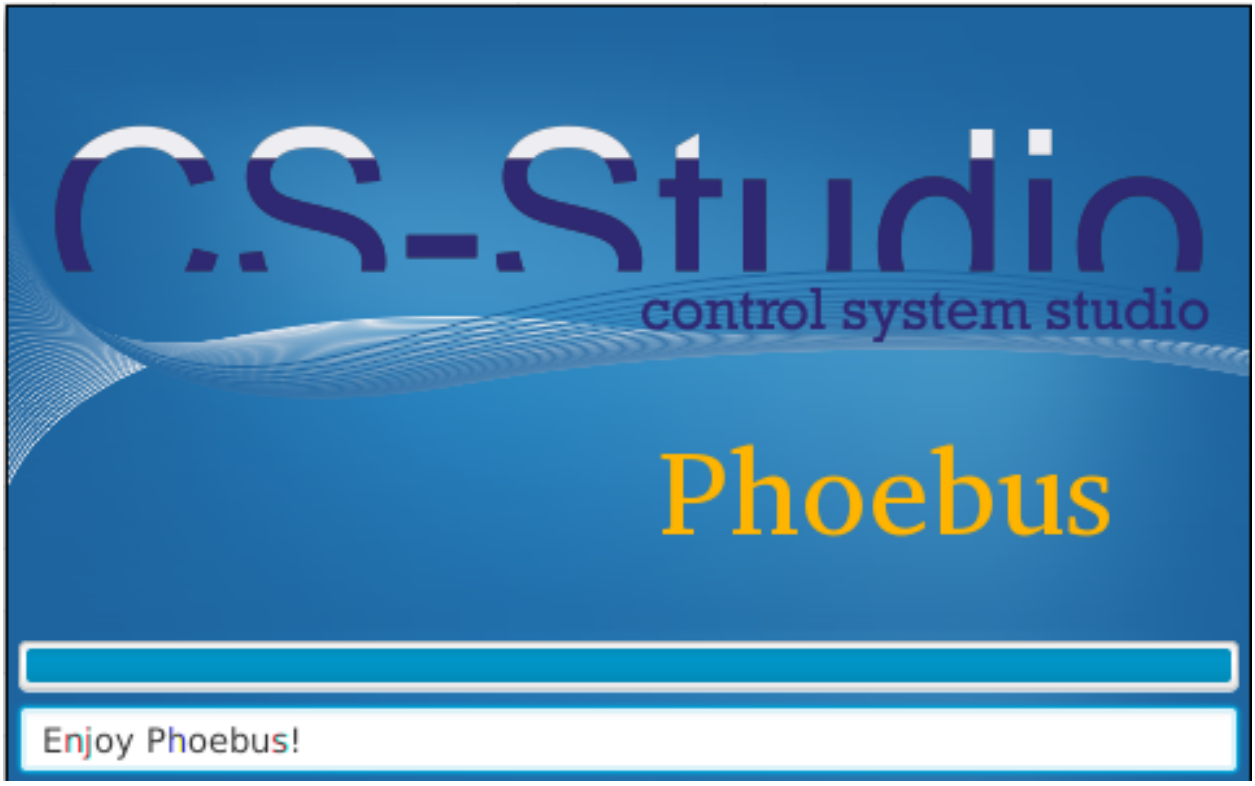
Phoebus Documentation

Release 1.0

Phoebus Developers

Dec 03, 2018

Contents

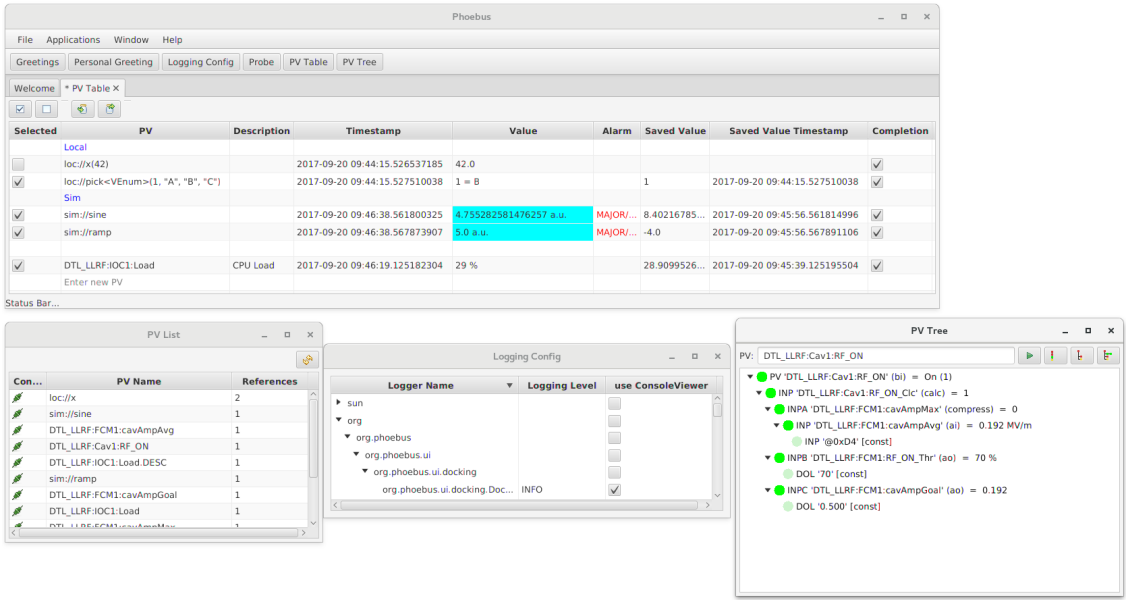


User Documentation:

CHAPTER 1

Introduction

Phoebus is an update of the Control System Studio toolset that removes dependencies on Eclipse RCP and SWT.



While Eclipse RCP has served CS-Studio well for about a decade, we are beginning to experience its limitations for control system user interface development.

Key Goals of the Phoebus project:

- Retain functionality of key CS-Studio tools, specifically the Display Builder, Data Browser, PV Table, PV Tree, Alarm UI, Scan UI, .. supporting their original configuration files with 100% compatibility.
- Provide full control of window placement free from RCP restrictions.
- Use Java FX as the graphics library to overcome limitations of SWT.
- Prefer core Java functionality over external libraries whenever possible: Java FX as already mentioned, Java 9 modules for bundling, SPI for locating extensions, java.util for logging and preferences, . . .

- Reduce build system complexity, fetching external dependencies in one initial step, then supporting a fully standalone, reproducible build process.

For more, see https://docs.google.com/document/d/11W52PRlsRjpIvP81HxUxxR9g180DHDByCohYQ9TQv7U

CHAPTER 2

Starting Phoebus

For build instructions, refer to the README.md on https://github.com/shroffk/phoebus

For pre-built binaries, see https://controlssoftware.sns.ornl.gov/css_phoebus/

From the command-line, invoke `phoebus.sh -help`:

```
 _________                _________  _________  _______                 _________
(  _____ )|\     /|(  ____  )(  _____ \(  ____ \ |\     /|(  _____ \
| (     )|| )   ( || (    )  || (     \/| (    ) )| )   ( || (     \/
| (_____)|| (___) || |    | || (__     | (__/ / | |   | || (______
|  ______)|  ___  || |    | ||  __)    |  __ (  | |   | |(______  )
| (       | (   ) || |    | || (       | (  \ \ | |   | |       ) |
| )       | )   ( || (____) || (_____/\| )___) )| (___) |/\_____) |
|/        |/     \|(________)(________/|/ \___/ (_________)\_________)

Command-line arguments:

-help                              -  This text
-settings settings.xml             -  Import settings from file, either exported XML or␣
↪property file format
-export_settings settings.xml      -  Export settings to file
-list                              -  List available application features
-app probe                         -  Launch an application with input arguments
-resource  /tmp/example.plt        -  Open an application configuration file with the␣
↪default application
-server port                       -  Create instance server on given TCP port
```

2.1 Command Line Parameters for Applications

To open an application feature like “probe” or the “pv_tree” from the command line, use the following example parameters.

Open empty instance of probe:

```
phoebus.sh -app probe
```

Open empty PV Table:

```
phoebus.sh -app pv_table
```

Open a file with the appropriate application feature (PV Table in this case):

```
phoebus.sh -resource /path/to/example.pvs
```

The '-resource' parameter can be a URI for a file or web link:

```
phoebus.sh -resource http://my.site/path/to/example.pvs
```

Some resource types are supported by multiple applications. For example, a display file "my_display.bob" can be handled by both the "display_runtime" and the "display_editor" application. A preference setting "org.phoebus.ui/default_apps" defines which application will be used by default, and a specific application can be requested like this:

```
phoebus.sh -resource /path/to/my_display.bob?app=display_editor
```

The schema 'pv://?PV1&PV2&PV3' is used to pass PV names, and the 'app=..' query parameter picks a specific app for opening the resource.

Open probe with a PV name:

```
phoebus.sh -resource pv://?sim://sine&app=probe
```

Open PV Table with some PVs:

```
phoebus.sh -resource pv://?MyPV&AnotherPV&YetAnotherPV&app=pv_table
```

Note that all these examples use the internal name of the application feature, for example "pv_table", and not the name that is displayed the user interface, like "PV Table". Use the `-list` option to see the names of all available application features.

2.2 Server Mode

By default, each invocation of `phoebus.sh ...` will start a new instance, with its own main window etc.

In a control room environment it is often advantageous to run only one instance on a given computer. For this scenario, invoke `phoebus.sh` with the `-server` option, using a TCP port that you reserve for this use on that computer, for example:

```
phoebus.sh -server 4918
```

The first time you start phoebus this way, it will actually open the main window. Follow-up invocations, for example:

```
phoebus.sh -server 4918 -resource /path/to/some/file.pvs
```

will contact the already running instance and have it open the requested file.

CHAPTER 3

Window Environment

When you start Phoebus for the first time, it opens a main window. As you use the *Applications* menu to open for example Probe, the PV Tree etc., these all open up as new Tabs in the original window.

3.1 Tabs and Windows

The behavior of these tabs is very similar to the handling of tabs in a web browser. You can rearrange the tabs within a window by dragging them around. You can also drag a tab out of the window to detach it into its own window. Alternatively, you can invoke the *Detach* option from the context menu of the tab to detach it. Tabs can be dragged between the main window and such detached windows.

3.2 Split Panes within a Window

The tab context menu options to *Split Horizontally* or *Split Vertically* will create sub-panels witin the window between which tabs can be arranged.

To un-do a split, simply move all tabs out of a split section. When a split section is empty, it will be merged back with its sibling (unless the pane is named, see below).

3.3 Hide Tabs

By default, even a window with just one tab will show that tab. This has the advantage that you can then grab that tab to move it into another window, or arrange tabs in the split subsections of a window. At times, however, you may prefer to hide such singular tabs to preserve screen space. In the *Window* menu, select *Always show Tabs* to show respectively hide singular tabs.

