. If you share your workspace with other users through a version control system, you should consider *not* committing this `.metadata` folder. This way everybody can have his own preferred arrangement without colliding with each other.

Perspectives

Perspectives contain an organization of views. As you were performing the above actions with views, you were working within a certain *perspective*.

In the top right bar, you can see the Perspective Switcher. This is where you choose your current perspective. By default Yamcs Studio puts you in the OPI Runtime perspective, but by clicking the plus icon  you can switch to the OPI Editor perspective, which has a different arrangement of views.

Again you can modify the views in this perspective to your heart's content, but as a general precaution we would advise that you distinguish between 'Running Displays' (OPI Runtime), and 'Editing Displays' (OPI Editor). In future versions of Yamcs Studio we may make this distinction more apparent, or even go as far as to offer two different products.

Notice, as you go back to the OPI Runtime perspective, that your earlier view arrangement is nicely restored.

If at any time you want to reset your perspective to the defaults, select **Window > Reset Perspective...**



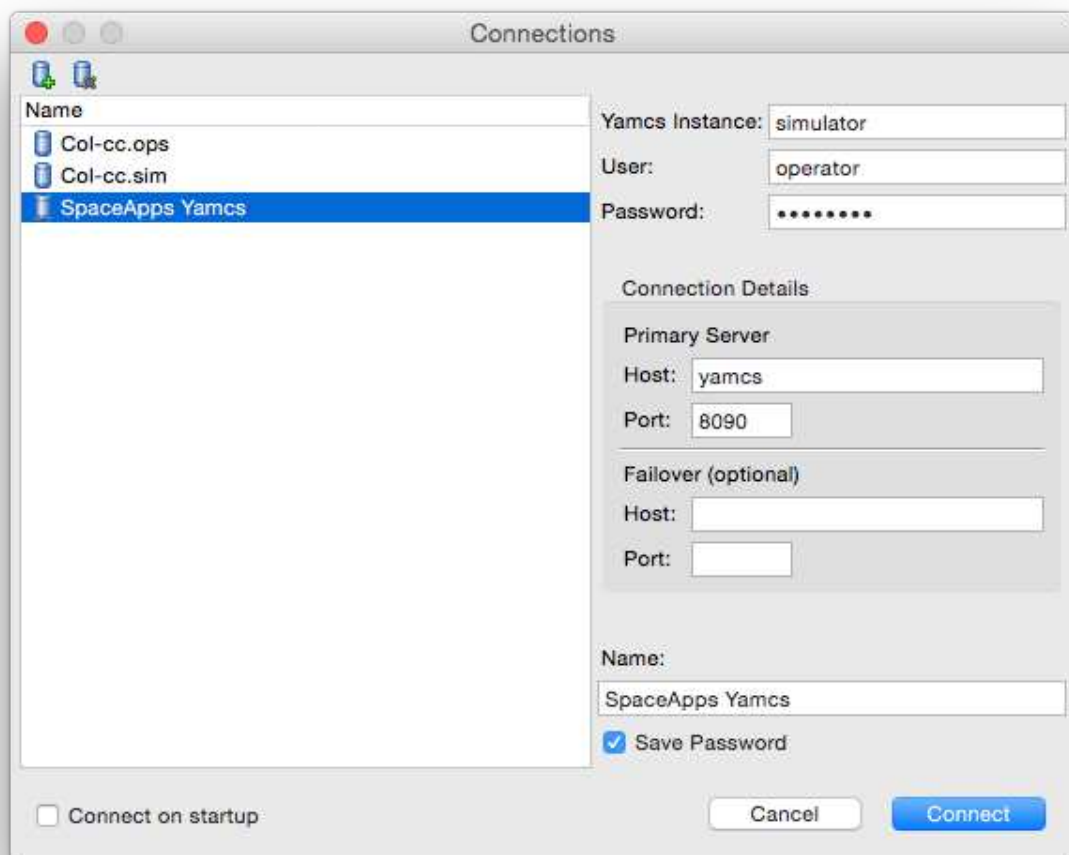
Some people prefer to have a separate window for every perspective. To do so, select **Window > New Window**. This action will duplicate your current window. You can then switch the new window to a different perspective, without impacting your original window.



1.5. Connecting to Yamcs

Yamcs Studio is a client application that's meant to be connected with Yamcs Server.

Yamcs Server, or 'Yamcs' handles the processing, archiving and dispatching of telemetry data. Yamcs Studio is one of the possible Yamcs clients for receiving telemetry data.

To configure a Yamcs connection, select **File > Connect...** This will open the Connections window where you can manage your connections. For many missions, one connection will do just fine, but depending on how Yamcs is deployed at your site, you may have multiple Yamcs instances on the same server, or even multiple Yamcs servers.



Click  **Add Server** to add a server connection, or  **Remove Server** to remove the selected server connection.

Connection Properties

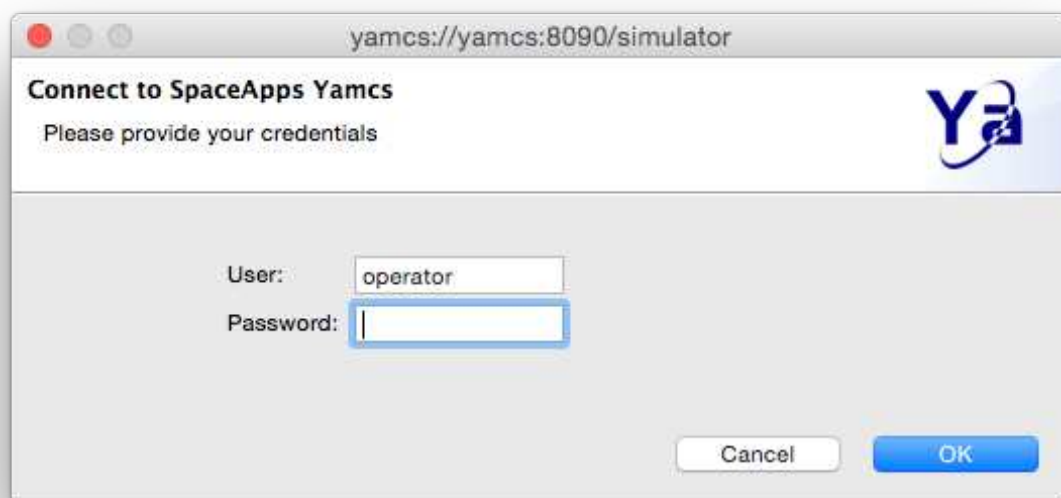
The right panel contains editable details for the selected server connection. We document the available properties below, but if you're unsure what to fill in, ask details to the person that is responsible for installing Yamcs at your site.

Yamcs Instance	Required	Yamcs can run multiple instances in parallel. You can think of instances like different environments, where every instance is completely separated from the other instance. While Yamcs Server may be running multiple instances in parallel, Yamcs Studio will always connects the user to one specific instance, which you have to configure here.
User / Password	Optional	If your Yamcs instance is secured, fill in your user and password here.
Primary Server	Required	Specify your actual host and port connection details here. The port is usually 8090.
Failover Server	Optional	<p>If you specify a second host/port configuration, then Yamcs Studio will automatically failover to this second server in case connection with the primary server could not be established, or was lost.</p> <p>On the server-end, this setup requires two distinct Yamcs servers that are being kept in sync.</p>
Name	Required	You can give your configuration a name of your choosing. This name will be used to represent this connection in the left panel of the Connections window.
Save Password	Optional	If you prefer not to enter your password at every occasion, tick this box to save your password to disk. Please be aware that your password will be saved in a manner that is difficult, but not impossible, for an intruder to break.

Connecting

All changes you make are automatically saved when you click **Connect**. If you want to discard your changes click **Cancel**.

Select the **Connect on startup** option, if you would like Yamcs Studio to automatically reconnect to the last used Yamcs instance during start-up. If this connection requires privileges and you chose not to save your password to disk, you will see a specialised login window everytime you start Yamcs Studio:





Connection preferences are stored in a hidden folder under your home directory, and will continue functioning whenever you upgrade your copy of Yams Studio.

You can verify that your copy of Yams Studio is properly connected by looking at the top left processor indicator of the OPI Runtime perspective:

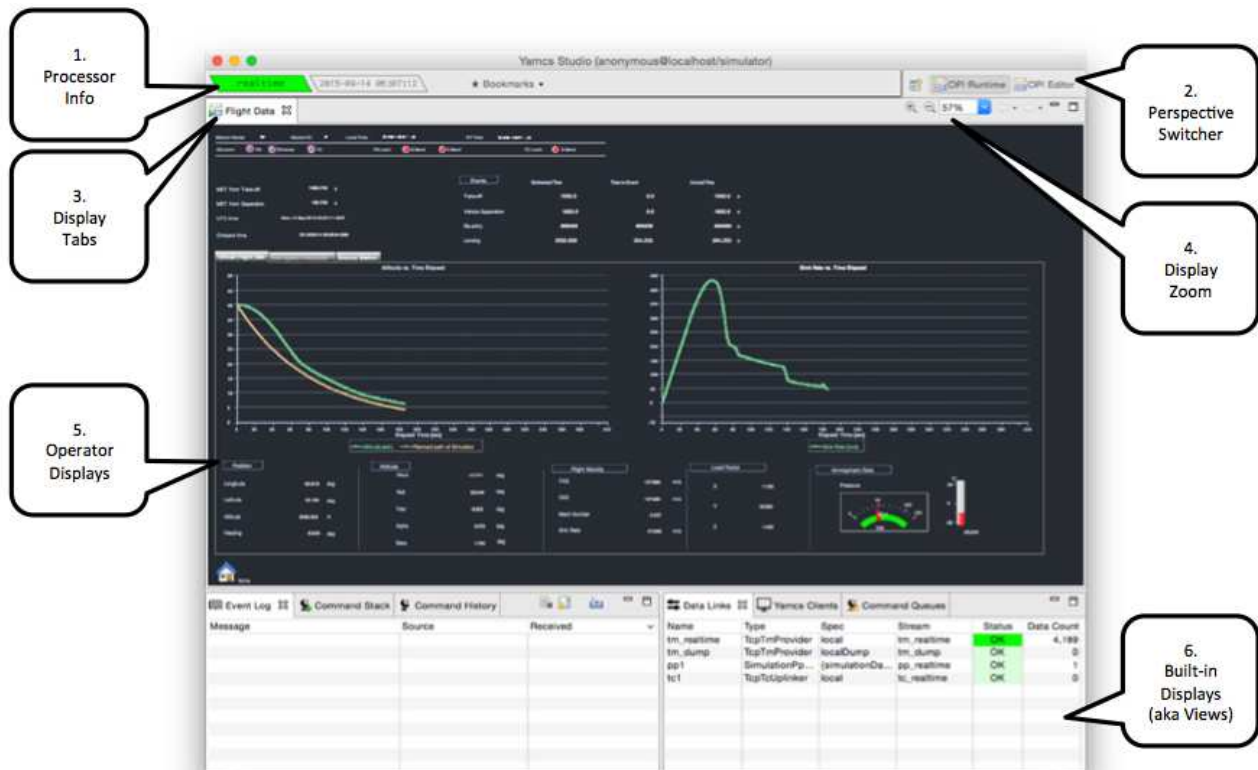


If it says `realtime`, then you've successfully connected.

Chapter 2. Running Displays

2.1. OPI Runtime Perspective

The OPI Runtime perspective is useful for realtime operations, or for testing out displays as they are being built. The default layout looks like this:



1. Processor Info

This zone holds two status indicators. The first indicator light shows the processor that Yamcs Studio is currently listening to. Yamcs supports many concurrent processors (realtime, replay channels). By default Yamcs Studio will always connect to `realtime`.

Next to that we see a second indicator which currently shows the processor time as provided by Yamcs. The simulator outputs generation times equal to the local system clock. If however we were to start a replay of archived data, we would notice this time adjusting to the location of our replay channel.

2. Perspective Switcher

When you launch Yamcs Studio it will open in OPI Runtime mode (OPI means Operator Interface). With the perspective switcher you can switch Yamcs Studio to the OPI Editor mode. Doing so will store and close your current arrangement of windows and views, and will open a different arrangement that is optimised for editing displays.

Note that it is possible to make builds of Yamcs Studio that include *only* the runtime perspective. This can significantly improve UX during operations.

3. Display Tabs

Displays open in different tabs. These are not normal tabs, though. By clicking and dragging these tabs we can easily create split screens, or different tab stacks. We can also drag a tab out of its parent window into a new window. In fact, Yamcs Studio is optimised for multi-monitor systems. Window layouts are

restored through restarts of Yamcs Studio.

4. Display Zoom

The above display was configured in such a way that it automatically stretches (while preserving aspect ratio) to fit the available screen space. This behaviour can be turned on or off by the display author. Regardless of its setting, as a display user we can always zoom in or out of the display using these controls.

5. Operator Displays

This area contains displays that were authored in the OPI Editor perspective. Displays contain any number of widgets. Most widgets can be connected to TM, which will also make them alarm-sensitive. In practice this means that they will be highlighted with different decorations depending on the alarm level. There are also things like button widgets which can for example open other displays, or launch a telecommand, or open dialog boxes, etc. All widgets are highly customisable using scripts and/or rules (rules are a user-friendly layer on top of scripts). We are in the process of documenting and expanding the library of functions that can be called from scripts.

6. Built-In Displays

Yamcs Studio comes with an array of built-in displays that offer more dynamic views on different aspects of Yamcs. These built-in displays (or Views, as Yamcs Studio calls them) cover concepts like commanding, event logging, alarm overviews (upcoming) and archive insight.

2.2. Processed Variables

Processed Variable or ‘PV’ is a term used by Yamcs Studio that covers the different types of data sources that a widget can be connected to. It is a more general term than parameter, which is a Yamcs Server notion. In Yamcs Studio there are other types of PVs than parameters.

PVs are uniquely identified by a *PV Name*. If multiple widgets have dependencies on the same PV, only one instance will be created and shared between these widgets.

The term PV is used to indicate both the name of a specific data source definition, as well as any instances of that definition. Context usually makes it apparent which of the two is meant.

A PV is considered *connected* if the data source is available, and at least one widget within Yamcs Studio is subscribing to it. As soon as no more widgets are connected to a PV, the PV gets *disconnected*.



This last property currently has as a side effect that for example chart widgets lose their history when closing and reopening the containing display. We are aware of this, and have ideas on how to address this shortcoming.

PV Info

PVs always have a *name* uniquely identifying the specific data source. Depending on the type of PV, this name is prefixed with a different schema.

Schema	PV Type
loc://	Local PV
sim://	Simulated PV
para://	Yamcs Parameter
sw://	Yamcs Software Parameter



While exploring existing displays you may sometimes notice widgets that are backed by a PV *without* a schema. By convention these are all resolved using the `para://` schema. Displays authors often choose not to specify this schema.

Local PVs are read and written entirely in a running Yamcs Studio instance. They are never communicated to Yamcs, nor to any other copies of Yamcs Studio. Local PVs are typically used by the display author as a means to store information that needs to be communicated from one widget to another. They also form a powerful building block when scripting advanced displays due to their ability to store runtime state. This makes it possible to script logic based on a historical window of values.

Local PVs are transient, and are reset when Yamcs Studio is restarted.

Example PV Names:

- todo

Parameter

Parameter PVs represent a readonly value that is provided by Yamcs. Typically this denotes telemetry.

The PV Name for parameters is the fully qualified XTCE name as specified in the Yamcs Mission Database.

Example PV Names:

- `/YSS/SIMULATOR/BatteryVoltage1`
Short for `para:///YSS/SIMULATOR/BatteryVoltage1`
- `/YSS/SIMULATOR/BatteryTemperature1`
Short for `para:///YSS/SIMULATOR/BatteryTemperature1`

In these examples YSS is the name of the root space system. SIMULATOR is the name of the space system directly below, which defines both measurements BatteryVoltage1 and BatteryTemperature1.

Software Parameter

Same concept as a Parameter, but has additional support for writing values from the client to the server. In this regard they can be used as a means of communicating information from one client to another using Yamcs Server as the medium.

Remark that software parameters are not currently archived by Yamcs Server, and will therefore be reset when Yamcs is restarted.

Simulated Value

Locally generated functions. Mainly useful during testing, or in combination with other PVs using formulas.

Formulas

PVs can be combined together using client-side formulas. Formulas always start with `=` followed by a formula expression. Expressions are similar to Excel formulas.

2.3. Widgets

Most widgets are backed by a PV. Some widgets (e.g. layouting widgets) are typically not connected to a PV, other widgets (e.g. charts) can be backed by more than one PV.

Color Decorations

When a widget is backed by a PV, it will be decorated according to its runtime state. The specific colors of these decorations can vary since the default colors can be overridden (or disabled) by the display author.

State	Decoration
Connected	No decorations
Connected, but no value (yet)	Dashed pink border around the widget
Disconnected	Solid pink border around the widget and the label 'Disconnected' in the top left corner (space-permitting)
Expired	Blinking solid pink border around the widget
Minor Alarm	Solid orange border around the widget
Major Alarm	Solid red border around the widget









Note that the color information for alarms is currently not as rich as it could be. Yamcs parameters support five different levels of alarms, as well as a range of special monitoring values. This information is for now transformed using the following mapping:

Yamcs Server	Yamcs Studio
WATCH WATCH_LOW WATCH_HIGH	MINOR
WARNING WARNING_LOW WARNING_HIGH	
DISTRESS DISTRESS_LOW DISTRESS_HIGH	
CRITICAL CRITICAL_LOW CRITICAL_HIGH	MAJOR
SEVERE SEVERE_LOW SEVERE_HIGH	












Catalogue

We limit ourselves to just listing the default widgets. Their runtime behaviour should be fairly straightforward. The various properties are detailed when we address the OPI Editor.

Graphics
















 Arc	 Rectangle	 Label
 Polyline	 Rounded Rectangle	 Image
 Polygon	 Ellipse	

Monitors









 LED	 Progress Bar	 XY Graph ¹
 Image Boolean Indicator	 Gauge	 Intensity Graph
 Text Update	 Thermometer	 Byte Monitor
 Meter	 Tank	

¹ Clear the view on this widget by right-clicking on it and selecting **Clear Graph**. If you want advanced controls, like zooming, activate the toolbar by right-clicking on your widget and selecting **Show/Hide Graph Toolbar**.

Controls

 Action Button	 Knob	 Image Boolean Button
 Menu Button	 Scrollbar	 Check Box
 Text Input	 Thumb Wheel	 Radio Box
 Spinner	 Boolean Switch	 Choice Button
 Scaled Slider	 Boolean Button	 Combo

Others

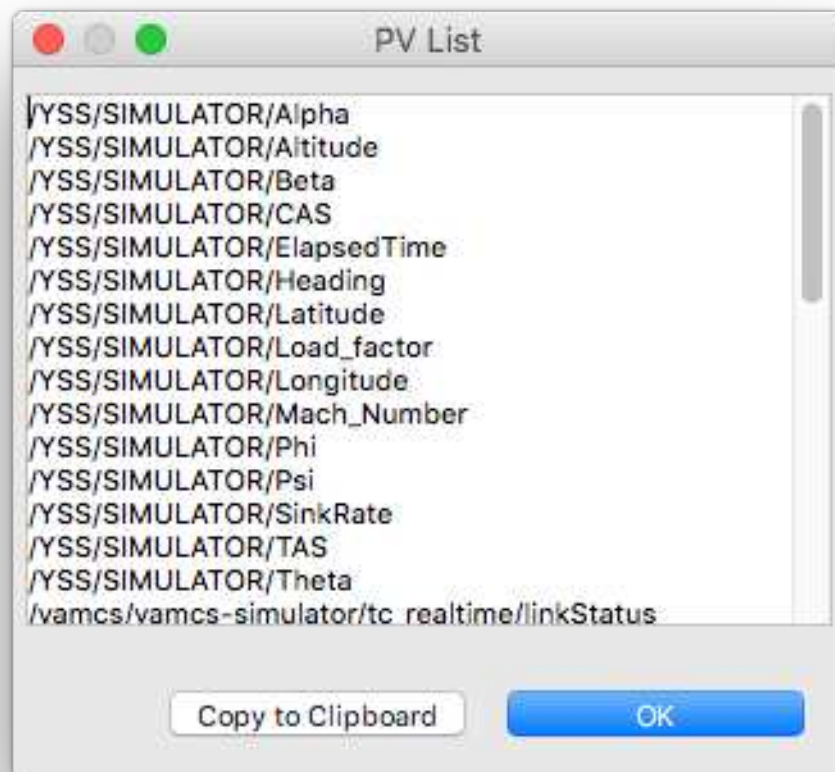
 Table	 Grouping Container	 Sash Container
 Web Browser	 Linking Container	 Grid Layout
 Array	 Tabbed Container	

2.4. Inspector Windows

There are a few standalone windows that can be opened for inspecting widgets.

PV List

If you right-clicking anywhere in a display, and choose **Dump PV List**, you will see a window listing the unique PVs that are defined inside any widget of that display. This is mostly useful for quick-fixing runtime issues.

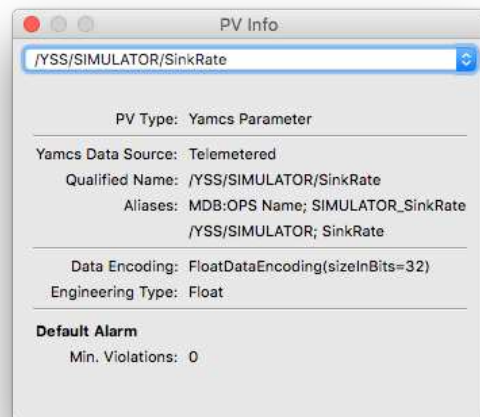
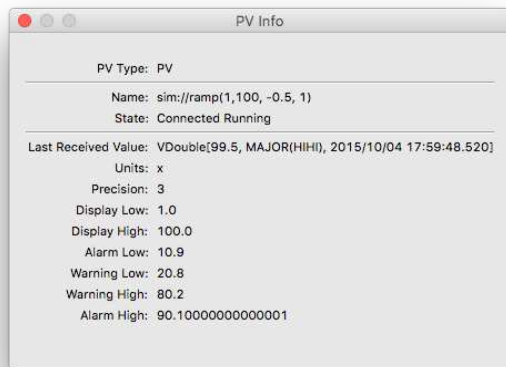


PV Info

Right-click on a widget backed by a PV, and select **PV Info**. This opens a window where you get extra information on the PVs in that widget. If there are multiple PVs for that widget, select the PV of your interest using the top dropdown selector. For Yamcs parameters, you will see various properties that were defined in the Mission Database.

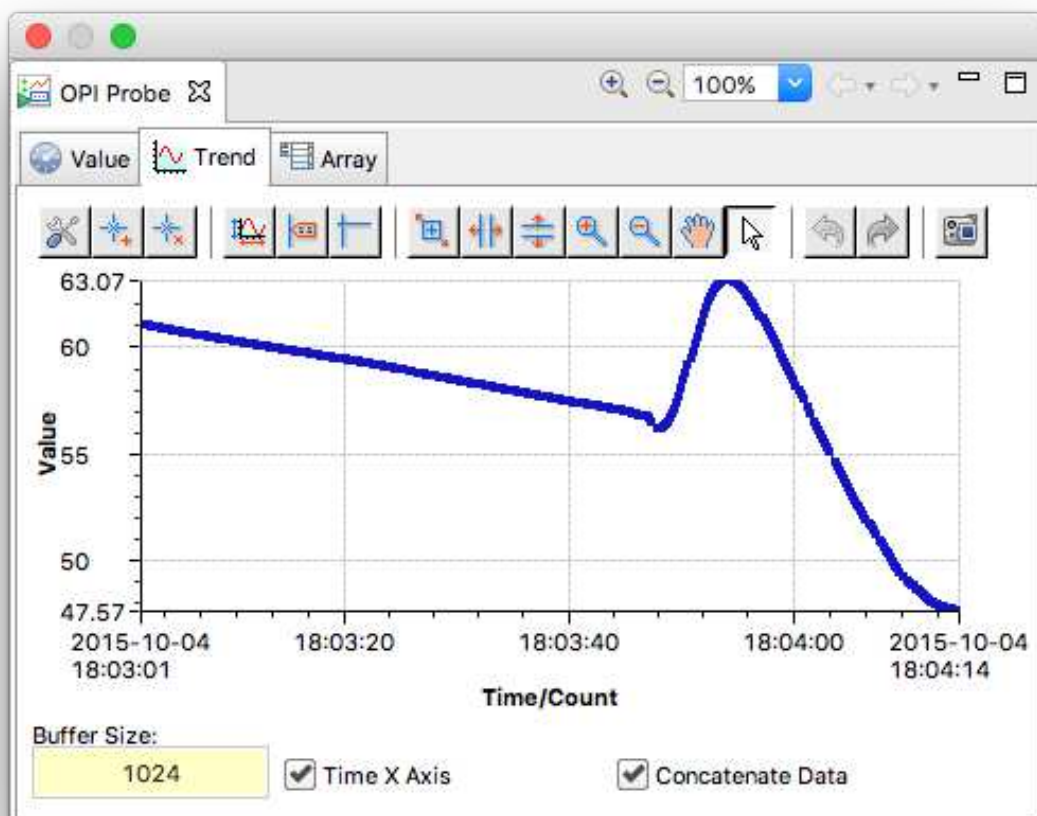
Currently the displayed information is mostly static, but we plan on improving this view to:

- show continuously updated information on the latest value, and its alarm info.
- show which context is applicable based on the latest value, for context-dependent Mission Database definitions (e.g. *alarm rule x only applies in contingency mode*).



OPI Probe

Right-click on a widget backed by a PV, and select **Process Variable > OPI Probe**. This opens the OPI Probe view which shows you in the tab **Trend** a graphical evolution of this PVs value. There is currently no way to navigate to archived data.



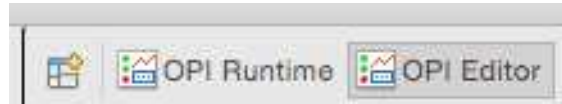


Given the similarities, in the future, we are likely to bring the content provided by the PV Info and OPI Probe windows together in one dialog. We also foresee improvements to explore archived data.