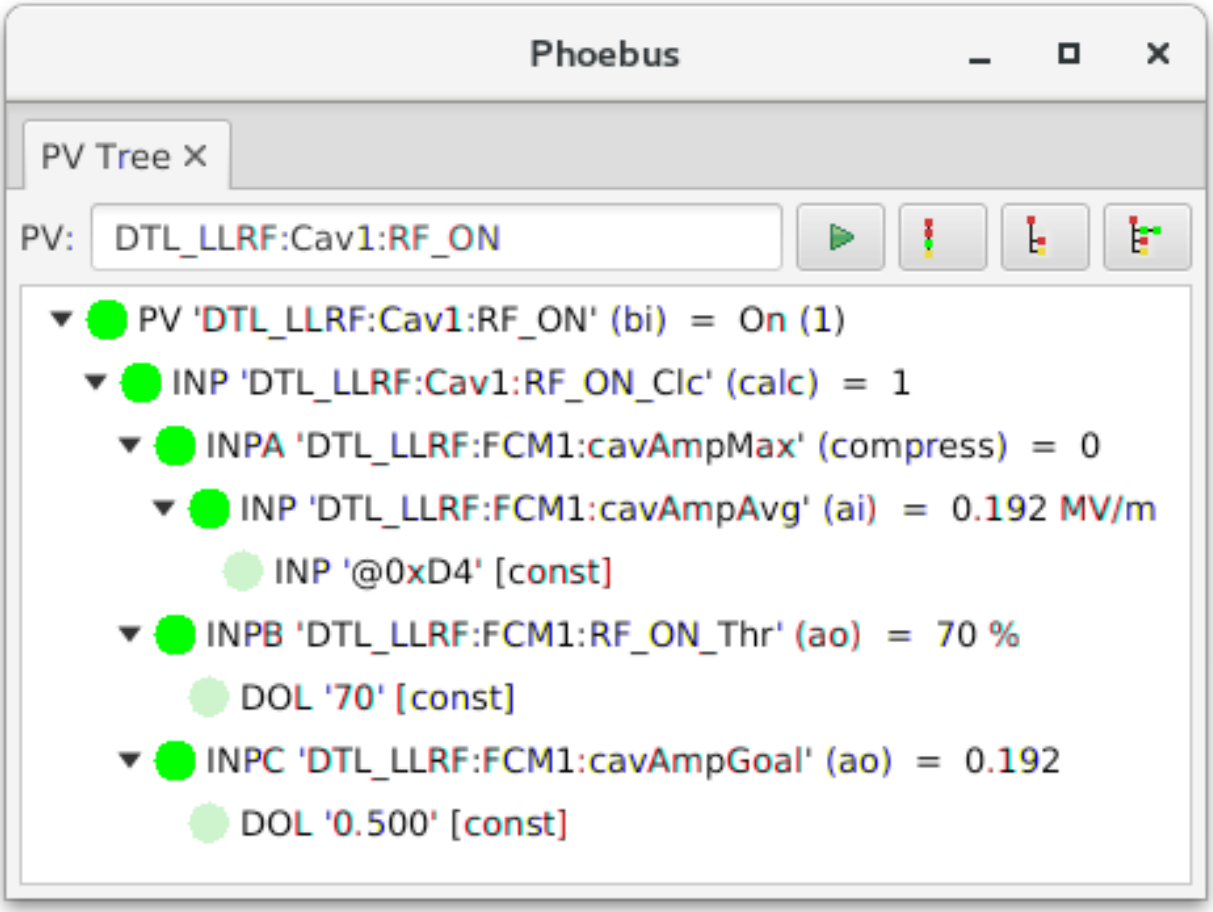


Fig. 1: Window with just one tab, showing that tab so you can drag it.

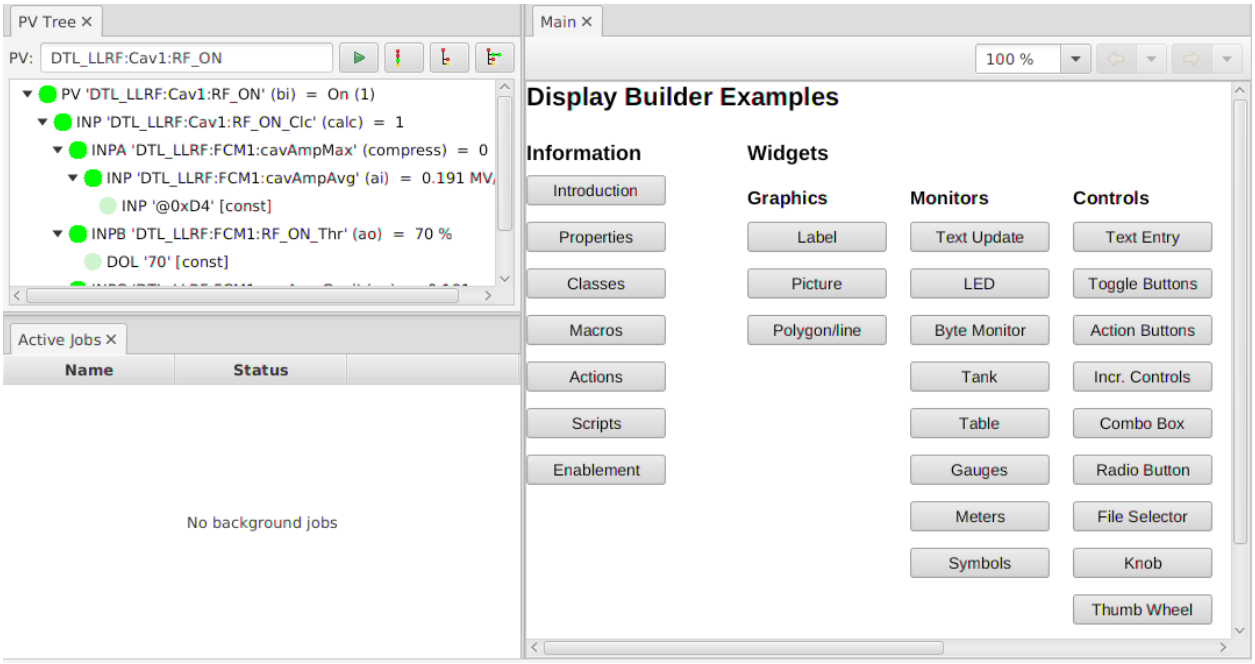


Fig. 2: In this example, the original PV Tree tab has been split *horizontally*, creating a left and right section. The left section contains the original PV Tree. A new Display Builder panel has been placed into the originally empty right section. Next, the PV Tree tab on the left has once more been split, this time *vertically*, and a new Jobs viewer has been placed in the newly created bottom half.

3.4 Saving & Restoring the Window Layout

The current window layout is saved to a `memento` file when exiting the program. This `memento` file is by default located within a `.phoebus` subdirectory of the user's home directory. To change the location from `$HOME/.phoebus` to a custom location, set the Java System property `phoebus.user` to the desired location.

When later starting the program back up, it will load the saved window layout.

By making the `memento` file read-only, system administrators can prevent the program from updating the file on exit. Each time the program is started, it will thus start out with a known window layout.

The application *Window* menu option *Save Layout As..* allows saving the current window layout under a name. The *Window* menu option *Load Layout* offers a list of all saved layout. Selecting a saved layout switches from the current display layout to a saved one.

Saved, named layouts are stored in files similar to the default `memento`, but including the name of the saved layout. These saved layout memento files can be deleted is no longer needed, copied to different installations, or made read-only to prevent replacement by end users.

3.5 Locking the Window Layout

The context menu of a tab within a pane allows locking and un-locking a pane.

A locked pane keeps its current set of tabs. Tabs cannot be moved out of a locked pane, they cannot be closed, nor can new tabs be added to a locked pane. A locked pane cannot be split.

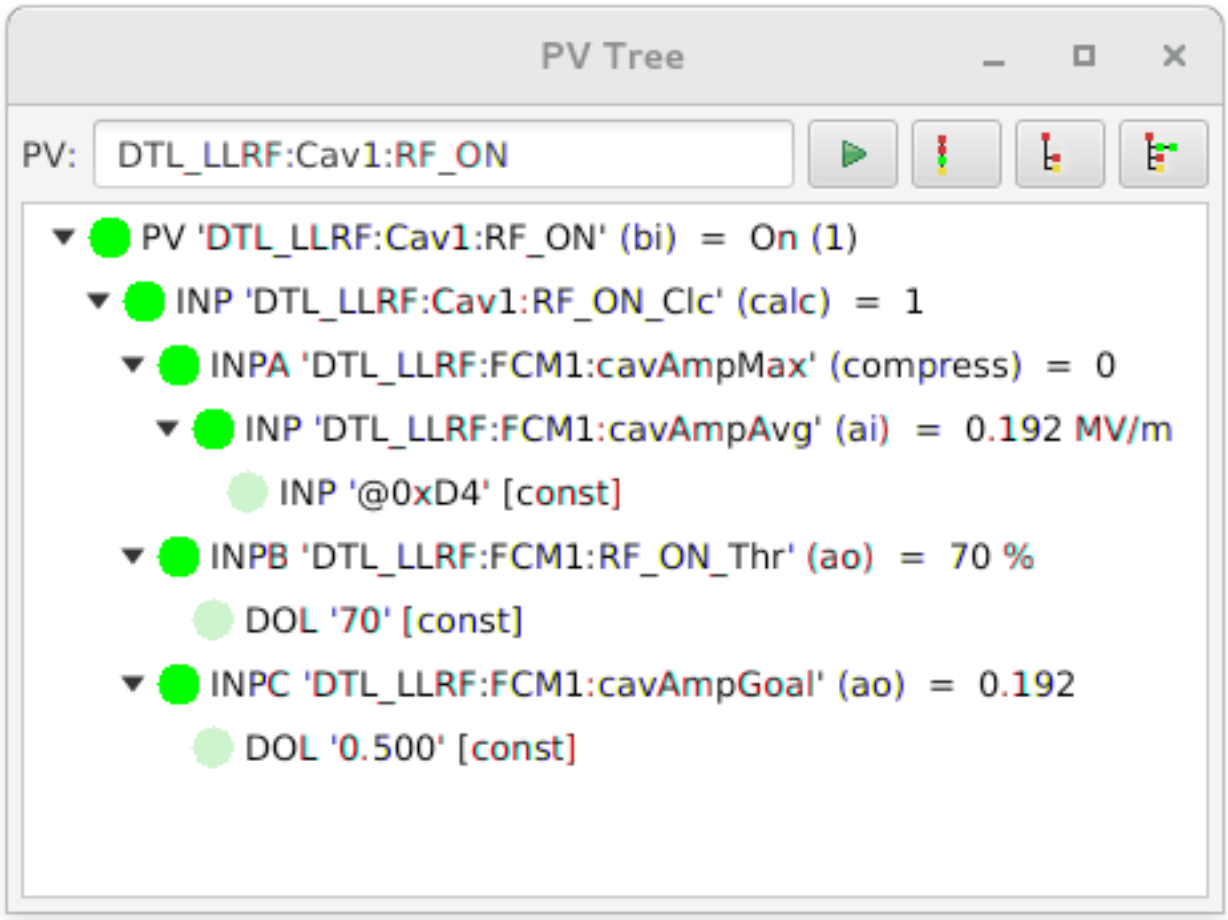


Fig. 3: Window with just one tab, hiding the actual tab to offer more screen space.