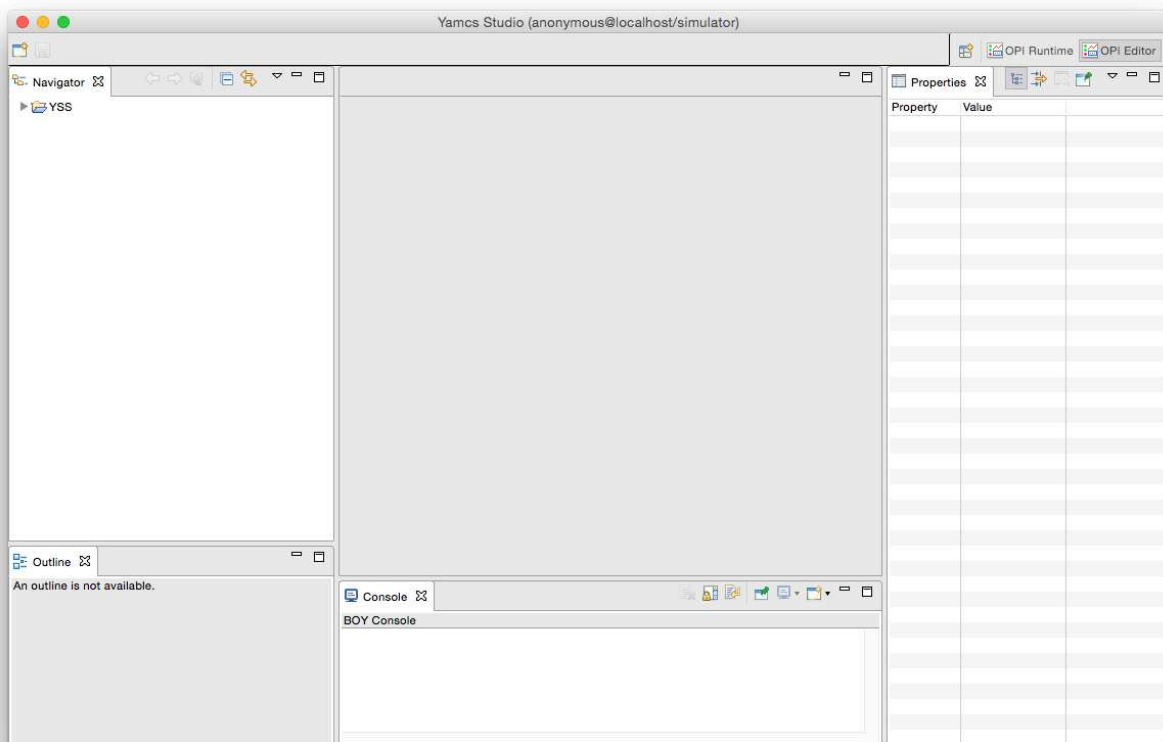
Chapter 3. Editing Displays

3.1. OPI Editor Perspective

The OPI Editor perspective is used to create or edit displays. In the top right, change your copy of Yamcs Studio to OPI Editor mode (in case you don't see it, choose it from the dialog that opens up when clicking the plus-icon).



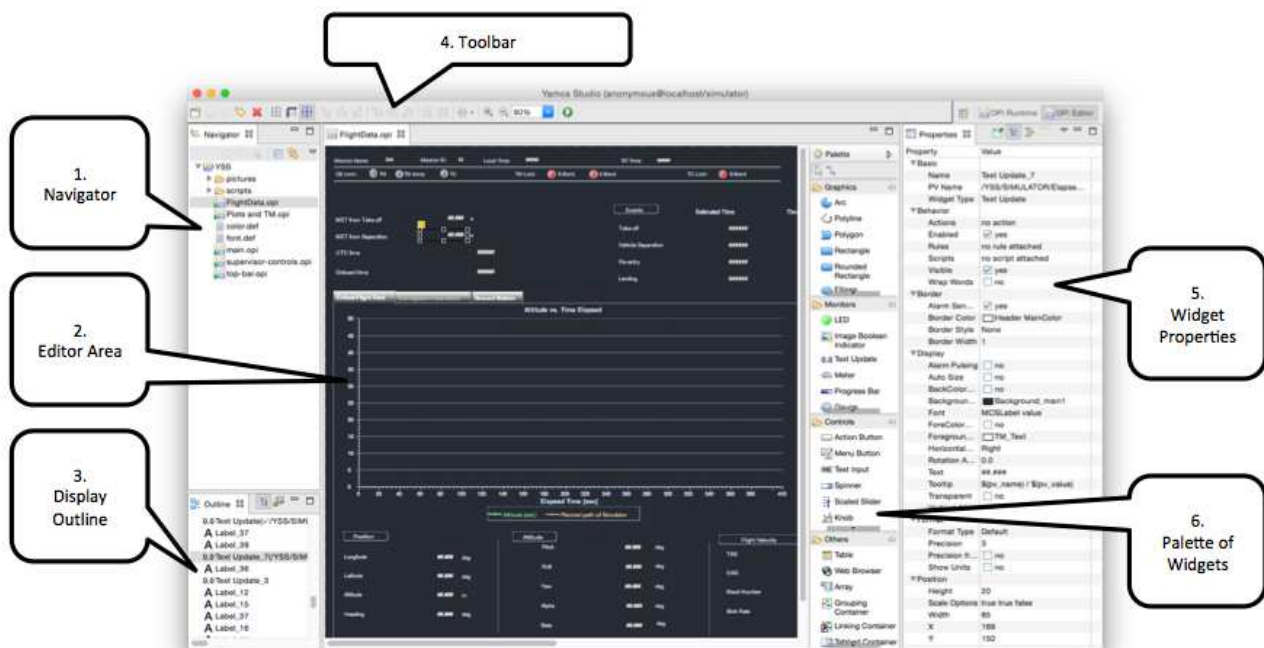
Your window arrangement changes to something like this.



In the left navigator, expand the YSS project and open for example our `FlightData.opi` by right-clicking and choosing **Open With > OPI Editor**.

Note: we are aware that right-clicking it is slightly annoying. The left-click action by default opens the OPI file with OPI Runtime. Once you've successfully opened an OPI with the OPI Editor the left-click action will from that point always open it with the OPI Editor, as it remembers its last handler. We definitely want to improve the user experience here. But for now, please bear with us as we do the needed development work.

The window layout can be decomposed like this:



1. Navigator

The navigator contains all projects within the current workspace. In general a project is at the same level as a mission, but this is not strictly necessary. When we launch Yamcs Studio with a new workspace, it will always automatically create the YSS project. Once you have added your own project, you can remove YSS and it won't be autocreated anymore.

A project contains Operator Displays (*.opi), images, color profiles (*.def), custom scripts (*.js or *.py), etc. Familiarise yourself with the right-click option as you go about opening displays. Displays can be opened in a few different modes within the OPI Editor.

- In editing mode
- In runtime mode in a Standalone window (*beta*)
- In runtime mode within the workbench itself (this will split your window to make room for it)
- In a new window using the green launch button in the toolbar

It is useful to have all these options when you're in the process of editing and testing displays with realtime telemetry, but do pay attention to treat the OPI Editor like an editor, not like a runtime viewer. During operations you should switch back to the OPI Runtime.

2. Editor Area

This area contains tabs for every OPI that was opened for editing. This offers familiar editing controls. Widgets can be selected, grouped, dragged and deleted to your personal taste.

3. Outline

The Outline view gives a hierarchical breakdown of all the widgets within the currently active editor tab. It is useful for finding back widgets. Widgets that were named will be easily identifiable.

4. Toolbar


The toolbar offers context-sensitive controls. This includes general *Save* functionality, as well as handy features like grid toggling or space distribution among different widgets.

5. Properties

This view shows the properties of widgets (or of the display itself). Notable properties include the **PV Name** which allows you to connect a widget with a specific Yamcs parameter (with autocompletion support). Other properties allow the display author to greatly tweak default widget behaviour. And in cases where the properties are not sufficient, we can always escape to more customization options using scripts and rules (there are properties for adding these as well).

6. Palette

The palette contains the widgets that are available in your copy of Yamcs Studio. Select a widget from the Palette, and then click somewhere in the editor area to place it down.

When you're done doing changes, make sure to save them (**File > Save All**). You can now test out your changes by clicking the launch button  from the toolbar.

This will open a new runtime window (notice it uses the OPI Runtime perspective). If you leave this window open, and you save more changes, do a right-click in your display tab and choose **Refresh OPI**. You will do this a lot as you go about editing displays. You can also refresh by hitting **F5**, but make sure that your display actually has focus (for example by clicking somewhere in the editor before hitting **F5**).

3.2. Resource Management

Workspace

Projects

Searching

Projects

Within a workspace we can have one or more **projects**, which provides a way to group similar resources together. For many missions, having just one project is more than enough.

Opening a display

Customizing Yamcs Studio









3.3. Widget Properties

This documents the default widgets that come with Yames Studio. Depending on your particular version of Yames Studio you may have additional widgets at your disposal.












3.4. Palette

These are the default widgets that come with Yamcs Studio. Depending on your specific version of Yamcs Studio you may have less or more widgets at your disposal.
















Graphics

 Arc	 Rectangle	 Label
 Polyline	 Rounded Rectangle	 Image
 Polygon	 Ellipse	









Monitors

 LED	 Progress Bar	 XY Graph
 Image Boolean Indicator	 Gauge	 Intensity Graph
 Text Update	 Thermometer	 Byte Monitor
 Meter	 Tank	

Controls

 Action Button	 Knob	 Image Boolean Button
 Menu Button	 Scrollbar	 Check Box
 Text Input	 Thumb Wheel	 Radio Box
 Spinner	 Boolean Switch	 Choice Button
 Scaled Slider	 Boolean Button	 Combo

Others

 Table	 Grouping Container	 Sash Container
 Web Browser	 Linking Container	 Grid Layout
 Array	 Tabbed Container	

Graphics

Arc

Draws an arc contained by the bounding box. To modify the size of the arc, modify the **X**, **Y**, **Width** and **Height** properties.

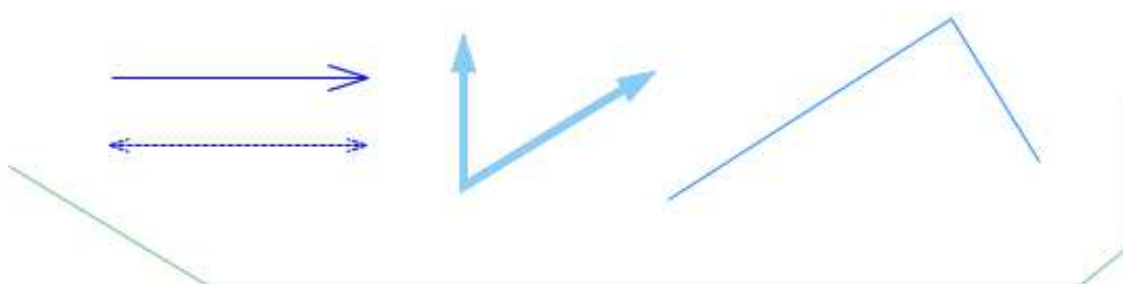
By default the arc starts with an angle of 0 degrees (right center). Modify this property using **Start Angle** (in degrees). The arc length is determined using **Total Angle**.



Polyline

Use the polyline tool to draw lines of two or more points. After selecting the widget from the Palette, start by clicking on your desired start location. Every next click will add a new point. Double-click if this is your last point. You can modify your points by moving around the yellow handles. Finetune your points using the **Points** properties.

Polylines have a direction and can be decorated with arrow heads using the various **Arrow** properties.



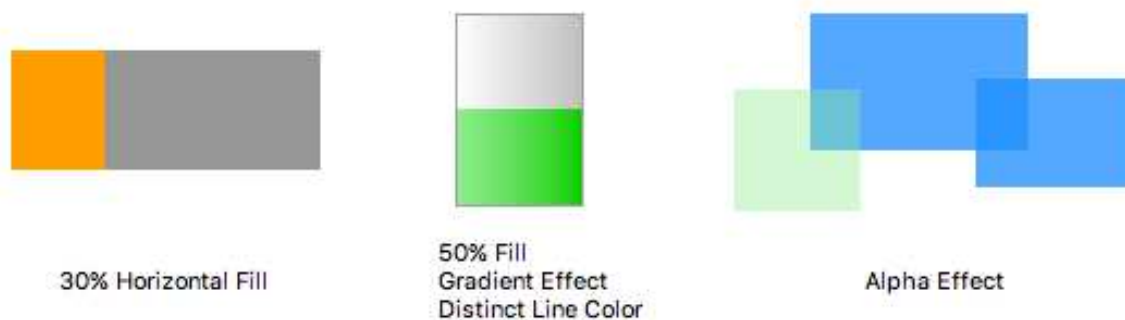
Polygon

Similar in functionality to polylines, but the last point is always connected to the first point to form a closed shape. Unlike polylines, polygons are not directed. They can be filled from left to right or from bottom to top using the **Fill Level**, and **Horizontal Fill** properties. In the section Rules we'll learn that this (or just about any other property) can also be dynamically modified.



Rectangle

Rectangles are constrained polygons. They can be customized in much the same way as polygons. Unlike polygons the background and foreground can also be colored using gradients which makes them look less flat.



Rounded Rectangle

Rounded rectangles are a specialised form of Rectangles. They support additional properties for controlling **Corner Height** and **Corner Width**.

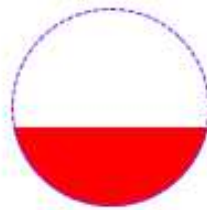


Ellipse

The Ellipse's shape is determined by its bounding box using the **Width** and **Height** properties. Ellipses support similar properties as rectangles.