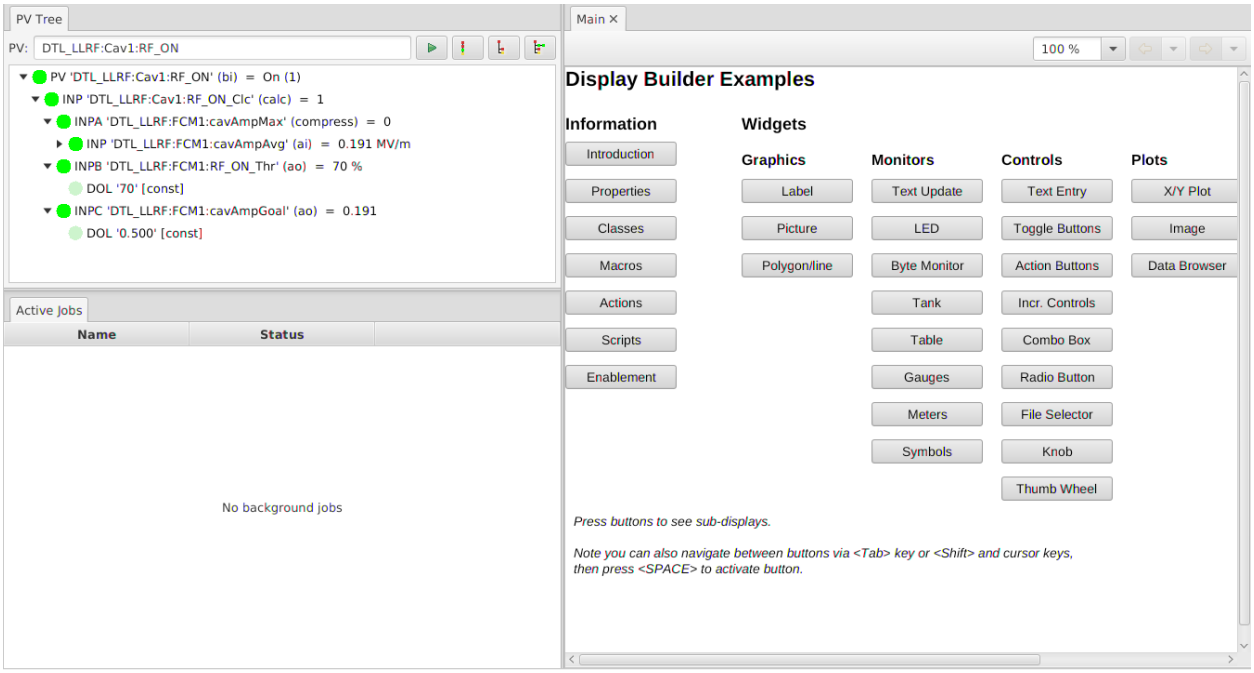


Fig. 4: In this example, the upper and lower panes on the left are locked. Note that the tabs for the PV Tree and Active Jobs have no x to close them. These tabs cannot be closed, they cannot be moved to other window sections, and you can no longer add new tabs into these *locked* panes.

Locked panes allows you to create a default layout that contains certain fixed panes which the user cannot accidentally delete at runtime.

3.6 Named Panes

The context menu of a tab allows naming the pane which contains the tab.

Display Builder builder panels can be configured to open new tabs in a specific, named pane. If that pane does not exist, it will be created, but ideally such displays are used within a layout that already contains the appropriately named panes.

A named pane will remain visibly even when empty. It will not be merged with sibling tabs.

CHAPTER 4

Preference Settings

When you run Phoebus, you may find that it cannot connect to your control system because for example the EPICS Channel Access address list is not configured.

To locate available preferences, refer to the complete list of *Preference Settings* or check the source code for files named `*preferences.properties`, for example in the `core-pv` sources:

```
# ----------------------------------------
# Package org.phoebus.applications.pvtable
# ----------------------------------------

# Show a "Description" column that reads xxx.DESC?
show_description=true

# -------------------------
# Package org.phoebus.pv.ca
# -------------------------

# Channel Access address list
addr_list=
```

Create a file `settings.ini` that lists the settings you want to change:

```
# Format:
#
#  package_name/setting=value
org.phoebus.pv.ca/addr_list=127.0.0.1 my_ca_gateway.site.org:5066
```

Start Phoebus like this to import the settings from your file:

```
phoebus.sh -settings /path/to/settings.ini
```

Conceptually, preference settings are meant to hold critical configuration parameters like the control system network configuration. They are configured by system administrators, and once they are properly adjusted for your site, there is usually no need to change them.

Most important, these are not settings that an end user would need to see and frequently adjust during ordinary use of the application. For such runtime settings, each applicaition needs to offer user interface options like context menus or configuration dialogs.

When you package phoebus for distribution at your site, you can also place a file `settings.ini` in the installation location (see *Locations*). At startup, Phoebus will automatically load the file `settings.ini` from the installation location, eliminating the need for your users to add the `-settings ..` command line option.

4.1 Developer Notes

In your code, create a file with a name that ends in `*preferences.properties`. In that file, list the available settings, with explanatory comments:

```
# ---------------------------------------
# Package org.phoebus.applications.my_app
# ---------------------------------------

# Note that the above
#
#    "# Package name.of.your.package"
#
# is important. It is used to format the online help,
# and users will need to know the package name to
# assemble their settings file.

# Explain what each setting means,
# what values are allowed etc.
my_setting=SomeValue

# Enable some feature, allowed values are true or false
my_other_setting=true
```

Load that as the default, then read the `java.util.prefs.Preferences` like this:

```
package org.phoebus.applications.my_app

import org.phoebus.framework.preferences.PreferencesReader;

# The class that you pass here determines the package name for your preferences
final PreferencesReader prefs = new PreferencesReader(getClass(), "/my_app_
↪preferences.properties");

String pref1 = prefs.get("my_setting");
Boolean pref2 = prefs.getBoolean("my_other_setting");
// .. use getInt, getDouble as needed
```

The `PreferencesReader` loads defaults from the property file, then allows overrides via the `java.util.prefs.Preferences` API. By default, the user settings are stored in a `.phoebus` folder in the home directory. This location can be changed by setting the Java property `phoebus.user`.

In the future, a preference UI might be added, but as mentioned the preference settings are not meant to be adjusted by end users.

CHAPTER 5

Runtime Settings

When you run Phoebus, you may open a tab with the "PV Tree", enter a PV name, move the window around etc.

When you exit Phoebus, the current location of windows, tabs, and for example the PV name of an active "PV Tree" are stored.

When you later restart Phoebus, it restores the windows with their saved location and content.

5.1 Developer Notes

The state is persisted in a `memento` file, located in the same directory as the user preferences. To change the default from `.phoebus/memento` in your home directory to a different location, see Preference Settings *Developer Notes*.

To always start Phoebus with a known window layout, save the `memento` from a desired layout, and then place that saved file in the user preferences directory before starting a new instance of Phoebus.

CHAPTER 6

Authorization

6.1 Authentication vs Authorization

Phoebus depends on the operating system to authenticate the user. The currenly logged in user is who we expect to be interating with Phoebus.

Phoebus does add basic authorization to control if the current user may configure details of the window layout (lock and unlock panes) or alarm system (add, remove, acknowledge alarms).

6.2 Configuring Authorization

A preference setting selects a more detailed autorization configuration file:

```
org.phoebus.ui/authorization_file=/path/to/authorization.conf
```

See details of the `org.phoebus.ui` preferences for the possible locations of that file.

Example autorization configuration file:

```
# Authorization Settings
#
# Format:
# authorization = Comma-separated list of user names
#
# The authorization name describes the authorization.
# FULL is a special name to obtain all authorizations.
#
# Each entry in the list of user names is a regular expression for a user name.

# Anybody can lock a dock pane, i.e. set it to 'fixed'
lock_ui = .*
```

(continues on next page)

(continued from previous page)

```
# Anybody can acknowledge alarms
alarm_ack = .*

# Specific users may configure alarms, including both "jane" and "janet"
#alarm_config = fred, jane.*, egon,

# Anybody can configure alarms
alarm_config = .*

# Full authorization.
FULL = root
```

CHAPTER 7

Applications

The following sections describe details of specific application features.

7.1 Active Jobs

Opened from the `Applications`, `Debug` menu, the Active Jobs panel displays ongoing background jobs.

Examples include jobs that load configuration files. Long-running jobs can be cancelled.

7.1.1 Implementation Detail

Canceling a job asks the application to abort an ongoing job and depends on a proper implementation of the application feature to honor the request.

7.2 PV List

Opened from the `Applications`, `Debug` menu, the PV List panel lists all active PVs, their connection state and the reference count.

7.3 Process Variables

Several types of process variables are supported. A prefix of the form "xx://.." is typically used to select the PV type.

7.3.1 Channel Access

Process variables that are to be accessed over the channel access protocol are simply identified by the channel name.

Channel access is the default protocol. If desired, 'ca://' can be used to specifically select channel access, but for the time being no protocol identification is necessary for channel access.

Examples:

```
SomePVName
ca://SomePVName
SomeMotor.RBV
```

7.3.2 PV Access

Process variables that are to be accessed over the PV Access protocol must be identified by a formatted string that contains the process variable's name.

As PV Access is not the default protocol, process variables accessed over it must have the protocol specified with 'pva://'.

The PV Access format is as follows:

```
pva://SomePVName
pva://SomePVName/subfield/subelement
```

As shown, when accessing structures, the path to a nested structure element can be provided.

7.3.3 Simulated

Simulated process variables are ueful for tests. They do not communicate with the control system.

The provided simulated process variables are:

- flipflop(update_seconds)
- gaussianNoise(center, std_dev, update_seconds)
- gaussianWave(period, std_dev, size, update_seconds)
- intermittent(update_seconds)
- noise(min, max, update_seconds)
- noisewave(min, max, update_seconds)
- ramp(min, max, update_seconds)
- sawtooth(period_seconds, wavelength, size, update_seconds)
- sine(min, max, update_seconds)
- sinewave(period_seconds, wavelength, size, update_seconds)
- strings(update_seconds)

Examples:

```
sim://sine
sim://ramp
sim://ramp(1, 10, 0.2)
sim://noise
```

7.3.4 Local

Local process variables can be used within the application, for example to send a value from one display to another display within the same application. They do not communicate with the control system.

Unless a type selector and initial value are provided, a local value will be of type 'double' with initial value of 0.

Examples:

```
loc://example
loc://a_number(42.2)
loc://an_array(3.8, 2.5, 7.4)
loc://a_text("Hello")
loc://large<VLong>(42)
loc://options<VEnum>(2, "A", "Initial", "B", "C")
```

7.3.5 MQTT

Data that is to be read over the MQTT network protocol must be referenced with a formatted string which contains the name of the MQTT topic and the VType that corresponds to the type of data published on the topic.

All MQTT topics are obtained from the same MQTT broker URL, based on a preference setting that defaults to:

```
org.phoebus.pv.mqtt/mqtt_broker=tcp://localhost:1883
```

If the VType is omitted, 'double' is assumed. Examples:

```
mqtt://some_topic
mqtt://some_topic<VDouble>
mqtt://some_topic<VString>
mqtt://some/nested/topic
```

7.4 Performance Monitor

Invoking the menu Applications, Debug, Performance monitor adds a button to the status bar that displays performance information. Invoking it again will remove it.

7.4.1 Memory

Displays how much memory is allocated, and how much of that is available. For example, "Avail: 28% of 0.21GB".

Pressing the button triggers a garbage collection.

7.4.2 Frame Rate

Measures the JavaFX frame rate, nominally 60Hz.

System properties that can influence the result:

-Dquantum.multithreaded=true -Djavafx.animation.fullspeed=true -Djavafx.animation.framerate=120 -Djavafx.animation.pulse=120

7.5 Update

The 'update' application allows a product to self-update.

Assume you have a site-specific product, i.e. a ZIP file that users at your site can download. By including the 'update' application in your product and setting two preference settings, your product can self-update.

7.5.1 Configuration

The `current_version` setting needs to provide the current version of your product in the format `YYYY-MM-DD HH:MM`. The `update_url` setting needs to point to a file or web URL that contains the latest product.

Example:

```
org.phoebus.applications.update/current_version=2018-06-18 13:10 org.phoebus.applications.update/update_url=http://my.site.org
for-my-site.zip
```

7.5.2 Usage

On startup, the update mechanism checks the *update_url*. If that file is dated after the *current_version*, an "Update" button is added to the status bar to indicate that an update is available.

Clicking that "Update" button opens a dialog with details on the current version, the updated version, and the installation location that will be replaced in the update.

When pressing "OK", the update is downloaded and the current installation is replaced. Finally, a prompt indicates completion of the update, and the product exists for you to start the updated version.

7.6 PV Table

The PV Table provides a tabular view of PV names and their current value with time stamp and alarm state.

You can take a "snapshot" of current values and dates, and the table will now highlight rows where the current value differs from the snapshot.

The configuration (PV names, saved value, saved date) can be saved and later re-loaded, see details on the file format described below.