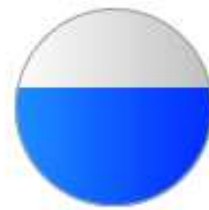
Width: 90
Height: 90
100% Fill



Width: 90
Height: 90
40% Fill
Dashed Line



Width: 140
Height: 70
100% Fill
Transparent



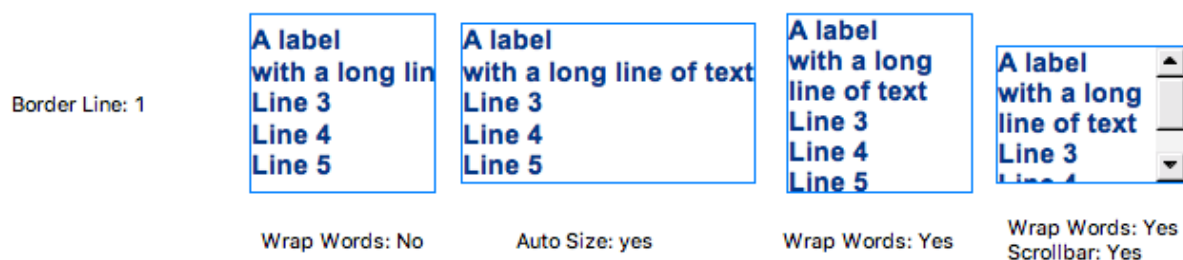
Width: 90
Height: 90
60% Fill
Line + Gradient

Label

The Label widget allows displaying text within a bounding box indicated by the **X**, **Y**, **Width** and **Height** properties. Adapt your bounding box to the content using **Auto Size** property.

For long texts consider using the **Wrap Words** and **Show Scrollbar** properties.

To edit the text of a label, either select the **Label** property to open an input dialog, or press **F2** to edit the label in-place.



Image

Yamcs Studio supports GIF, JPG, PNG and BMP images. Images must be added to your project before you can select them with the **Image File** property. Transparency is automatically supported as well, but make sure to set the background of the Image widget to the same color as the container.



PNG
Background: Green



PNG
Background: White



Auto-Size: No
Stretch To Fit: Yes
Angle: 270

Monitors

LED

By default LED widgets are dark green, and light up when the connected PV is non-zero. Using the **State Count** the number of states can be increased to more than two. When doing so, a number of extra properties will be added in the Properties view. You then need to set explicit colors per value.

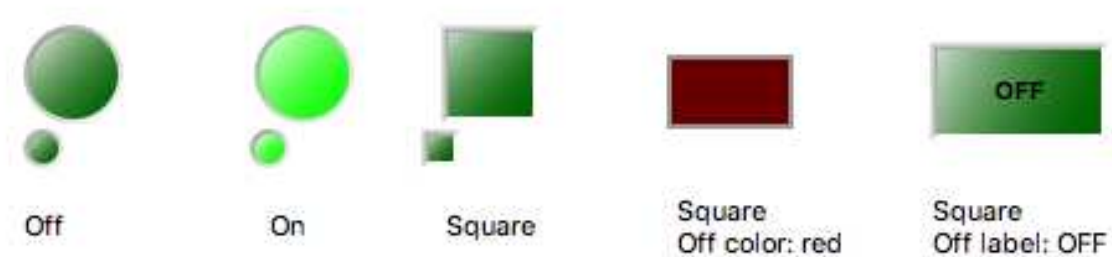


Image Boolean Indicator

With the Image Boolean Indicator you can render images depending on the ON or OFF state of the connected PV. The images need to be part of your workspace.

Among other use cases, this widget provides a practical way to replace the default LED widget with your own custom styling.



Text Update

The Text Update widget shows the textual value of the connected PV. The format can be controlled using properties like **Precision** and **Format Type**.

Both the background and the foreground color can be made alarm-sensitive. This will then apply the alarm colors that are defined in the color scheme.

Like Labels, Text Update widgets are drawn in a bounding box which you can control on the canvas by dragging the handles, or by finetuning the properties **Width** and **Height**.

Rounded Border Format Type: Hex	0x5876
Ridged Border Format Type: Exponential	2.2646E4
Green Background Format Type: Decimal	22,646

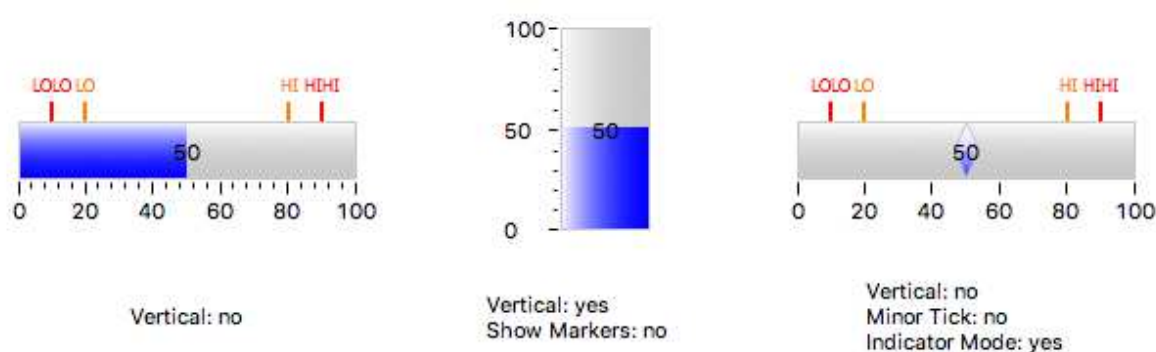
Meter

The Meter widget shows the numeric value of the connected PV on a meter. The exact display of this widget can be controlled using various properties, among which colors for the needle, LOLO, LO, HI and HIHI values, major tick separation, as well as various toggles for the different elements that a meter is composed of.



Progress Bar

A bar graph widget that shows the numeric value of the connected PV as a bar or indicator.



Gauge

The Gauge widget shows the numeric value of the connected PV on a gauge. The exact display of this widget can be controlled using various properties, among which colors for the needle, LOLO, LO, HI and HIHI values, major tick separation, as well as various toggles for the different elements that a gauge is composed of.



Show Ramp: yes



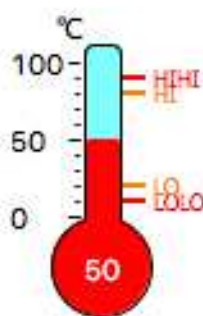
Show Ramp: yes
Ramp Gradient: yes



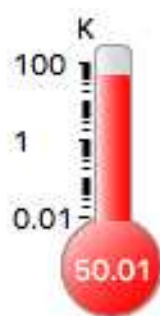
Show Ramp: yes
Ramp Gradient: no
Log Scale: yes

Thermometer

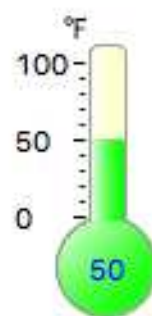
Specialised form of the Progress Bar widget that shows the numeric value of the connected PV as a thermometer.



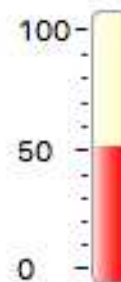
Show Markers: yes



Show Markers: no
Log Scale: yes
3D Effect: yes



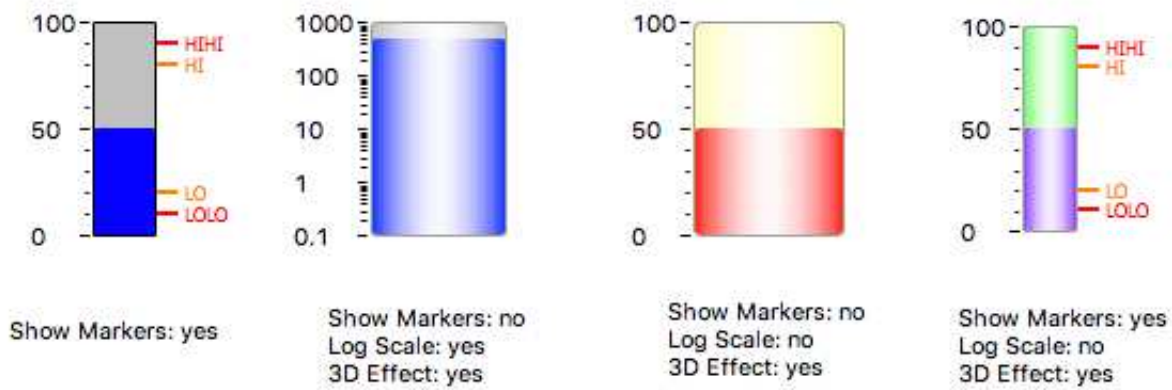
Show Markers: no
Log Scale: no
3D Effect: yes



Show Bulb: no

Tank

Specialised form of the Progress Bar widget that shows the numeric value of the connected PV as a tank.

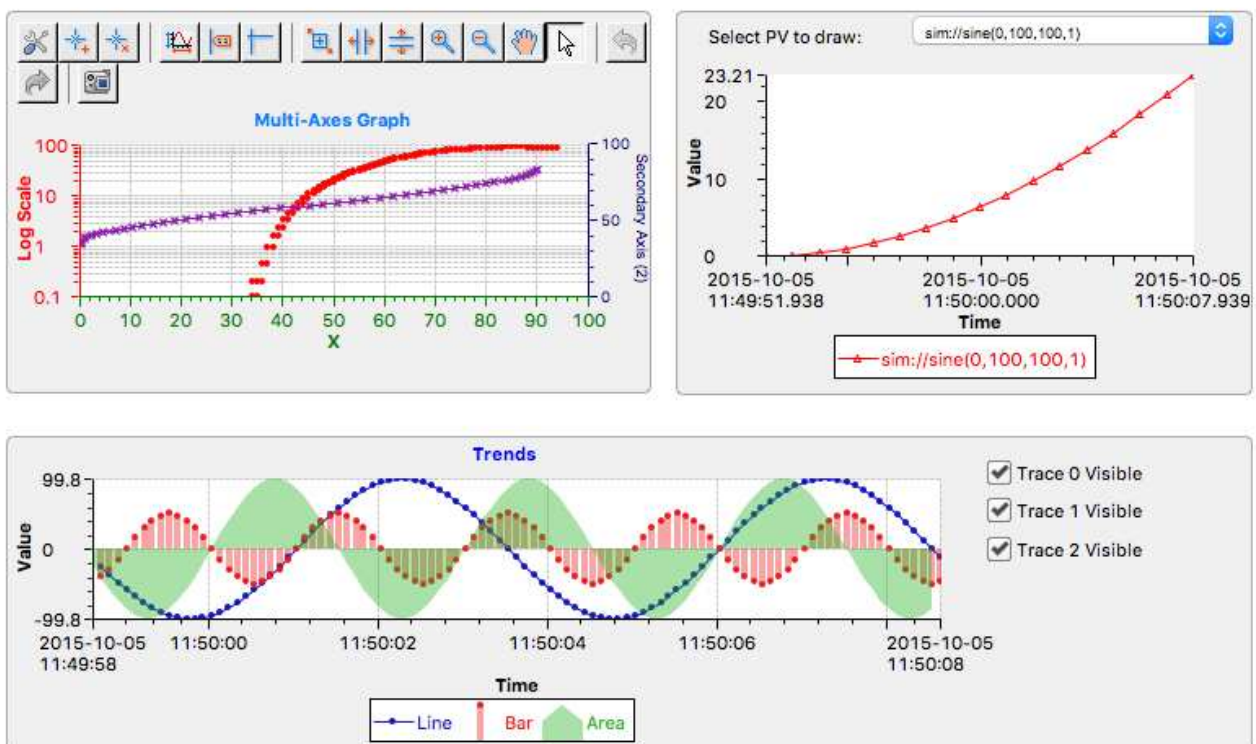


XY Graph

Advanced widget for plotting the numeric value of one or more connected PVs. By combining the different properties it supports line charts, scatter charts, bar charts, step charts and area charts.

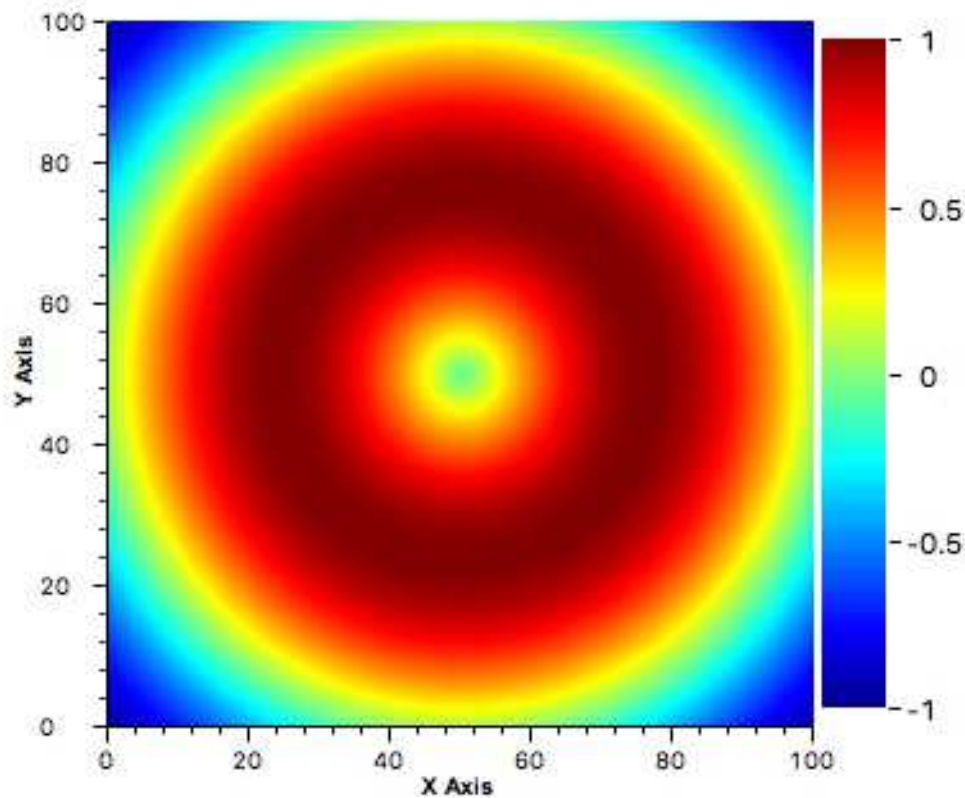
To show multiple traces in the same chart, increase the **Trace Count**. This will add a set of properties for every added trace. Every trace can be linked with a separate PV.

Use the **Show Toolbar** property to toggle visibility of a toolbar. When visible, this toolbar can be used by the operator to perform zooming operations on the plot.



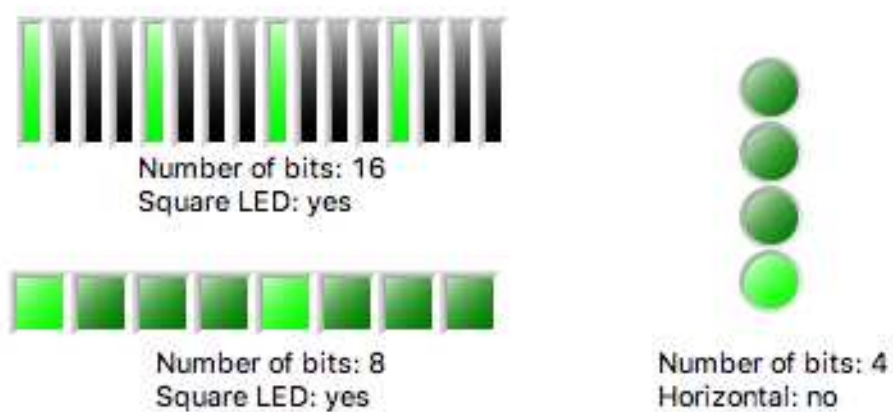
Intensity Graph

Advanced widget for displaying a 2D array as an image. This widget is meant to be used in combination with scripts. Further documentation pending.



Byte Monitor

Specialised form of the LED widget that shows the numeric long value of the connected PV as a series of LEDs, each LED lighting up if the corresponding bit is `true`. The rightmost or bottom bit corresponds with the least-significant bit of the PV, but this can be reversed using the **Reverse Bits** property.



Controls

Action Button

A button that can be used to trigger Actions. Use the **Toggle** property to have the option of adding a different action on press and release.



Menu Button

A button that will show a menu when it's clicked. The menu will be filled with either the actions from **Actions** property or the string values from the connected enumerated PV.



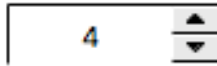
Text Input

A widget that allows the user to write data to the connected (writable) PV. For dates set the **Selector Type** to **Datetime**.



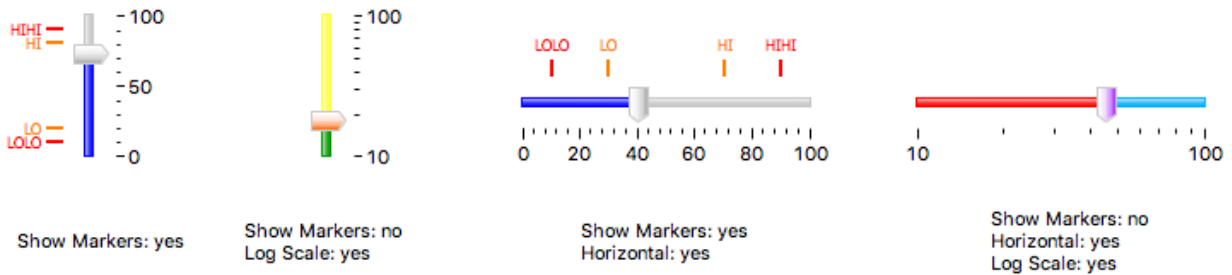
Spinner

The Spinner widget is similar to the Text Input, but allows updating the PV in incremental steps using up and down arrow buttons.



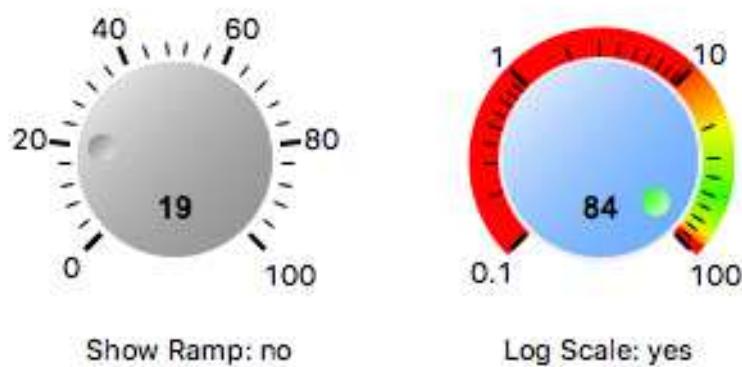
Scaled Slider

This widget allows adjusting the connected PV value using a configurable slider.



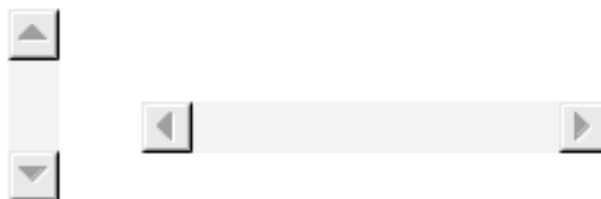
Knob

Knobs allow adjusting the connected PV value by dragging the thumb around.



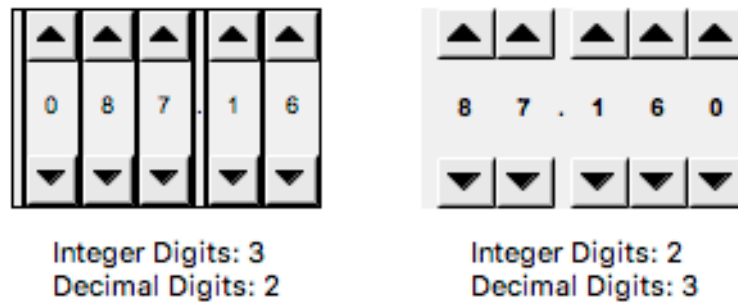
Scrollbar

Scrollbar that allows adjusting the connected PV value by. Useful as a building block for advanced display customizations.



Thumb Wheel

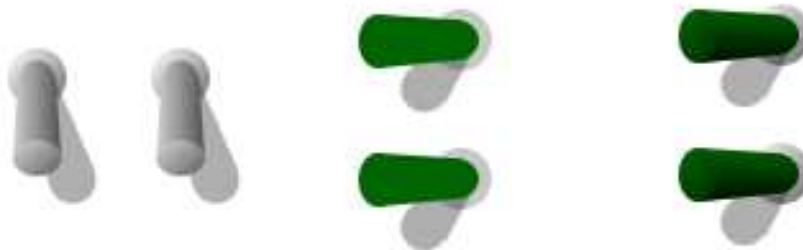
A widget that allows adjusting the decimal value of a PV digit by digit.



Boolean Switch

A Boolean Switch is able to write 0 or 1 to a single configurable bit of the connected PV or the whole PV if the **Bit** property is set to -1.

If **Width** is greater than **Height**, the switch will render horizontally, otherwise vertically.



Boolean Button

A Boolean Button is able to write 0 or 1 to a single configurable bit of the connected PV or the whole PV if the **Bit** property is set to -1.

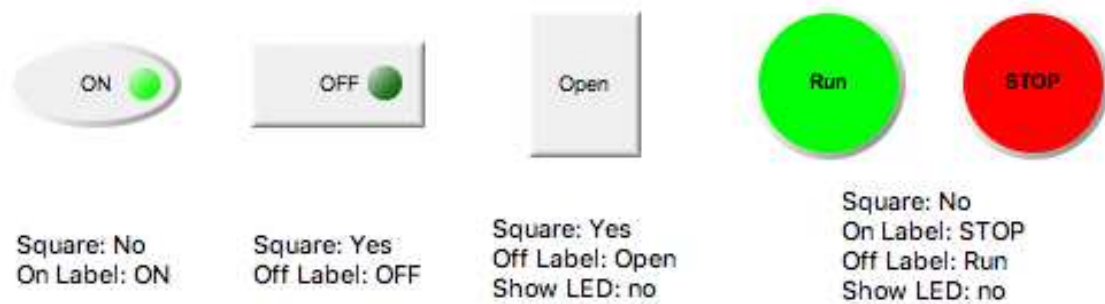


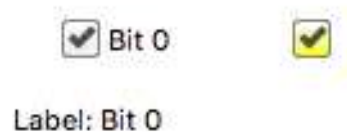
Image Boolean Button

Same as the Boolean Button widget, but you can customize the look further by specifying your own images with the **Off Image** and **On Image** properties. The selected images must be present in the workspace.



Check Box

A Check Box is able to write 0 or 1 to a single configurable bit of the connected PV or the whole PV if the **Bit** property is set to -1. Use the **Label** property to render a value next to the check box.



Radio Box

A Radio Box allows the user to choose between a configurable set of values. Use the **Items** property to specify these values.



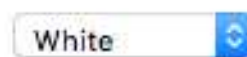
Choice Button

Similar to the Radio Box, but with buttons. Toggling a button within the widget, untoggles the previously active button.



Combo

The Combo widget, as well, makes the user choose between one of its **Items**.



Other Widgets

Table

Advanced widget for structuring tabular data. Does not connect to PVs by itself. Its main use comes when combined with scripts.

Col 1	Col 2	Col 3	Col 4	
<input type="checkbox"/> row 1	row 1	row 1	row 1	
<input type="checkbox"/> row 2	row 2	row 2	row 2	
<input type="checkbox"/> row 3	row 3	row 3	row 3	
<input type="checkbox"/> row 4	row 4	row 4	row 4	

Web Browser

The Web Browser widget allows incorporating a web browser inside a display. This may be useful in managed environments, but in general we would recommend using a standalone web browser. The option is there though. Choose the start page with the **URL** property.

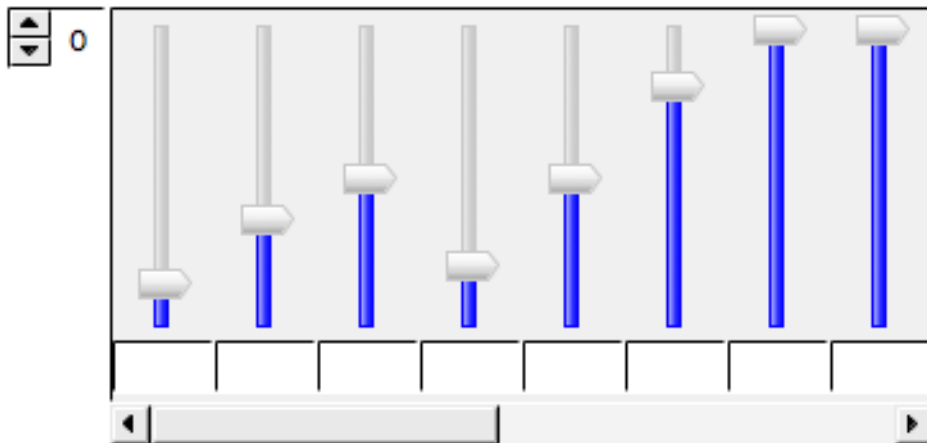


Array

Container widget that repeats an other widget. First create an Array instance, than drag another widget on top of it and release. The widget will be repeated for the defined **Array Length**.

Modifying the property of any child modifies it for all the other children as well.

The Array widget must be connected to a an array PV.

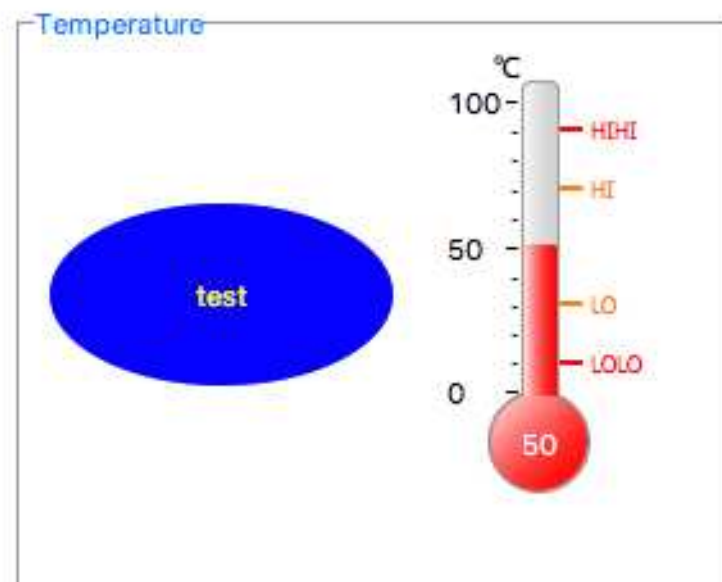


Grouping Container

You can group widgets together using a Grouping Container. Drag any widget on top of it to make it part of the group. Being widgets themselves, grouping containers can have their own backgrounds and borders.