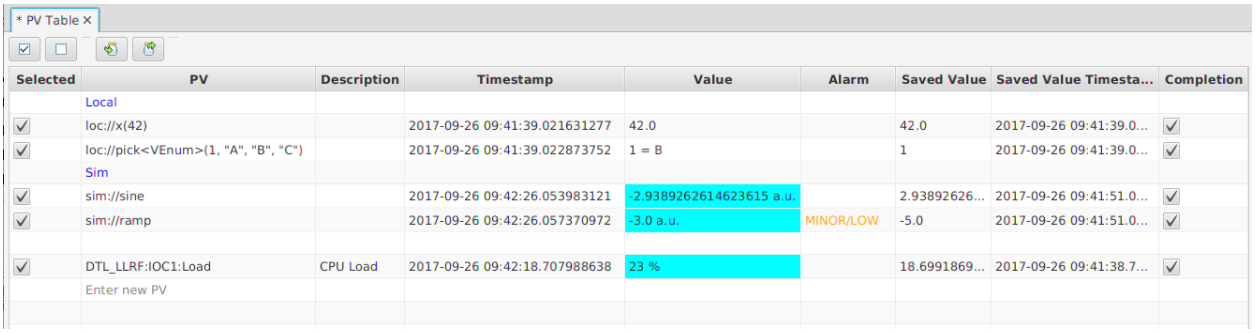


Fig. 1: PV Table

7.6.1 Adding and Removing PVs

The simplest way is to enter new PV names in last row of table.

To insert a new PV in the middle of the table, open the context menu on the desired table row and select "Insert Row (above)" to add a row above, then change the PV name of the new row.

Finally, you use drag-and-drop to move existing rows within the table, or to 'drop' PV names into the table from tools that support dragging PV names.

Delete PV names by changing their name to an empty name, or by selecting one or more PVs and deleting them via the context menu.

7.6.2 Comments

PV names that start with "# " are considered a comment. This can also be used to add empty lines into the table by entering just "#" as a PV name.

7.6.3 Checking PVs for Snapshot/Restore

By default, the check mark at the start of each table row is set. When taking a snapshot of current values or restoring PVs to the snapshot, this typically applies to rows where the check mark is set.

You can un-check table rows if they should be excluded. The context menu of the table offers shortcuts to select or de-select the whole table.

In addition, the context menu also allows taking a snapshot or restoring the row on which the context menu was invoked, which can be useful to operate on just one PV and not the whole table.

7.6.4 Restoring PVs

The value of PVs can be restored, i.e. the saved value will be written to the PV. By default, this affects every row of the table, but the check-mark at the start of each table row can be use to de-select rows.

7.6.5 Completion

By default, saved values are restored to PVs by simply writing to them. When checkbox in the "Completion" column is selected for an PV, the saved value will be restored by using the "Put-Callback" method of writing, awaiting the completion of the write. This can be useful with PVs that support put-callback, a typical example being motors.

The PV Table has one global timeout that is used for each write operation that uses completion. It defaults to 60 seconds and can be changed via the "Completion Timeout" option in the context menu.

7.6.6 Tolerance

Values are highlighted when they differ from the saved snapshot value by a certain amount. The currently used tolerance is displayed in the tool-tip of a table row. This 'tolerance' value can be configured via the context menu of selected table rows.

When configuring the 'tolerance', note that it applies to the rows which are selected in the table via the usual selection mechanism (click on one row, shift-click to select multiple rows, ...). If no row specific rows are selected to set their tolerance, the tolerance for every row in the table will be updated. This is independent of the check mark in the first table column which marks rows to be restored by writing their saved value back to the PVs.

7.6.7 File Formats

The original PVTable file format uses a ".pvs" extension for its file names. The files have an XML format which is described by the <file>pv_table.xsd</file> contained in the PV Table sources.

Since version 4.0.0, the PVTable also supports file format used by the EPICS synApps `autosave` module, http://www.aps.anl.gov/bcda/synApps/autosave/autosave.html. Whenever loading or saving a PVTable from a file with a ".sav" extension, the autosave format will be used.

Advantages of the original PVTable ".pvs" file format:

- Tracks which rows were selected.
- Saves not only the value but also the timestamp of saved values.
- Contains global as well as per-element 'tolerance'.
- Allows using 'completion'.
- Best for standalone operation of the PVTable.

Advantages of the autosave ".sav" file format:

- It can be used by the IOC to load/save settings.
- PVTable allows easy comparison of last settings written by IOC against current values.
- Best for use together with on-demand save/restore on the IOC.

7.7 Display Builder

7.7.1 Install Examples

Invoke `Install Example Display` from the menu `Applications`, `Display`, `Examples` to install the display builder examples.

You will be prompted for a location where you wish to install them.

Then use the menu `File`, `Open` to open the file `01_main.bob` from the examples to get started.

7.7.2 Create new Display

Use the menu option `Applications`, `Display`, `New Display` to create a new display.

7.7.3 Internals

7.8 Logging Configuration

Allows for the runtime configuration of loggers.

Application developers are encouraged to use `java.util.logging` for log messages.

To debug a problem, the logging configuration allows you to for example change the log level for the `org.phoebus.pv` module to use FINE or even ALL to see all received PV value updates.

7.9 3D Viewer

7.9.1 Overview

The 3d Viewer is a tool that allows users to configure 3 dimensional structures using spheres, cylinders, and boxes.

These structures are defined in shape file (**.shp*) and parsed by the application.

The resultant structure is then rendered on screen. This structure can be viewed in the application which allows rotation, zoom, and movement.

The individual spheres, cylinders, and boxes that make up the structure can have their coordinates, sizes, and colors specified.

7.9.2 Shape (.shp) File Syntax

The viewer parses shape files from beginning to end. Any error in the shape file will cause the parsing of the entire file to fail and no resulting structure will be rendered.

Comments Shape files can have comments. A comment is a line of text that starts with a '#'. This line will be ignored when parsing the shape file.

Background Color The background color of the viewer can be controlled using the following command.

```
background(r, g, b, A)
```

This command has four parameters. The red, green, and blue values for the color are the first three. Each color value is an integer from 0 through 255. These are followed by the alpha value which allows you to control the transparency of the color. Alpha is a floating point number in the range [0, 1]. An alpha of 0 is transparent while an alpha of 1 is opaque.

Multiple background colors may be defined, however only the last defined background color will be used.

Spheres A sphere may be defined using the following command.

```
sphere(x, y, z, R, r, g, b, A)
```

This command has eight parameters. The first three parameters are the x, y, and z values which represent the center point of the sphere in the three dimensional space. The x, y, and z parameters are floating point values. These are followed by the radius of the sphere, another floating point value. The final four parameters are the red, green, blue, and alpha values used to define the color of the sphere. Red, green, and blue are integer values in [0, 255] and alpha is a floating point value in [0, 1].

Cylinders A cylinder may be defined using the following command.

```
cylinder(x1, y1, z1, x2, y2, z2, R, r, g, b, A)
```

This command has eleven parameters. The first three parameters are the x, y, and z values which represent one end point of the cylinder. The second three parameters are the x, y, and z values which represent the other end point of the cylinder. All x, y, and z parameters are floating point values. These are followed the cylinder's radius. The radius is a floating point value. The final four parameters are the red, green, blue, and alpha values used to define the color of the sphere. Red, green, and blue are integer values in [0, 255] and alpha is a floating point value in [0, 1].

Boxes A box may be defined using the following command.

```
box(x1, y1, z1, x2, y2, z2, r, g, b, A)
```

This command has 10 parameters. The first three parameters are the x, y, and z values which represent one corner of the box.The second three parameters are the x, y, and z values which represent the opposite corner of the box. All x, y, and z parameters are floating point values. The final four parameters are the red, green, blue,

and alpha values used to define the color of the sphere. Red, green, and blue are integer values in [0, 255] and alpha is a floating point value in [0, 1].

If the first corner of a box was defined at (0, 0, 0) and the second corner at (100, 100, 100) then the box would have one corner at the origin and one corner at (100, 100, 100). Each of the boxes sides would be of length 100, and the boxes center point would be (50, 50, 50).

Cones A cone may be defined using the following command.

```
cone(x1, y1, z1, R, x2, y2, z2, r, g, b, A)
```

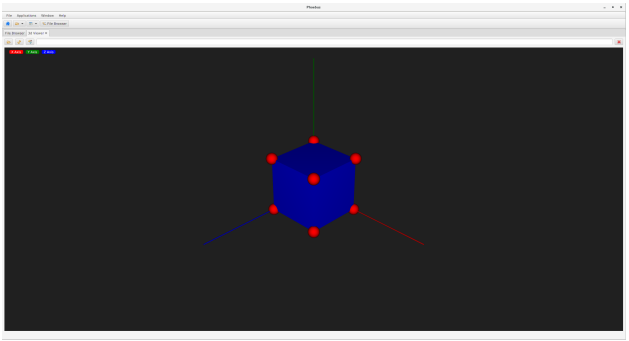
This command has eleven parameters. The first three parameters are the x, y, and z values of the base, followed by the radius of the base. The second three parameters are the x, y, and z values of the tip of the cone. The final four parameters are the red, green, blue, and alpha values used to define the color.

Tool Tips A final string added to a shape defines a tool tip for the shape.

7.9.3 Example Shape File

```
# This is a comment. It will be ignored when the file is parsed.

# The background of the viewer is set to be nearly black.
background(32, 32, 32, 1)

# A red sphere of radius 10, is placed at each corner of the box we are about to␣
↪define.
sphere(  0,   0,   0, 10, 255, 0, 0, 1, "Origin")
sphere(100,   0,   0, 10, 255, 0, 0, 1)
sphere(  0,   0, 100, 10, 255, 0, 0, 1)
sphere(100,   0, 100, 10, 255, 0, 0, 1)
sphere(  0, 100,   0, 10, 255, 0, 0, 1)
sphere(100, 100,   0, 10, 255, 0, 0, 1)
sphere(  0, 100, 100, 10, 255, 0, 0, 1)
sphere(100, 100, 100, 10, 255, 0, 0, 1)

# A blue box is defined with one corner at the origin, and the opposite
# corner at (100, 100, 100). This will result in a cube with each side
# being of magnitude 100.
box(0, 0, 0, 100, 100, 100, 0, 0, 255, 1)

# Cone along the X axis, base at x=200, radius 20, tip at x=300
cone ( 200,   0,   0, 10,   300,   0,   0,   255, 100, 100, 1, "X")
```

Resulting Structure



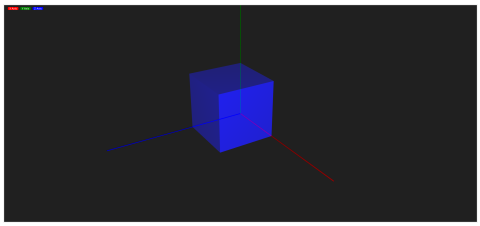
7.9.4 Transparency

JavaFX does not sort 3D objects by depth. What this means is that you have to be thoughtful of the order you add 3D shapes to a scene. For example, if a sphere needed to be displayed inside a translucent box, the sphere would have to be added *before* the box. If the box first were added first, it would still be translucent, but the JavaFX renderer would not draw the sphere because it doesn't sort the scene graph by depth.

Examples

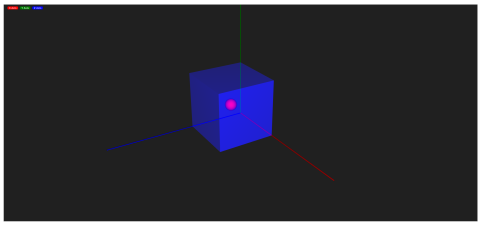
Here, the box is added first and the sphere is not drawn.

```
background(32, 32, 32, 1)
box(0, 0, 0, 100, 100, 100, 0, 0, 255, 0.1)
sphere(50, 50, 50, 10, 255, 0, 0, 1)
```



Here, the box is added second and the sphere is drawn correctly.

```
background(32, 32, 32, 1)
sphere(50, 50, 50, 10, 255, 0, 0, 1)
box(0, 0, 0, 100, 100, 100, 0, 0, 255, 0.1)
```



7.9.5 Compatibility

The 3D Viewer requires support from the graphics system.