To remove a widget out of a group, select it and drag it out of its container. To remove a group, right-click on it and select **Ungroup**. To add existing widgets to a new group, select them, and then choose **Group** from the right-click menu.

To prevent nested widgets from being individually selectable, set the **Lock Children** property of the group to yes.



Linking Container

Displays can include other displays, or groups of other displays. This is particularly useful to avoid duplication when creating shared components, such as top bars or side bars.

Define the included OPI file with the **OPI File** property, then optionally define the group with the **Group Name** property. If you don't specify a group the entire display is embedded. Tweak properties such as the **Resize Behaviour** to get your desired outcome.

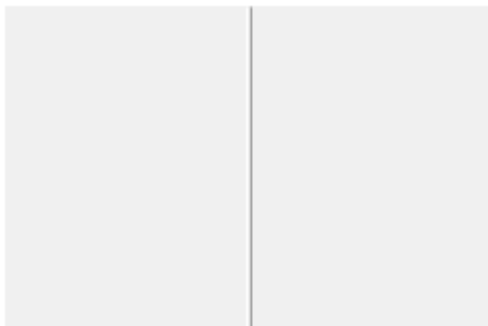
Tabbed Container

Creates a stack of tabs. Define the number of tabs with the **Tab Count** property.

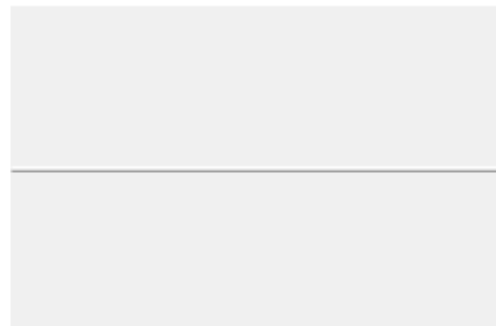


Sash Container

Use the Sash Container to create resizable split panes.



Horizontal: yes



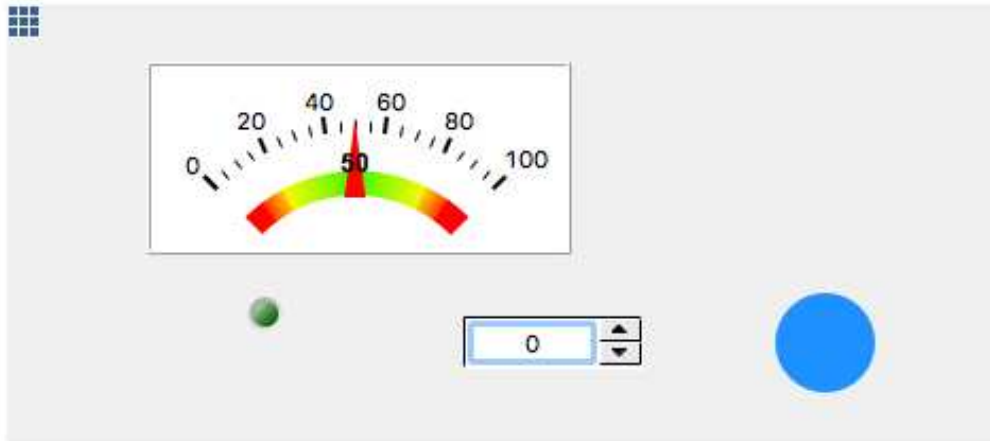
Horizontal: no

Grid Layout

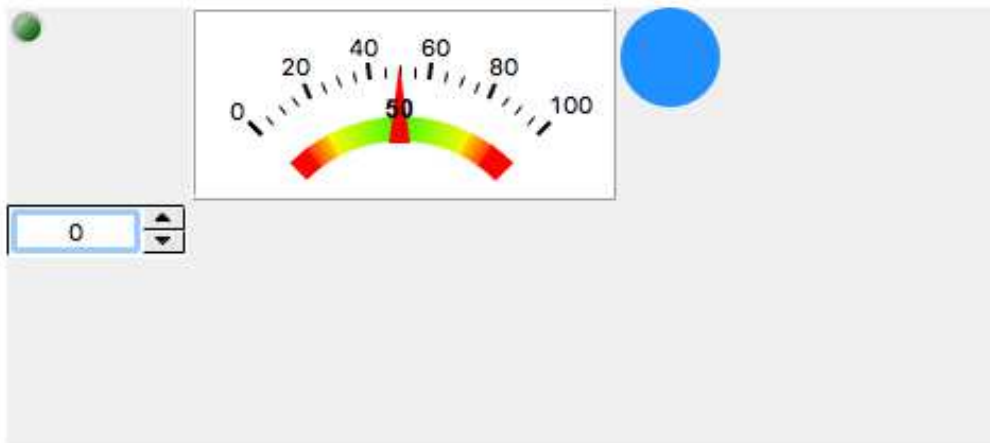
This is a special type of widget that is meant to be attached to a container or to the display itself. At runtime it reorganizes the children of that container in a grid. This usually works best with similarly sized widgets.

Create a Grid Layout by selecting it from the Palette, and clicking on top of the targeted container. You'll notice a grid decoration element in the top left of the container.

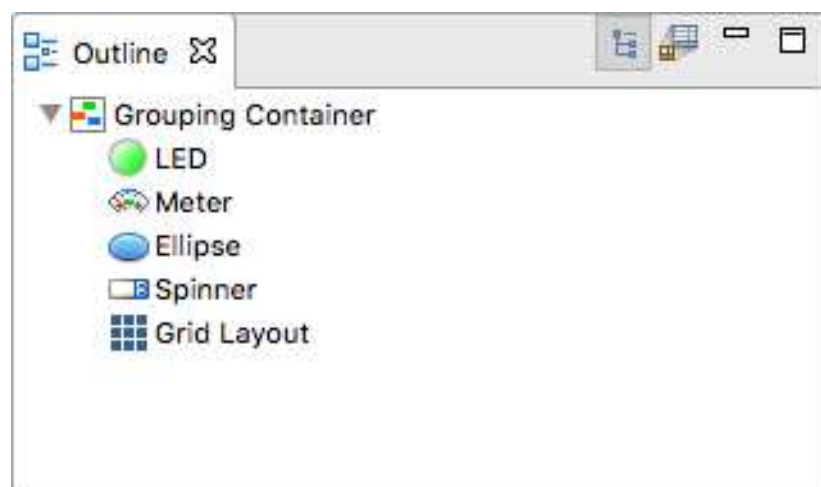
For example, while editing it may look like this:



But at runtime, with 3 columns, it renders like this:



The order of the container's children is important for this layout, and can be modified in the **Outline** view.



3.5. Rules & Scripts

abc

3.6. Theming

bla

Chapter 4. Views

4.1. Archive

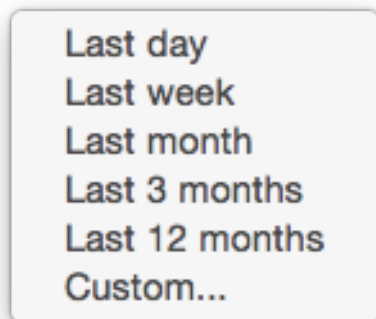
The Archive view represents a visual view of the data stored in the Yamcs archive. Through this view we can also initialize and control replays of archived telemetry.

User Interface

We are aware that the current user interface takes some getting used to. The Archive view always works on a range of indexed data, which it fetches from the server. All further actions like zooming happen client-side on the loaded data range.

Choosing a Data Range

As a first step you should select your data range. Click the pull-down icon  to bring up this menu:



You can choose one of the predefined half-open time intervals, or you can select **Custom...** to specify your preferred range. Ranges can be half-open, which means they will always grow to include more bordering data as it becomes available.

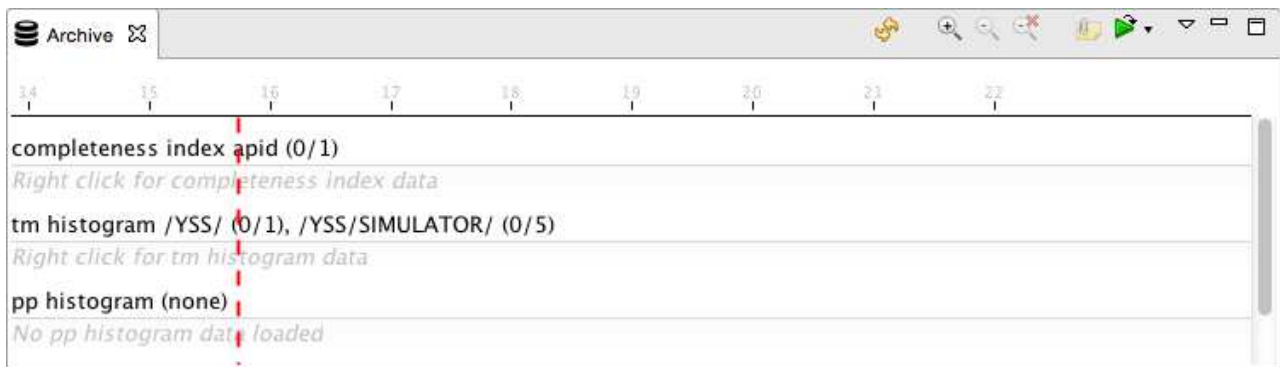
If you choose for example **Last Day**, Yamcs Studio will fetch an index of the archive for that time period, and refresh your view.

Your chosen data range is stored in your user preferences and will be restored the next time you open Yamcs Studio.

Selecting Data

If this is the first time you have opened Yamcs Studio on your workstation, you will still not be able to see anything, other than some empty zones named:

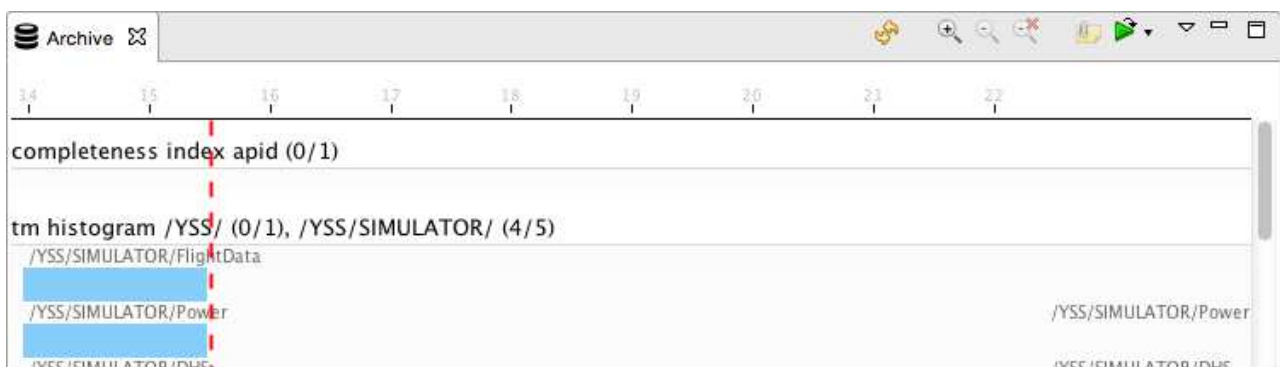
- Completeness Index APID
- TM Histogram
- PP Histogram
- CMDHIST Histogram



You need to choose which index data you actually want to display in your view. If there is data available for a zone, you can right-click it to bring up a pop-up menu where you select **Add Packets** > ... > **All Packets**. Your view will then update to show the selected packets.





We say *packets* since this is typically what we're interested in when browsing the Archive, but really it covers any kind of data that can be displayed through the Archive view.




Note that the view does not refresh itself, so hit  **Refresh** whenever you want to load the latest data for your selected time range.

Navigating

The vertical red locator shows the current time as provided by Yamcs. When we hover the mouse over the view, a grayed-out locator indicates the current mouse position.

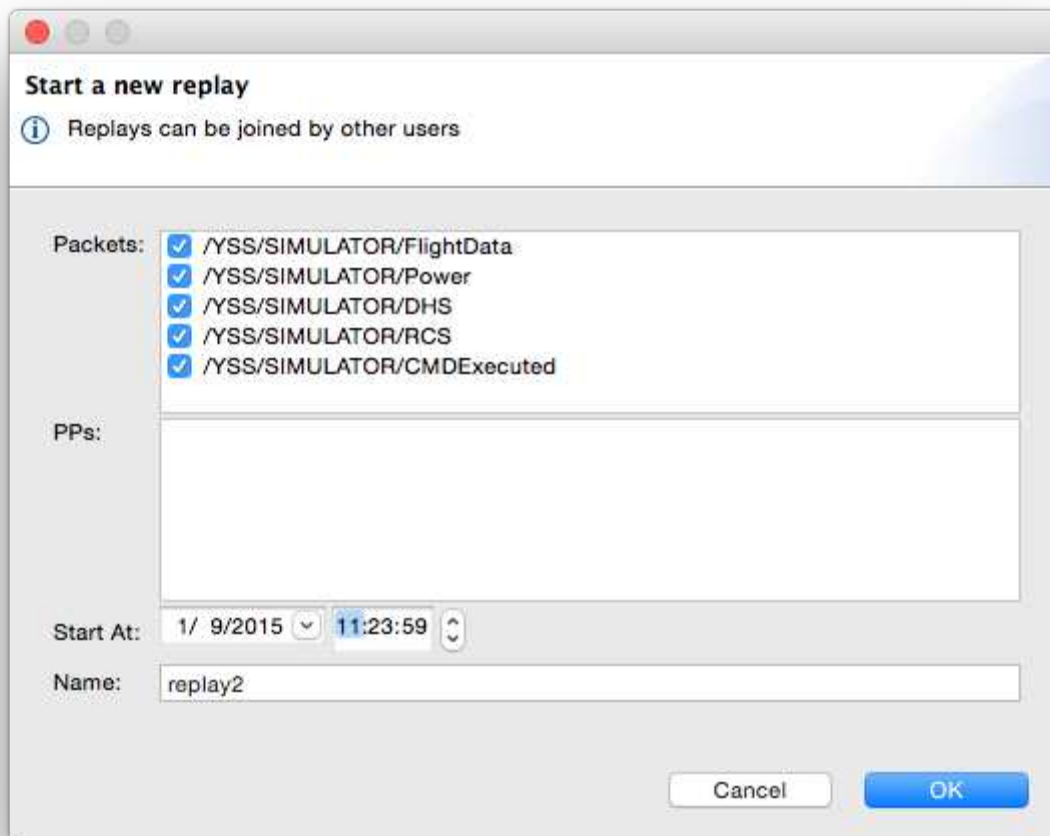
You can **Zoom In**  or **Zoom Out** . If you are interested in a specific range, then select a time range by clicking and dragging your mouse before you zoom in.