Known to work:

- OpenJDK 11 on Mac OS 10.13.6
- OpenJDK 11 on Windows 10
- Oracle JDK 9, Oracle JDK 10, and OpenJDK 11 on RHEL 7.6

Running on Linux requires direct graphics on a local machine.

Can be made to work:

- Oracle JDK 10 or OpenJDK 11 on Centros 7.5 running inside VirtualBox, hosted on RHEL 7.6, when setting `-Dprism.forceGPU=true`

In case of problems which usually include error messages `System can't support ConditionalFeature.SCENE3D`, start the program with `-Dprism.verbose=true` and `-Djdk.gtk.verbose=true`

7.10 PV Tree

The PV Tree displays the hierarchical data flow between EPICS records. It displays the record types and their current values as well as severity/status. It attempts to reflect the data flow by traversing input links (INP, INPA, DOL, . . .).

The PV Tree has two modes:

"Run" : In this mode it will always display the current value of each item in the tree.

"Freeze on Alarm" : In this mode, updates pause as soon as the root item of the PV Tree goes into alarm.

7.10.1 Usage

Enter a name into the "PV" text box, and see what happens.

7.10.2 Tool Bar Buttons

, Changes the PV Tree mode between "running" and "freeze on alarm".

Collapse the tree, i.e. close all sub-sections of the tree.

Display all items in the tree that are in an alarm state. Note that this is performed whenever you push the tool bar button. If the PV tree items update, branches will not automatically show or hide based on their alarm state, because this could result in a very nervous display for a rapidly changing PV tree. Whenever you desire to update the tree to show/hide items, push the button.

Expand all sub-sections of the tree.

7.10.3 Limitations

This tool uses the EPICS network protocol, Channel Access, to read PVs. There is no way to query EPICS V3 IOCs for their database information to determine the available "input" links.

The knowledge of which links to follow for each record type is therefore configured into the EPICS PV Tree application. It is at this time not configurable by end users, but people with access to the source code can determine the syntax from the Settings.java file and override the site-specific settings.

7.11 Channel Applications

7.11.1 Overview

The Channel Viewer is a CS-Studio application which can query the directory service for a list of channels that match certain conditions, such as physical functionality or location. It also provides mechanisms to create channel name aliases, allowing for different perspectives of the same set of channel names.

7.11.2 Launching

From within cs-studio `Applications --> Channel --> Channel Table/Channel Tree`

From command line

`-resource cf://?query=SR*&app=channel_tree` `-resource cf://?query=SR*&app=channel_table`

7.11.3 Channel Table

Displays the results of a channelfinder query as a table

The query can be based on the channel name or based on a group of tag and properties associated with the channel

Wildcard character like "*", "?" can be used in the queries

7.11.4 Channel Tree

Channel Tree by Property allows to create an hierarchical view of the channels by using properties and their values. It groups the channels returned by a query based on the value of the properties selected.

7.12 Alarms

7.12.1 Overview

The alarm system consists of an alarm server and a user interface.

The Alarm Server monitors a set of PVs, tracking their alarm state.

The user interface shows the current alarms, allows acknowledgement, and provides guidance, links to related displays.

Fundamentally, the alarm server detects when a PV enters an alarm state. Even if the PV should then leave the alarm state, the alarm server remembers when the PV first entered the alarm state until the user acknowledges the alarm.

7.12.2 Kafka

Kafka stores the alarm system configuration, and provides the communication bus between the alarm server and user interface.

Refer to *applications/alarm/Readme.md* for setting up Kafka.

7.12.3 Alarm Server

Run with `-help` to see command line options.

7.12.4 User Interface

Alarm Tree: Allows configuration.

Alarm Table: Main runtime interface, shows current alarms.

Annunciator: Annunciates alarms.

Each of the above alarm apps can be launched from the cmd line as follows

```
-resource alarm://localhost/Accelerator?app=alarm_tree
```

```
-resource alarm://localhost/Accelerator?app=alarm_table
```

```
-resource alarm://localhost/Accelerator?app=alarm_area
```

7.12.5 Guidance

7.12.6 Displays

7.12.7 Commands

7.12.8 Automated Actions

`mailto:user@site.org,another@else.com`: Sends email with alarm detail to list of recipients.

The email server is configured in the alarm preferences.

`cmd:some_command arg1 arg2:` Invokes command with list of space-separated arguments. The special argument "*" will be replaced with a list of alarm PVs and their alarm severity. The command is executed in the `command_directory` provided in the alarm preferences.

CHAPTER 8

Services

The following sections describe available services.

8.1 RDB Archive Engine Service

The RDB archive engine reads samples from PVs and writes them to an RDB. The RDB may be MySQL, Posgres or Oracle. For a production setup, the latter two offer a partitioned table space that allows managing the data by time. For smaller setups and to get started, MySQL is very straight forward.

Once the RDB is configured with the archive table schema, the archive engine is used both as a command line tool to configure the archive settings and as a service to write samples from PVs to the RDB. You can build the archive engine from sources or fetch a binary from https://controlssoftware.sns.ornl.gov/css_phoebus

8.1.1 Install MySQL (Centos Example)

Install:

```
sudo yum install mariadb-server
```

Start:

```
sudo systemctl start mariadb
```

Set root password, which is initially empty:

```
/usr/bin/mysql_secure_installation
```

In the following we assume you set the root password to `$root`. To start RDB when computer boots:

```
sudo systemctl enable mariadb.service
```

8.1.2 Create archive tables

Connect to mysql as root:

```
mysql -u root -p'$root'
```

and then paste the commands shown in `services/archive-engine/dbd/MySQL.dbd` (available online as https://github.com/shroffk/phoebus/blob/master/services/archive-engine/dbd/MySQL.dbd) to create the table setup for archiving PV samples.

8.1.3 View Archive Data

The default settings for the Phoebus Data Browser check for archived data in `mysql://localhost/archive`. To access MySQL on another host, change these settings in your *Preference Settings*

```
org.csstudio.trends.databrowser3/urls=jdbc:mysql://my.host.site.org/archive|RDB
org.csstudio.trends.databrowser3/archives=jdbc:mysql://my.host.site.org/archive|RDB
```

The `MySQL.dbd` used to install the archive tables adds a few demo samples for `sim://sine(0, 10, 50, 0.1)` around 2004-01-10 13:01, so you can simply add that channel to a Data Browser and find data at that time.

8.1.4 List, Export and Import Configurations

List configurations:

```
archive-engine.sh -list
Archive Engine Configurations:
ID  Name     Description        URL
 1  Demo     Demo Engine        http://localhost:4812
```

Extract configuration into an XML file:

```
archive-engine.sh -engine Demo -export Demo.xml
```

Modify the XML file or create a new one to list the channels you want to archive and to configure how they should be samples. For details on the 'scanned' and 'monitored' sample modes, refer to the CS-Studio manual chapter http://cs-studio.sourceforge.net/docbook/ch11.html

Finally, import the XML configuration into the RDB, in this example replacing the original one:

```
archive-engine.sh -engine Demo -import Demo.xml -port 4812 -replace_engine
```

8.1.5 Run the Archive Engine

To start the archive engine for a configuration:

```
archive-engine.sh -engine Demo -port 4812 -settings my_settings.ini
```

The engine name ('Demo') needs to match a previously imported configuration name, and the port number (4812) needs to match the port number used when importing the configuration. The settings (my_settings.ini) typically contain the EPICS CA address list settings as well as archive engine configuration details, see archive engine settings in *Preference Settings*.

In a production setup, the archive engine is best run under `procServ` (https://github.com/ralphlange/procServ).

The running archive engine offers a simple shell:

```
INFO Archive Configuration 'Demo'
...
INFO Web Server : http://localhost:4812
...
>
> help
Archive Engine Commands:
help            -  Show commands
disconnected    -  Show disconnected channels
restart         -  Restart archive engine
shutdown        -  Stop the archive engine
```

In addition, it has a web interface accessible under the URL shown at startup for inspecting connection state, last archived value for each channel and more. The engine can be shut down via either the `shutdown` command entered on the shell, or by accessing the `stop` URL. For the URL shown in the startup above that would be `http://localhost:4812/stop`.

Developer Documentation:

CHAPTER 9

Developer Information

To get started with developing for phoebus, refer to the instructions on https://github.com/shroffk/phoebus .

CHAPTER 10

Architecture

The fundamental phoebus architecture consists of **core** modules, user-iterface related **core-ui** modules, and **app** modules. The core modules provide the infrastructure, while app modules implement specific application functionality. Everything is based on Java version 9 or higher, using Java FX as a graphics library.

A Phoebus product may contain just one application, for example only one of Probe, PV Tree, PV Table, Display Builder Runtime, so you end up with several Phoebus products that each perform one function. Alternatively, you can assemble a Phoebus product that contains all these applications. This allows integration between the applications, for example via context menues that start other PV-related applications based on the current selection.

10.1 Core Modules

core-framework: Fundamentals that many applications use, for example prerences, persistence, jobs, macros, localization, autocompletion.

Defines the `AppDescriptor` and `AppResourceDescriptor` Java Service Provider Interfaces (SPI) which are used to locate applications. Each application feature identifies itself by implementing an application descriptior that describes to the Phoebus framework what the name of the application is, which types of resources (e.g. data files) it might accept, and most importantly how to start one or more instances of the application.

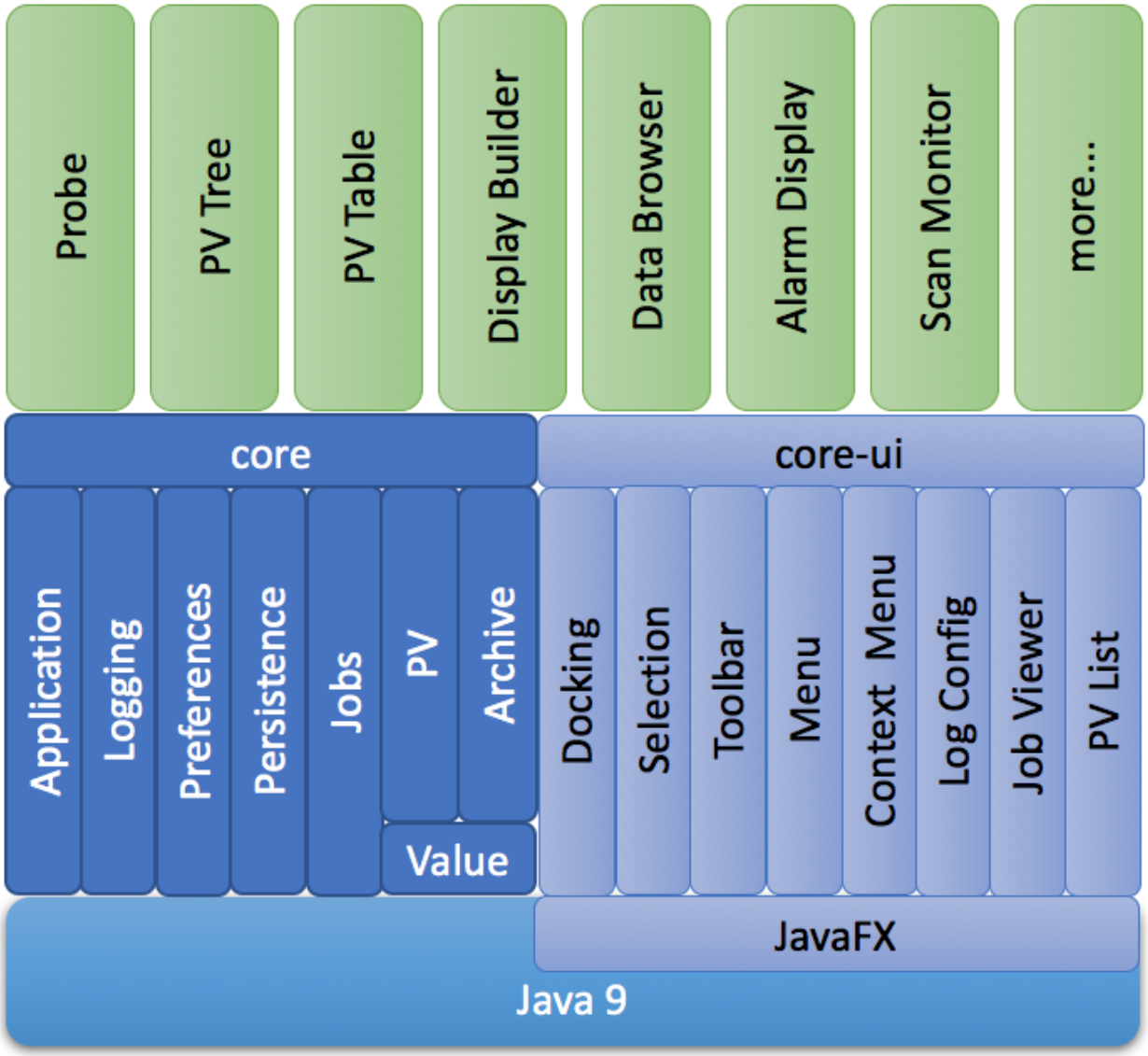
To create an `AppInstance`, i.e. an application instance, the framework invokes the `create()` method of the application descriptor. This will typically result in a new application instance, i.e. a new tab in the UI. Certain applications like the job viewer will create a singleton application instance.

On shutdown, the state of all windows and tabs is persisted in a memento file, and each `AppDescriptor` can also persist its own state. On startup, each window and tab is restored, the applications are restarted, and each application can restore its specific state from the memento.

The `JobManager` API allows submitting jobs based on a `JobRunnable` that supports progress reporting and cancellation.

core-pv: API for access to life data from Process Variables.

core-logbook: API for accessing a logbook, with SPI for site-specific implementations.


Probe
PV Tree
PV Table
Display Builder
Data Browser
Alarm Display
Scan Monitor
more...
core
core-ui
Application
Logging
Preferences
Persistence
Jobs
PV
Archive
Value
Docking
Selection
Toolbar
Menu
Context Menu
Log Config
Job Viewer
PV List
JavaFX
Java 9

core-email: API for creating emails.

core-security: API for authorization and secure storage.

core-ui: The `docking` package supports a window environment similar to a web browser. Each window can have multiple tabs. Users can move tabs between existing windows, or detach them into newly created windows.

The top-level Java FX `Node` for each application's UI scene graph is basically a `Tab`, wrapped in a Phoebus `DockItem` that tracks the `AppInstance` to allow it to be saved and restored.

The toolbar, main menu and context menues accept SPI-based contributions.

The `selection` package allows publishing and monitoring a selection of for example PVs.

The `undo` package simplifies the implementation of undo/redo functionality.

10.2 App Modules

Each app module implements a certain application feature, for example Probe, PV Tree, Display Builder. While application modules depend on one or more core modules, there should be few if no inter-application dependencies, allowing a product to contain only the desired applications.

10.3 Products

Each core and application module is built into a `jar` file. A product contains a `lib/` directory with the desired modules. When invoking the `Launcher`, it locates the available applications, menu and toolbar entries via SPI. Adding or removing Probe, PV Tree, .. from a product is done by simply adding or removing the respective jar file.

10.4 Services

Similar to a product, a service is a runnable tool, but typically for a non-UI functionality like a scan server, archive engine, alarm handler or logger.

CHAPTER 11

Locations

Phoebus uses the following Java System properties to locatehelp files, saved state etc.

These system variables are typically set automatically as described below, but when necassary they can be set when starting the product.

`phoebus.install`: Location where phoebus is installed. Has subdirectories `lib/`, `doc/`, and is used to locate the online help. Is automatically derived from the location of the framework JAR file.

`phoebus.user`: Location where phoebus keeps the memento and preferences. Defaults to `.phoebus` in the user's home directory.

CHAPTER 12

Help System

Help files are **.rst* files with reStructuredText markup as described at http://www.sphinx-doc.org/en/stable/rest.html.

The top-level repository for help files is phoebus-doc (https://github.com/kasemir/phoebus-doc) and a snapshot of the current version is accessible on http://phoebus-doc.readthedocs.io.

The top-level help repository provides the overall structure and content that describes the Phoebus-based CS-Studio in general. Each phoebus application can contribute help content that describes the specific application in more detail. This is done by adding a `doc/` folder with an `index.rst` file to the application sources. When phoebus-doc is built, it includes all `phoebus/**/doc/index.rst` in the Applications section of the manual. While the `*.rst` markup is ultimately converted into HTML, some applications might have already have HTML content generated by other means, for example from Javadoc. Any `doc/html` folder found in the applications source code is copied into the file html folder. To appear in the manual, it needs to be linked from the `index.rst` of an application via `raw` tags. For an example, refer to the display builder editor help.

Overall, the help content is thus generated from a combination of

1. Top-level content defined in `phoebus-doc/source/*.rst`
2. Application specific help copied from the `phoebus/**/doc/index.rst` source tree
3. Pre-generated HTML folders copied from the `phoebus/**/doc/html` source tree

In addition to building the help files locally as described below, or viewing a snapshot of the current version online under the link mentioned above, the help content is also bundled into the phoebus-based CS-Studio product. When the phoebus product is built, it checks for the HTML version of the manual in `phoebus-doc/build/html`. If found, it is bundled into the product.

Complete build steps of manual and product:

```
# Obtain sources for documentation and product
git clone https://github.com/kasemir/phoebus-doc.git
git clone https://github.com/shroffk/phoebus.git

# Some application code may contain html content
# that needs to be generated, for example from Javadoc
( cd phoebus/app/display/editor;  ant -f javadoc.xml clean all )
```

(continues on next page)

(continued from previous page)

```
# Building the manual will locate and include
# all ../phoebus/applications/**/doc/index.rst
( cd phoebus-doc; make clean html )
# Windows: Use make.bat html

# Fetch dependencies
( cd phoebus/dependencies; mvn clean install )

# Build the product, which bundles help from
# ../phoebus-doc/build/html
# as phoebus-product/target/doc
( cd phoebus; ant clean dist )

# Could now run the product
( cd phoebus/phoebus-product; sh phoebus.sh )

# or distribute the ZIP file,
# phoebus/phoebus-product/target/phoebus-0.0.1.zip
```

12.1 Internals

In `phoebus-doc/source/conf.py`, the `createAppIndex()` method checks for the phoebus sources and builds the application section of the manual.

When invoking the Phoebus `Help` menu, it looks for a `doc/` folder in the installation location (see *Locations*).

As a fallback for development in the IDE, it looks for `phoebus-doc/build/html`.

CHAPTER 13

Preference Settings

The following preference settings are available for the various application features. To use them in your settings file, remember to prefix each setting with the package name.

13.1 alarm_logging_preferences.properties

File phoebus/app/alarm/logging-ui/src/main/resources/alarm_logging_preferences.properties:

```
# location of elastic node/s
es_host=130.199.219.152
es_port=9200
es_index=accelerator_alarms
```

13.2 org.csstudio.archive

File phoebus/services/archive-engine/src/main/resources/archive_preferences.properties:

```
# ----------------------------
# Package org.csstudio.archive
# ----------------------------

# RDB URL for archived data
#
# Oracle example
# url=jdbc:oracle:thin:user/password@//172.31.73.122:1521/prod
#
# MySQL example
url=jdbc:mysql://localhost/archive?rewriteBatchedStatements=true

# RDB user and password
```

(continues on next page)

(continued from previous page)

```
# Some applications also provide command-line option to override.
user=archive
password=$archive

# Schema name. Used with an added "." as prefix for table names.
# For now this is only used with Oracle URLs and ignored for MySQL
schema=

# Timeout [seconds] for certain SQL queries
# Fundamentally, the SQL queries for data take as long as they take
# and any artificial timeout just breaks queries that would otherwise
# have returned OK  few seconds after the timeout.
# We've seen Oracle lockups, though, that caused JDBC to hang forever
# because the SAMPLE table was locked. No error/exception, just hanging.
# A timeout is used for operations other than getting the actual data,
# for example the channel id-by-name query which _should_ return within
# a shot time, to catch that type of RDB lockup.
# timeout_secs=120
# With PostgreSQL, the setQueryTimeout API is not implemented,
# and calling it results in an exception.
# Setting the timeout to 0 disables calls to setQueryTimeout.
timeout_secs=0

# Use a blob to read/write array samples?
#
# The original SAMPLE table did not contain an ARRAY_VAL column
# for the array blob data, but instead used a separate ARRAY_VAL table.
# When running against an old database, this parameter must be set to false.
use_array_blob=true

# Name of sample table for writing
write_sample_table=sample

# Maximum length of text samples written to SAMPLE.STR_VAL
max_text_sample_length=80

# Use postgres copy instead of insert
use_postgres_copy=false

# Seconds between log messages for Not-a-Number, futuristic, back-in-time values,␣
↪buffer overruns
# 24h = 24*60*60 = 86400
log_trouble_samples=86400
log_overrun=86400

# Write period in seconds
write_period=30

# Maximum number of repeat counts for scanned channels
max_repeats=60

# Write batch size
batch_size=500

# Buffer reserve (N times what's ideally needed)
buffer_reserve=2.0
```

(continues on next page)

(continued from previous page)

```
# Samples with time stamps this far ahead of the local time
# are ignored
# 24*60*60 = 86400 = 1 day
ignored_future=86400
```

13.3 org.csstudio.display.builder.editor

File phoebus/app/display/editor/src/main/resources/display_editor_preferences.properties:

```
# ----------------------------------------
# Package org.csstudio.display.builder.editor
# ----------------------------------------

# Widget types to hide from the palette
#
# Comma separated list of widget types that will not be shown
# in the palette.
# Existing displays that use these widgets can still be edited
# and executed, but widgets do not appear in the palette to
# discourage adding them to new displays.

# Hiding widgets where representation has not been imported because of dependencies
hidden_widget_types=meter,linear-meter,knob,gauge,clock,digital_clock
```

13.4 org.csstudio.display.builder.model

File phoebus/app/display/model/src/main/resources/display_model_preferences.properties:

```
# ----------------------------------------
# Package org.csstudio.display.builder.model
# ----------------------------------------


# Widget classes
# One or more *.bcf files, separated by ';'
# Defaults to built-in copy of examples/classes.bcf
class_files=examples:classes.bcf

# Named colors
# One or more *.def files, separated by ';'
# Defaults to built-in copy of examples/color.def
color_files=examples:color.def

# Named fonts
# One or more *.def files, separated by ';'
# Defaults to built-in copy of examples/font.def
font_files=examples:font.def

# Global macros, used for all displays.
#
# Displays start with these macros,
```

(continues on next page)

(continued from previous page)

```
# and can then add new macros or overwrite
# the values of these macros.
#
# Format:
# Entries where the XML tag name is the macro name,
# and the XML content is the macro value.
# The macro name must be a valid XML tag name:
# * Must start with character
# * May then contain characters or numbers
# * May also contain underscores
#
macros=<EXAMPLE_MACRO>Value from Preferences</EXAMPLE_MACRO><TEST>true</TEST>


# Timeout [ms] for loading files: Displays, but also color, font, widget class files
read_timeout=10000

# Timeout [sec] for caching files loaded from a URL
cache_timeout=60


# 'BOY' *.opi files provide the font size in 'points'.
# All other positions and sizes are in 'pixels'.
# A point is meant to represent 1/72th of an inch.
# The actual on-screen size display settings.
# Plugging a different monitor into the computer can
# potentially change the DPI settings of the graphics driver,
# resulting in different font sizes.
# The display builder uses fonts in pixels to avoid such changes.
#
# When reading legacy display files, we do not know the DPI
# scaling that was used to create the display.
# This factor is used to translate legacy font sizes
# from 'points' into 'pixel':
#
# legacy_points = pixel * legacy_font_calibration
#
# The test program
#   org.csstudio.display.builder.representation.swt.SWTFontCalibation
# can be used to obtain the factor when executed on the original
# platform where the legacy display files were created.
#
# When loading legacy files,
# _increasing_ the legacy_font_calibration will
# result in _smaller_ fonts in the display builder
legacy_font_calibration=1.01

# Maximum re-parse operations
#
# When reading legacy *.opi files and for example
# finding a "TextUpdate" widget that has no <pv_name>,
# it will be changed into a "Label" widget and then re-parsed.
# If more than a certain number of re-parse operations are triggered
# within one 'level' of the file (number of widgets at the root of the display,
# or number of childred for a "Group" widget),
# the parser assumes that it entered an infinite re-parse loop
# and aborts.
```