Notice as you are zooming that a horizontal scroll bar appears. This allows you to scroll left and right within the initially load time range.

To clear your zoom stack, select **Clear Zoom** .

Replaying data

We can use this view to replay archived data. Click **Replay** .



In the dialog, confirm or filter the suggested selection of data. Currently only telemetry packets and processed parameters that were made visible in the Archive are part of the selectable data.

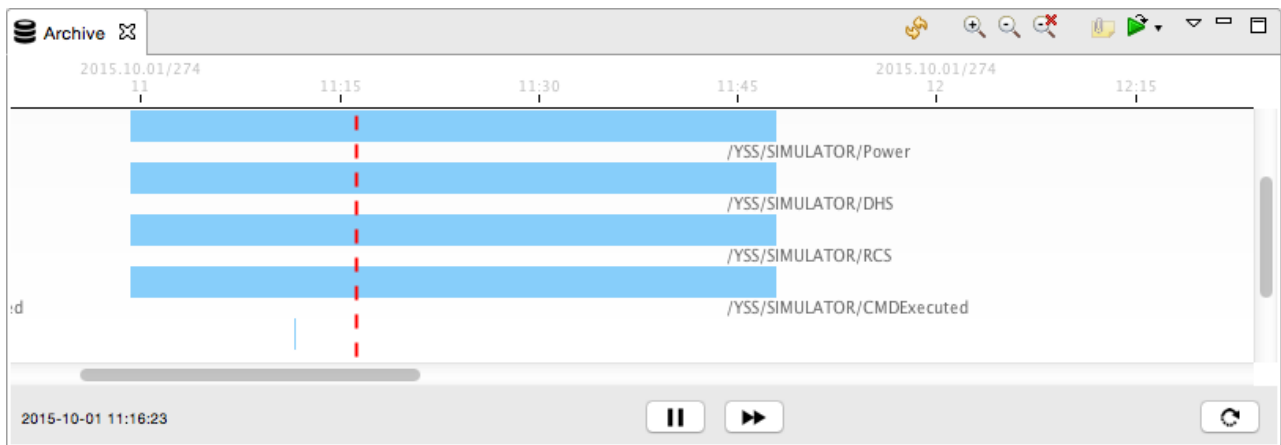
Modify **Start At** to where you want to start the replay from.

Yamcs will create a processor (next to the built-in *realtime* processor) with the name that you provide in the **Name** field. The exact name that you choose is of no importance (although it needs to be unique), it helps you to identify the processor.

Click OK to start the replay. Yamcs Studio will reset any operator displays you may have opened, and will automatically switch to the newly create replay processor, as visible in the processor indicator in the top-left of your window.



Notice also that the Archive view is now equipped with an additional control bar.



The red locator shows you the current time of the replay processing. Double click anywhere to the left or to the right to make the processing jump to another point in time.

Click **Pause** || to pause the processing, and use **Forward** ▶▶ to increase the speed of the processing. This button currently cycles through 5 predefined speed steps.

▶▶	Original Speed
▶▶ _{2x}	2x Original Speed
▶▶ _{4x}	4x Original Speed
▶▶ _{8x}	8x Original Speed
▶▶ _{16x}	16x Original Speed

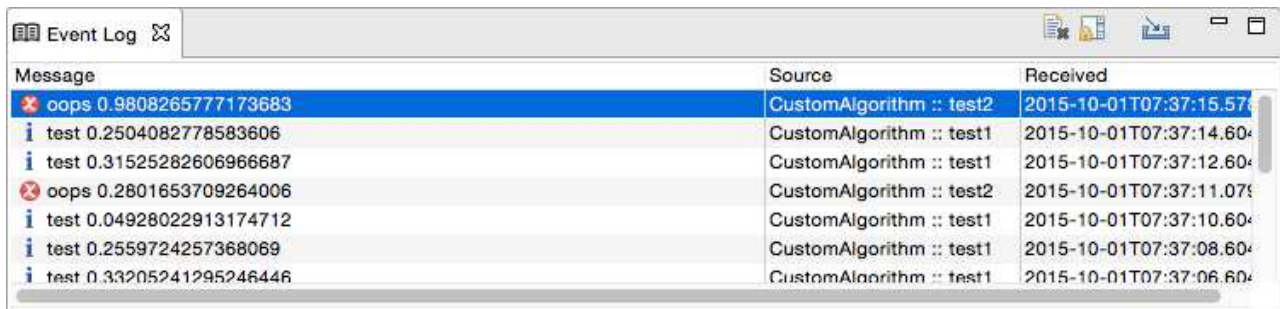
Speeding up will not cause any reset of your displays, as the same data is arriving, just faster.

When you want to leave the replay, there are several ways to do so:


- Hit **Return to Realtime** ↺ ;
- Open the pull-down menu ▼ next to the **Replay** ▶ button to choose a different processor;
- Click on the processor info bar in the top left of the window, to choose a different processor.

4.2. Event Log


The Event Log view displays events from Yamcs Server. This could be on-board events, or events generated by Yamcs itself, whenever something significant occurs.



Message	Source	Received
oops 0.9808265777173683	CustomAlgorithm :: test2	2015-10-01T07:37:15.574
test 0.2504082778583606	CustomAlgorithm :: test1	2015-10-01T07:37:14.604
test 0.31525282606966687	CustomAlgorithm :: test1	2015-10-01T07:37:12.604
oops 0.2801653709264006	CustomAlgorithm :: test2	2015-10-01T07:37:11.074
test 0.04928022913174712	CustomAlgorithm :: test1	2015-10-01T07:37:10.604
test 0.2559724257368069	CustomAlgorithm :: test1	2015-10-01T07:37:08.604
test 0.33205241295246446	CustomAlgorithm :: test1	2015-10-01T07:37:06.604

To load events for an earlier time range, select  **Import**.

Clear your view by clicking  **Clear**. You can always re-import events again at a later moment.

When Yamcs Studio becomes aware of a new event, it will automatically select and reveal it. You can prevent this default behaviour by toggling the  **Scroll Lock**.

In a next iteration we plan to add features for exporting to CSV, adding a property view, and for filtering events, similar to how the current standalone Event Viewer tool already supports today.

4.3. Alarms

This view is under development, and will offer a centralised view of all active alarms, with the opportunity to acknowledge alarms or to mute any sounds. The Alarms view is not yet bundled in current copies of Yamcs Studio, but will be so in the short term.



Currently the only way to be noticed of alarms in Yamcs Studio is by following events (if your Yamcs Server is configured to report alarm state changes as events), or by manually iterating your displays to look for widgets that have red (= major alarm) or orange (= minor alarm) colored borders around them. Our upcoming Alarms view aims to improve this process.

4.4. Command Stack

The Command Stack allows operators to prepare stacks of commands for manual command issuing. The process is intentionally click-heavy to make sure that operators are aware of what they are doing.

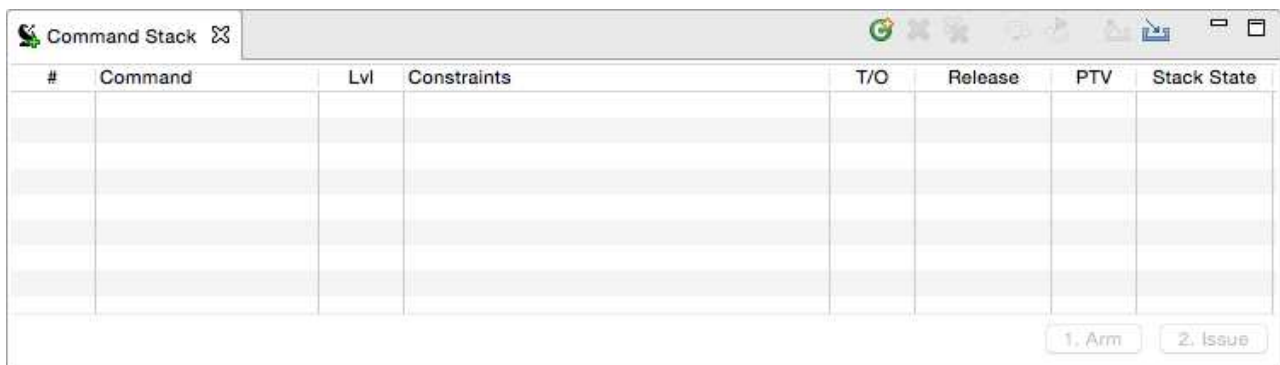
When you issue commands from Yamcs Studio, these are queued to Yamcs Server, which will perform any further steps.

We're keen on bringing many improvements to this view for better editing, but it is usable in its current state.

Preparing a Stack


You can prepare a stack of commands only when you are connected to Yamcs. Yamcs Studio uses this connection to retrieve the list of available commands or to perform server-side validations.

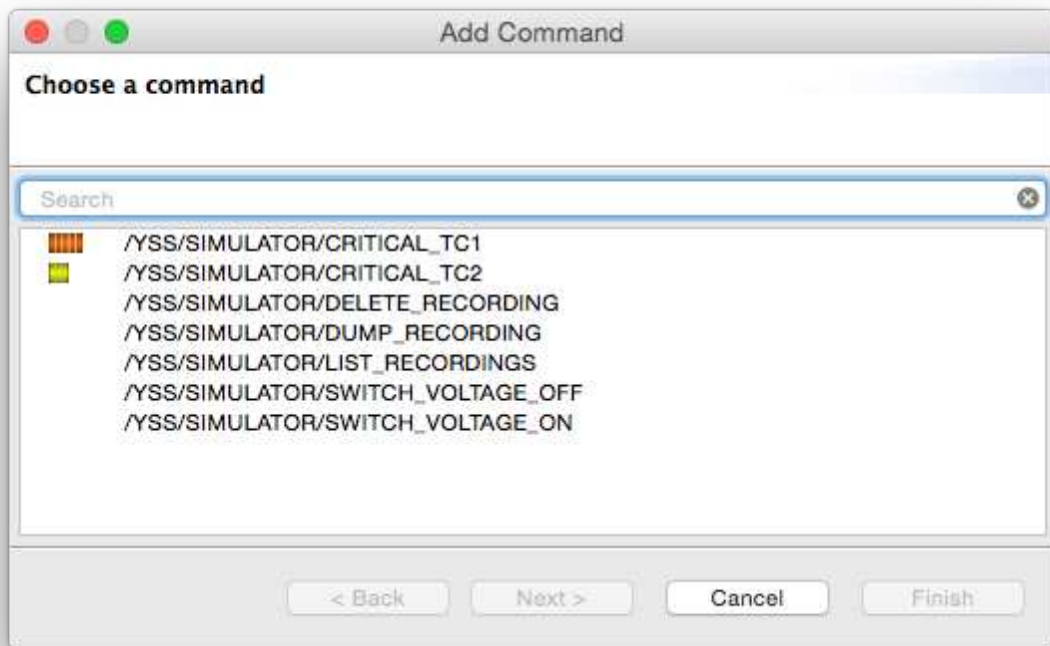
When you start Yamcs Studio, the Command Stack view (available from the OPI Runtime perspective) is by default shown below the operator displays. If you can't find it back, select **Window > Show View > Command Stack**.



#	Command	Lvl	Constraints	T/O	Release	PTV	Stack State

1. Arm 2. Issue

Add a command by clicking the  **Add Command** button.



This opens a wizard dialog with the list of available commands. You can filter the list with the search box on top.

Commands are identified by their fully qualified XTCE name. This name matches the hierarchical structure of the commands as defined in the mission database of the connected Yamcs instance. In future versions we may include a tree representation in addition to the current flat representation.

Commands can have varying levels of criticality (called *significance* in XTCE parlance). The icon in the leftmost column indicates the defined criticality for the command.

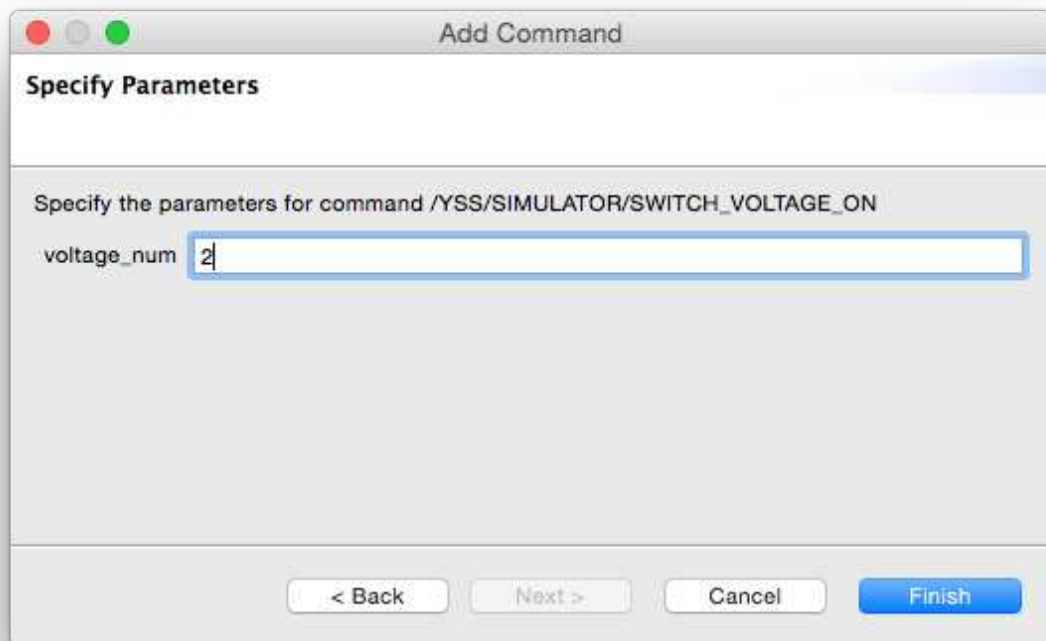
Icon	Criticality
	Watch
	Warning
	Distress
	Critical
	Severe

If an explanatory message regarding the criticality was defined in the mission database, this will show in the top title area of the dialog when the command is selected. Currently, only numbers or text can be entered.

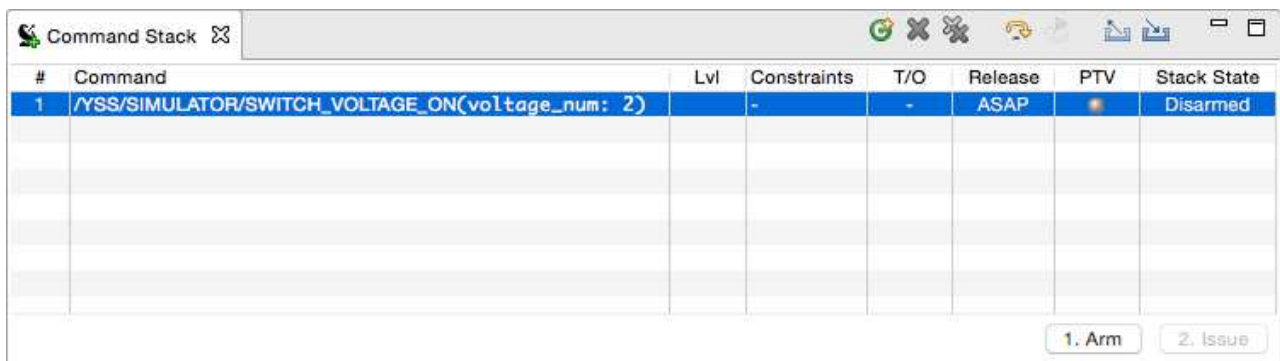
Once you've chosen your command, hit **Next** to go the next page in the wizard, where you can specify any arguments that need to be provided for the command.





You can close the wizard from the first page as well by clicking **Finish** instead of **Next**. If the command requires any arguments, you will have a chance to add them afterwards as well by editing your stacked command.



Click **Finish** to append your command to the end of your current stack.

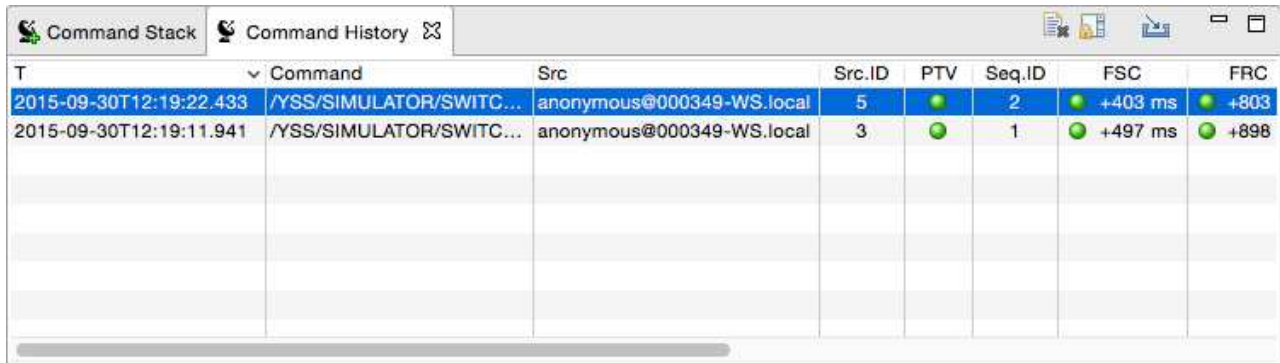


You can review your provided arguments by double clicking the command. To remove the selected command from the stack select  **Delete**. Clear the entire stack with  **Delete All**.

If a stacked command does not pass static validation checks (sometimes referred to as SPTVs – Static PreTransmission Verification) it will be marked with error indicators. This will prevent the user from attempting further execution of the stack until the error is resolved.

4.5. Command History

The Command History keeps track of all commands that were issued using Yamcs (not just by yourself, but by anyone).



T	Command	Src	Src.ID	PTV	Seq.ID	FSC	FRC
2015-09-30T12:19:22.433	/YSS/SIMULATOR/SWITC...	anonymous@000349-WS.local	5		2	+403 ms	+803
2015-09-30T12:19:11.941	/YSS/SIMULATOR/SWITC...	anonymous@000349-WS.local	3		1	+497 ms	+898

It will by default only show commands that were received on the realtime processor since you started your copy of Yamcs Studio. To load the command history for an earlier time range, select **Import**.

Clear your view by clicking **Clear**. You can always import the cleared commands again at a later moment.

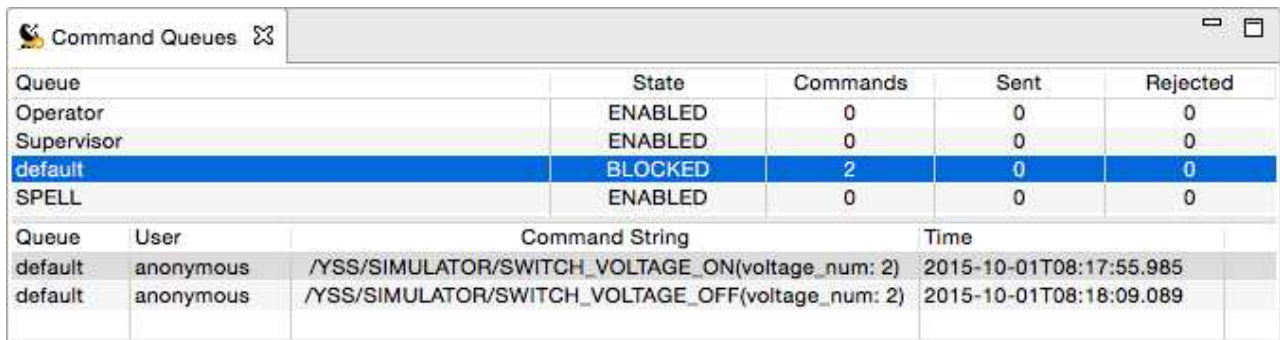
When Yamcs Studio becomes aware of a new command that was issued by Yamcs, it will automatically select and reveal it. You can prevent this default behaviour by toggling the **Scroll Lock**.

The displayed columns are as follows.

T	Time when the command was issued
Command	The command in textual format
Src	The origin of the command. Typically in <i>user@host</i> format
Src.ID	The ID that was given to the command by the issuing application. This number is assigned by the source application. In case of Yamcs Studio it is an incremental number that resets to 1 on every restart of Yamcs Studio.
PTV	Result of the Pretransmission Verification as performed by Yamcs. For example, some commands may only be applicable for 10 seconds and needs certain other parameters to be set to specific values. When the PTV bubble colors red, these type of context-dependent checks could not be validated, and therefore the command was not actually issued.
Seq.ID	The id that was determined by Yamcs before further dispatching the command. This is an incremental number that resets on every restart of Yamcs.
Further Columns	Indicate acknowledgments of ground hops as the command is being dispatched. The exact number and name of the columns depends largely on how Yamcs is deployed at your site. Yamcs typically calculates the state of these bubbles based on incoming telemetry. The bubble becomes green or red depending on the verification result. The column value shows the time difference with the issuing time <i>T</i> .

4.6. Command Queues

This view allows controlling the Yamcs queues from the point of view of Yamcs Server. With sufficient privileges, queues can be blocked or disabled.



Queue	State	Commands	Sent	Rejected
Operator	ENABLED	0	0	0
Supervisor	ENABLED	0	0	0
default	BLOCKED	2	0	0
SPELL	ENABLED	0	0	0

Queue	User	Command String	Time
default	anonymous	/YSS/SIMULATOR/SWITCH_VOLTAGE_ON(voltage_num: 2)	2015-10-01T08:17:55.985
default	anonymous	/YSS/SIMULATOR/SWITCH_VOLTAGE_OFF(voltage_num: 2)	2015-10-01T08:18:09.089

The view is split into two panels:

- The upper panel contains a list with all the defined command queues. The queue name, the current state and the number of commands that are currently in the queue is displayed. Right-clicking on a command queue opens a menu with the possibility to change the state of the queue.

In addition to setting the new state of the queue, the following additional actions happen when changing the state of a blocked queue:

- blocked → disabled: all the commands in the queue will be automatically rejected;
- blocked → enabled: all the commands in the queue will be automatically sent.
- The bottom panel contains the list of commands currently present in the queue selected in the upper panel. For each command the queue name, the command id, the user that has sent the command and the command string are display. The list is empty if the selected queue is not in the state blocked.

Right-clicking on a command gives the possibility to **Send** or **Reject** the command.

4.7. Yamcs Clients

The Yamcs Clients view is a readonly view that shows you all the clients that are connected to the same Yamcs Server that you are connected to.

ID	User	Application	Instance	Processor
51	unknown	Yamcs Studio v1.0.0.qualifier	simulator	realtime

The displayed information includes:

ID	The client ID assigned by Yamcs Server. Useful for debugging.
User	The name of the connected user, or <code>unknown</code> if Yamcs was not secured
Application	The name of the application that the user is using to connect to Yamcs.
Instance	The instance this user is connected to
Processor	The processor this user is connected to

Note that our legacy standalone clients are not currently appearing in the list of connected clients. These clients connect using our previous API, and will be refactored towards the new API at some point.

4.8. Data Links

This view provides an overview of the data links of a running Yamcs server.

Data links represent input or output flows to Yamcs. There are three types of Data Links: TM (called TM Providers), TC (called TC Uplinkers) and PP (called PP Providers). TM and PP receive telemetry packets or parameters and inject them into the realtime or dump TM or PP streams. The TC data links subscribe to the realtime TC stream and send data to external systems.

There are different types of providers. For details refer to the [Yamcs Server Manual](#).

Data Links					
Name	Type	Spec	Stream	Status	Data Count
tm_realtime	TcpTmProvider	local	tm_realtime	OK	538,795
tm_dump	TcpTmProvider	localDump	tm_dump	OK	0
pp1	SimulationPpProvider	{simulationDataPath=...	pp_realtime	OK	160
tc1	TcpTcUplinker	local	tc_realtime	OK	0

The presented information includes:

Name	Identifier of this link as assigned by Yamcs
Type	The type of this link. For example, TcpTmProvider represents an input of TM over TCP
Spec	Configuration information passed to the provider instance. Significance depends on the type of provider
Stream	<p>The internal stream where the data is either:</p> <ul style="list-style-type: none">• sourced from (in the case of uplinkers), or• published to (in the case of providers) <p>It is perfectly valid for different providers or uplinkers to use the same stream.</p>
Status	<p>The status of this link. One of:</p> <ul style="list-style-type: none">• OK – if the link is alive• DISABLED – if the link was disabled• UNAVAIL – if the link is enabled, but not available <p>The <code>Status</code> background lights bright green if the data count increased within the last 1500 milliseconds.</p>
Data Count	The number of data elements (e.g. packets) that this link published or received from its stream since Yamcs started running.

With sufficient privileges, you can enable or disable a link manually by right-clicking the applicable row and selecting **Enable Link** or **Disable Link**.

Chapter 5. Troubleshooting

5.1. Capturing Log Output

In case you need to debug an issue with a deployed Yamcs Studio client, it can be useful to capture the logging output. Instructions are specific to the platform.

Linux

Launch the `Yamcs Studio` executable from a terminal window while redirecting all output to a file named `log.txt`

```
./Yamcs\ Studio >log.txt 2>&1
```

Mac OS X

With Terminal navigate into the Yamcs Studio application bundle and launch `./Yamcs Studio` directly from there while redirecting all output to a file named `log.txt`. For example:

```
cd Yamcs\ Studio.app/Contents/MacOS  
./Yamcs\ Studio >log.txt 2>&1
```

Windows

With Command Prompt navigate into the location where you installed Yamcs Studio and launch `Yamcs Studio.exe` while redirecting all output to a file named `log.txt`. For example:

```
"Yamcs Studio.exe" >log.txt 2>&1
```