(continues on next page)

(continued from previous page)

```
max_reparse_iterations=5000

# When writing a display file, skip properties that are still at default values?
skip_defaults=true
```

13.5 org.csstudio.display.builder.representation

File phoebus/app/display/representation/src/main/resources/display_representation_preferences.properties:

```
# ---------------------------------------------------
# Package org.csstudio.display.builder.representation
# ---------------------------------------------------

## Representation Tuning
#
# The representation 'throttles' updates to widgets.
# When a widget requests an update, a little accumulation time
# allows more updates to accumulate before actually performing
# the queued update requests on the UI thread.
#
# An update delay then suppresses further updates to prevent
# flooding the UI thread.
#
# Update runs that last longer than a threshold can be logged

# Time waited after a trigger to allow for more updates to accumulate
update_accumulation_time = 20

# Pause between updates to prevent flooding the UI thread
update_delay = 100

# Period in seconds for logging update performance
performance_log_period_secs = 5

# UI thread durations above this threshold are logged
performance_log_threshold_ms = 20

# Pause between updates of plots
# Limit to 250ms=4 Hz
plot_update_delay = 250
```

13.6 org.csstudio.display.builder.runtime

File phoebus/app/display/runtime/src/main/resources/display_runtime_preferences.properties:

```
# -----------------------------------------
# Package org.csstudio.display.builder.runtime
# -----------------------------------------

# Search path for Jython scripts used by the display runtime.
# Note that format depends on the OS.
```

(continues on next page)

(continued from previous page)

```
# On UNIX systems, path entries are separated by ':', on Windows by ';'.
# python_path=/home/controls/displays/scripts:/home/fred/my_scripts
python_path=

# PV Name Patches
#
# Translate PV names based on regular expression pattern and replacement
#
# Format:  pattern@replacement@pattern@replacement
#
# Setting must contain a sequence of pattern & replacement pairs,
# all separated by '@'.
#
# The regular expression for the pattern can includes "( )" groups,
# which are then used in the replacement via "$1", "$2", ..
#
# If the item separator character '@' itself is required within the pattern or␣
↪replacement,
# use '[@]' to distinguish it from the item separator, i.e.
#
#    [@]work@[@]home
#
# will patch "be@work" -> "be@home"
#
# Patches are applied in the order they're listed in the preference, i.e.
# later patches are applied to names already patched by earlier ones.
#
# Example:
# Remove PVManager's longString modifier,             'some_pv {"longString":true}' ->
↪ 'some_pv'
# turn constant formula into constant local variable, '=42'                        ->
↪ 'loc://const42(42)'
# as well as constant name into constant local var,   '="Fred"'                    ->
↪ 'loc://strFred("Fred")'
pv_name_patches=\\{"longString":true\\}"@@^=([0-9]+)@loc://const$1($1)@^="([a-zA-Z]+)
↪"@loc://str$1("$1")

# PV update throttle in millisecs
# 250ms = 4 Hz
update_throttle=250
```

13.7 org.csstudio.scan.client

File phoebus/app/scan/client/src/main/resources/scan_client_preferences.properties:

```
# ---------------------------------------
# Package org.csstudio.scan.client
# ---------------------------------------

# Name of host where scan server is running
host=localhost

# TCP port of scan server REST interface
port=4810
```

(continues on next page)

(continued from previous page)

```
# Poll period [millisecs] of the scan client (scan monitor, plot, ...)
poll_period=1000
```

13.8 org.csstudio.trends.databrowser3

File phoebus/app/databrowser/src/main/resources/databrowser_preferences.properties:

```
# ----------------------------------------
# Package org.csstudio.trends.databrowser3
# ----------------------------------------

# Default auto scale value
# Possible values are: true to enable the automatic calculation of the min/max Y-axis,
↪ or false to use min/max fixed values.
use_auto_scale=false

# Default time span displayed in plot in seconds
time_span=3600

# Default scan period in seconds. 0 for 'monitor'
scan_period=0.0

# Default plot update period in seconds
update_period=3.0

# .. elements in live sample buffer
live_buffer_size=5000

# Default line width
line_width=2

# Opacity of 'area'
#   0%: Area totally transparent (invisible)
#  20%: Area quite transparent
# 100%: Area uses  solid color
opacity=40

# Default trace type for newly created traces.
# Allowed values are defined by org.csstudio.trends.databrowser3.model.TraceType:
# AREA, ERROR_BARS, SINGLE_LINE, AREA_DIRECT, SINGLE_LINE_DIRECT, SQUARES, ...
trace_type=AREA

# Delay in milliseconds that delays archive requests when
# the user moves the time axis to avoid a flurry of archive requests
# while interactively zooming and panning
archive_fetch_delay=500

# Number of binned samples to request for optimized archive access.
# Negative values scale the display width,
# i.e. -3 means: 3 times Display pixel width.
plot_bins=-3

# Suggested data servers
```

(continues on next page)

(continued from previous page)

```
# Format:  <url>*<url>|<name>
# List of URLs, separated by '*'.
# Each URL may be followed by an "|alias"
urls=jdbc:mysql://localhost/archive|RDB*xnds://localhost/archive/cgi/
↪ArchiveDataServer.cgi

# Default data sources for newly added channels
# Format: Same as 'urls'
archives=jdbc:mysql://localhost/archive|RDB*xnds://localhost/archive/cgi/
↪ArchiveDataServer.cgi

# When opening existing data browser plot,
# use archive data sources specified in the configuration file (original default)
# or ignore saved data sources and instead use the preference settings?
use_default_archives=false

# If there is an error in retrieving archived data,
# including that the channel is not found in the archive,
# should this be displayed in a dialog box,
# or logged as a WARNING (and thus visible on the console)?
prompt_for_errors=false

# Re-scale behavior when archived data arrives: NONE, STAGGER
archive_rescale=STAGGER

# Shortcuts offered in the Time Axis configuration
# Format:
# Text for shortcut,start_spec|Another shortcut,start_spec
time_span_shortcuts=30 Minutes,-30 min|1 Hour,-1 hour|12 Hours,-12 hour|1 Day,-1␣
↪days|7 Days,-7 days

#It is a path to the directory where the PLT files for WebDataBrowser are placed.
plt_repository=/opt/codac/opi/databrowser/

#SendEmailAction default sender
# By defining a default email, users who select "Email.." from the context menu
# do not need to enter an email address.
# If left empty, elog dialog will require users to enter a "From:" address for the␣
↪sender.
email_default_sender=

# Automatically refresh history data when the liver buffer is full
# This will prevent the horizontal lines in the shown data when the buffer
# is too small to cover the selected time range
automatic_history_refresh=false

# Scroll step, i.e. size of the 'jump' left when scrolling, in seconds.
# (was called 'future_buffer')
scroll_step = 5

# Display the trace names on the Value Axis
# the default value is "true". "false" to not show the trace names on the Axis
use_trace_names = true

# Prompt / warn when trying to request raw data?
prompt_for_raw_data_request = true
```

(continues on next page)

(continued from previous page)

```
# Prompt / warn when making trace invisible?
prompt_for_visibility = true
```

13.9 org.phoebus.app.viewer3d

File phoebus/app/3dViewer/src/main/resources/3d_viewer_preferences.properties:

```
# ---------------------------------
# Package org.phoebus.app.viewer3d
# ---------------------------------

# Time out for reading from a URI
read_timeout=10000

# Default directory for the file chooser.
default_dir=$(user.home)

# Cone is approximated with these many faces.
# 3: Triangular base, most minimalistic
# 8: Looks pretty good
# Higher: Approaches circular base,
# but adds CPU & memory usage
# and doesn't really look much better
cone_faces=8
```

13.10 org.phoebus.applications.alarm

File phoebus/app/alarm/model/src/main/resources/alarm_preferences.properties:

```
# ------------------------------------
# Package org.phoebus.applications.alarm
# ------------------------------------

# Kafka Server host:port
server=localhost:9092

# Name of alarm tree root
config_name=Accelerator

# Names of selectable alarm configurations
# The `config_name` will be used as the default for newly opened tools,
# and if `config_names` is empty, it remains the only option.
# When one or more comma-separated configurations are listed,
# the UI shows the selected name and allows switching
# between them.
config_names=Accelerator, Demo

# Timeout in seconds for initial PV connection
connection_timeout=30
```

(continues on next page)

(continued from previous page)

```
## Area Panel

# Item level for alarm area view:
# 1 - Root element
# 2 - Top-level "area" elements just below root
# 3 - Show all the items at level 3
alarm_area_level=2

# Number of columns in the alarm area view
alarm_area_column_count=3

# Gap between alarm area panel items
alarm_area_gap=5

# Font size for the alarm area view
alarm_area_font_size=15

# Limit for the number of context menu items.
# Separately applied to the number of 'guidance',
# 'display' and 'command' menu entries.
alarm_menu_max_items=10

# Alarm table row limit
# If there are more rows, they're suppressed
alarm_table_max_rows=2500

# Directory used for executing commands
# May use Java system properties like this: $(prop_name)
command_directory=$(user.home)

# The threshold of messages that must accumulate before the annunciator begins to␣
↪simply state: "There are X Alarm messages."
annunciator_threshold=3

# The number of messages the annunciator will retain before popping messages off the␣
↪front of the message queue.
annunciator_retention_count=100

# Timeout in seconds at which server sends idle state updates
# for the 'root' element if there's no real traffic.
# Client will wait 3 times this long and then declare a timeout.
idle_timeout=10

# Name of the sender, the 'from' field of automated email actions
automated_email_sender=Alarm Notifier <alarm_server@example.org>

# Comma-separated list of automated actions on which to follow up
# Options include mailto:, cmd:
automated_action_followup=mailto:, cmd:

# Optional heartbeat PV
# When defined, alarm server will set it to 1 every heartbeat_secs
#heartbeat_pv=Demo:AlarmServerHeartbeat
heartbeat_pv=

# Heartbeat PV period in seconds
heartbeat_secs=10
```

(continues on next page)

(continued from previous page)

```
# Period for repeated annunciation
#
# If there are active alarms, i.e. alarms that have not been acknowleded,
# a message "There are 47 active alarms" will be issued
#
# Format is HH:MM:SS, for example 00:15:00 to nag every 15 minutes.
# Set to 0 to disable
nag_period=00:15:00
```

13.11 org.phoebus.applications.filebrowser

File phoebus/app/filebrowser/src/main/resources/filebrowser_preferences.properties:

```
# ------------------------------------------
# Package org.phoebus.applications.filebrowser
# ------------------------------------------

# Initial root directory for newly opened file browser
# May use system properties like "$(user.home)".
# At runtime, user can select a different base directory,
# but pressing the "Home" button reverts to this one.
default_root=$(user.home)

# Show hidden files (File.isHidden)?
show_hidden=false
```

13.12 org.phoebus.applications.pvtable

File phoebus/app/pvtable/src/main/resources/pv_table_preferences.properties:

```
# ---------------------------------------
# Package org.phoebus.applications.pvtable
# ---------------------------------------

# Should all BYTE[] values be considered "long strings"
treat_byte_array_as_string=true

# Show the units when displaying values?
show_units=true

# Show a "Description" column that reads xxx.DESC?
show_description=true

# Default tolerance for newly added items
tolerance=0.1

# Maximum update period for PVs in millisecs
max_update_period=500
```

13.13 org.phoebus.applications.pvtree

File phoebus/app/pvtree/src/main/resources/pv_tree_preferences.properties:

```
# ---------------------------------------
# Package org.phoebus.applications.pvtree
# ---------------------------------------

# The channel access DBR_STRING has a length limit of 40 chars.
# Since EPICS base R3.14.11, reading fields with an added '$' returns
# their value as a char[] without length limitation.
# For older IOCs, this will however fail, so set this option
# only if all IOCs are at least version R3.14.11
read_long_fields=true

# For each record type, list the fields to read and trace as 'links'.
#  Format: record_type (field1, field2) ; record_type (...)
#
# Fields can simply be listed as 'INP', 'DOL'.
# The syntax INPA-L is a shortcut for INPA, INPB, INPC, ..., INPL
# The syntax INP001-128 is a shortcut for INP001, INP002, ..., INP128
# The general syntax is "FIELDxxx-yyy",
# where "xxx" and "yyy" are the initial and final value.
# "xxx" and "yyy" need to be of the same length, i.e. "1-9" or "01-42", NOT "1-42".
# For characters, only single-char "A-Z" is supported, NOT "AA-ZZ",
# where it's also unclear if that should turn into AA, AB, AC, .., AZ, BA, BB, BC, ..,
↪ ZZ
# or AA, BB, .., ZZ
#
# bigASub is a CSIRO/ASCAP record type, doesn't hurt to add that to the shared␣
↪configuration
#
# scalcout is a bit unfortunate since there is no shortcut for INAA-INLL.

fields=aai(INP);ai(INP);bi(INP);compress(INP);longin(INP);mbbi(INP);mbbiDirect(INP);
↪mbboDirect(INP);stringin(INP);subArray(INP);waveform(INP);aao(DOL);ao(DOL);bo(DOL);
↪fanout(DOL);longout(DOL);mbbo(DOL);stringout(DOL);sub(INPA-L);genSub(INPA-L);
↪calc(INPA-L);calcout(INPA-L);aSub(INPA-U);seq(SELN);bigASub(INP001-128);
↪scalcout(INPA-L,INAA,INBB,INCC,INDD,INEE,INFF,INGG,INHH,INII,INJJ,INKK,INLL)


# Max update period in seconds
update_period=0.5
```

13.14 org.phoebus.applications.update

File phoebus/app/update/src/main/resources/update_preferences.properties:

```
# ---------------------------------------
# Package org.phoebus.applications.update
# ---------------------------------------

# Version time/date
#
# If the distribution found at the `update_url`
```

(continues on next page)

(continued from previous page)

```
# is later than this date, an update will be performed.
#
# The updated distribution must contain a new value for
# the org.phoebus.applications.update/current_version setting.
#
# By for example publishing updates with a 'current_version'
# that's one month ahead, you can suppress minor updates
# for a month.
#
# Format: YYYY-MM-DD HH:MM
#current_version=2018-06-18 13:10
current_version=


# Location where updates can be found
#
# The file:, http: or https: URL is checked.
# If it exists, and its modification time is after `current_version`,
# the updated distribution is downloaded
# and the current Locations.install() is replaced.
#
# Location may include system properties
# and $(arch) will be replaced by "linux", "mac" or "win"
# to allow locations specific to each architecture.
#
# Empty: Do not perform any update check
update_url=
# update_url=https://controlssoftware.sns.ornl.gov/css_phoebus/nightly/product-sns-
↪$(arch).zip
```

13.15 org.phoebus.archive.reader.appliance

File phoebus/app/databrowser/src/main/resources/appliance_preferences.properties:

```
# ---------------------------------------
# Package org.phoebus.archive.reader.appliance
# ---------------------------------------

useStatisticsForOptimizedData=true
useNewOptimizedOperator=true
```

13.16 org.phoebus.archive.reader.rdb

File phoebus/app/databrowser/src/main/resources/archive_reader_rdb_preferences.properties:

```
---------------------------------------
# Package org.phoebus.archive.reader.rdb
# ---------------------------------------

# User and password for reading archived data
user=archive
```

(continues on next page)

(continued from previous page)

```
password=$archive

# Table prefix
# For Oracle, this is typically the schema name,
# including "."
prefix=

# Timeout [seconds] for certain SQL queries
# Fundamentally, the SQL queries for data take as long as they take
# and any artificial timeout just breaks queries that would otherwise
# have returned OK a few seconds after the timeout.
# We've seen Oracle lockups, though, that caused JDBC to hang forever
# because the SAMPLE table was locked. No error/exception, just hanging.
# A timeout is used for operations other than getting the actual data,
# for example the channel id-by-name query which _should_ return within
# a shot time, to catch that type of RDB lockup.
timeout_secs=120
# Setting the timeout to 0 disables calls to setQueryTimeout,
# which may be required for PostgreSQL where the setQueryTimeout API is not␣
↪implemented.
# timeout_secs=0


# Use a BLOB to read array samples?
#
# The original SAMPLE table did not contain an ARRAY_VAL column
# for the array blob data, but instead used a separate ARRAY_VAL table.
# When running against an old database, this parameter must be set to false.
use_array_blob=true

# Use stored procedures and functions for 'optimized' data readout?
# Set to procedure name, or nothing to disable stored procedure.
stored_procedure=
starttime_function=

# MySQL:
# stored_procedure=archive.get_browser_data

# PostgreSQL
# stored_procedure=public.get_browser_data

# Oracle:
# stored_procedure=chan_arch.archive_reader_pkg.get_browser_data
# starttime_function=SELECT chan_arch.archive_reader_pkg.get_actual_start_time (?, ?,␣
↪?)  FROM DUAL


# JDBC Statement 'fetch size':
# Number of samples to read in one network transfer.
#
# For Oracle, the default is 10.
# Tests resulted in a speed increase up to fetch sizes of 1000.
# On the other hand, bigger numbers can result in java.lang.OutOfMemoryError.
fetch_size=1000
```

13.17 org.phoebus.email

File phoebus/core/email/src/main/resources/email_preferences.properties:

```
# -------------------------
# Package org.phoebus.email
# -------------------------

# smtp host
# When set to "DISABLE", email support is disabled
mailhost=smtp.bnl.gov

# smtp port
mailport=25

# User and password for connecting to the mail host, usually left empty
username=
password=
```

13.18 org.phoebus.framework.workbench

File phoebus/core/framework/src/main/resources/workbench_preferences.properties:

```
# ---------------------------------------
# Package org.phoebus.framework.workbench
# ---------------------------------------

# External applications
#
# Defines applications to use for specific file extensions
#
# Format:
#
# Each definition consists of name, file extensions, command.
#
# Name is the name of the definition, used to register the application.
# File extensions is a '|'-separated list of file extensions (not including the 'dot
↪').
# Command is the path to the command.
#
# Multiple definitions are separated by ';'
#
# Example:
#
# Start 'gedit' for text files, 'eog' for images, 'firefox' for PDF files
# external_apps=Text Editor,txt|dat|py|ini|db|xml|xsl|css|cmd|sh|st|log|out|md|shp,
↪gedit; Image Viewer,png|jpg|gif|jpeg,eog; PDF Viewer,pdf,firefox


#
# The command will be invoked with the full path to the file.
external_apps=

# Directory where external applications are started
```

(continues on next page)

(continued from previous page)

```
# May use system properties
external_apps_directory=$(user.home)
```

13.19 org.phoebus.logbook.ui

File phoebus/app/logbook/ui/src/main/resources/log_ui_preferences.properties:

```
# ------------------------------
# Package org.phoebus.logbook.ui
# ------------------------------

# Site specific log book implementation name.
# When empty, logbook submissions are disabled
logbook_factory=inmemory

# Comma-separated list of default logbooks for new log entries.
default_logbooks=Scratch Pad

# Whether or not to save user credentials to file so they only have to be entered␣
↪once when making log entries.
save_credentials=false
```

13.20 org.phoebus.pv

File phoebus/core/pv/src/main/resources/pv_preferences.properties:

```
# ----------------------
# Package org.phoebus.pv
# ----------------------

# Default PV Type
default=ca
```

13.21 org.phoebus.pv.ca

File phoebus/core/pv/src/main/resources/pv_ca_preferences.properties:

```
# -------------------------
# Package org.phoebus.pv.ca
# -------------------------

# Channel Access address list
addr_list=

auto_addr_list=true

max_array_bytes=100000000

server_port=5064
```

(continues on next page)

(continued from previous page)

```
repeater_port=5065

beacon_period=15

connection_timeout=30

# Support variable length arrays?
# auto, true, false
variable_length_array=auto

# Connect at lower priority for arrays
# with more elements than this threshold
large_array_threshold= 100000

# Is the DBE_PROPERTY subscription supported
# to monitor for changes in units, limits etc?
dbe_property_supported=false

# Mask to use for subscriptions
# VALUE, ALARM, ARCHIVE
monitor_mask=VALUE
```

13.22 org.phoebus.pv.mqtt

File phoebus/core/pv/src/main/resources/pv_mqtt_preferences.properties:

```
# ----------------------------
# Package org.phoebus.pv.mqtt
# ----------------------------

# MQTT Broker
# All "mqtt://some/tag" PVs will use this broker
mqtt_broker=tcp://localhost:1883
```

13.23 org.phoebus.security

File phoebus/core/security/src/main/resources/phoebus_security_preferences.properties:

```
# ----------------------------
# Package org.phoebus.security
# ----------------------------

# Authorization file
#
# If left empty, the built-in core/security/authorization.conf is used.
#
# When specifying a plain file name like "authorization.conf",
# the install location (Locations.install()) is searched for that file name.
#
# The file name can also be an absolute path like /some/path/auth.conf.
```

(continues on next page)

(continued from previous page)

```
#
# Finally, the file name may use a system property like $(auth_file)
# which in turn could be set to either BUILTIN, a file in the install location,
# or an absolute path.
#
# When set to an invalid file, the user will have no authorizations at all.

# Use built-in core/security/authorization.conf
authorization_file=

# Use authorization.conf in the install location
#authorization_file=authorization.conf
```

13.24 org.phoebus.ui

File phoebus/core/ui/src/main/resources/phoebus_ui_preferences.properties:

```
# ----------------------
# Package org.phoebus.ui
# ----------------------

# Show the splash screen?
# Can also be set via '-splash' resp. '-nosplash' command line options
splash=true

# Default applications
#
# When there are multiple applications that handle
# a resource, the setting determins the one used by default.
#
# Format is comma-separated list with sub-text of default application names.
# For example, "run, exe" would pick "display_runtime" over "display_editor",
# and "foo_executor" over "foo_creator".
# The patterns "edit, creat" would inversely open the editor-type apps.
#
# This makes the display_runtime and the 3d_viewer default apps,
# using display_editor and a potentially configured text editor for *.shp files␣
↪secondary
default_apps=run,3d

# Top resources to show in "File" menu and toolbar
#
# Format:
# uri1 | uri2,Display name 2 | uri3,Display name 3
top_resources=examples:/01_main.bob?app=display_runtime,Example Display | pv://?sim://
↪sine&app=probe,Probe Example | pv://?sim://sine&loc://x(10)&app=pv_table,PV Table␣
↪Example | http://www.google.com?app=web, Google

# Home display file. "Home display" button will navigate to this display.
home_display=examples:/01_main.bob?app=display_runtime,Example Display

# UI Responsiveness Monitor Period
# Period between tests [millisec],
# i.e. the minimum detected UI freeze duration
```

(continues on next page)

(continued from previous page)

```
# Set to 0 to disable
ui_monitor_period=500
```

CHAPTER 14

Appendix

- genindex
- modindex
- search