**Solaris Synchrotron Control Program Extensions: Device Group Panel**

User Manual

|  |  |
| --- | --- |
| Revision: |  |
| Status: | Released |
| Repository: |  |
| Project: |  |
| Folder: |  |
| Document ID: |  |
| File: |  |
| Owner: | Vid Juvan |
| Last modification: | June 16, 2015 |
| Created: | April 28, 2014 |

|  |
| --- |
| **Prepared by** |
| Vid Juvan |
|  |

Document History



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision | Date | Changed/reviewed | Section(s) | Modification |
| 1.0 | 23-4-2014 | Vid Juvan |  | Initial version |

Confidentiality

This document is classified as a public document. As such, it or parts thereof are openly accessible to anyone listed in the Audience section, either in electronic or in any other form.

Scope

This document provides a description and instructions how to use the Solaris Synchrotron Control Program.

Table of Contents

[1. Introduction 4](#_Toc387924571)

[2. Software installation 5](#_Toc387924572)

[2.1. Prerequisites 5](#_Toc387924573)

[2.2. Facility Configuration Software Installation 5](#_Toc387924574)

[2.3. Solaris Synchrotron Control Program Installation 5](#_Toc387924575)

[2.4. Running the application 5](#_Toc387924576)

[2.4.1. Manual start 6](#_Toc387924577)

[2.4.2. Runner application 6](#_Toc387924578)

[3. Configuration 7](#_Toc387924579)

[4. Operation 9](#_Toc387924580)

[4.1. Device Tree 9](#_Toc387924581)

[4.1.1. Device Filtering 9](#_Toc387924582)

[4.1.2. Standard Device Panel 9](#_Toc387924583)

[4.1.3. Custom Panel 10](#_Toc387924584)

[4.2. Device Group Tree 10](#_Toc387924585)

[4.2.1. Device Group Panel 11](#_Toc387924586)

Figures

**No table of figures entries found.**

Tables

[Table 1: Configuration Format 7](#_Toc387924567)

[Table 2: Configuration Specifications 8](#_Toc387924568)

[Table 3: Standard Device Panel Script Input Parameters 10](#_Toc387924569)

[Table 4: Device Group Script Input Parameters 11](#_Toc387924570)

Typography

This document uses the following styles:

|  |  |
| --- | --- |
|  | A box like this contains important information. |

|  |  |
| --- | --- |
|  | A box like this would contain sidebar text. |

|  |  |  |
| --- | --- | --- |
|  | Warning!  A box like this provides information, which should not be disregarded! |  |

Glossary of Terms

|  |  |
| --- | --- |
| Tango | TANGO is an object oriented distributed control system (http://www.tango-controls.org/) |
|  |  |
|  |  |
|  |  |

References

1. Tango-related documentation: <http://www.tango-controls.org/>
2. Solaris Synchrotron Control Program: <http://git.m.cps.uj.edu.pl/controlroomsoftware/app-cosylab-controlprogram>
3. Device Group Panel template <http://git.m.cps.uj.edu.pl/controlroomsoftware/app-cosylab-templategroupgui>

# Introduction

Solaris Synchrotron Control Program provides a high-level overview of the operation of the synchrotron devices and enables access to the device engineering screens. It is instantiated according to the provided main device CSV file and connects to the corresponding Tango database of the control system instance.

The main purpose of the application is to provide the user with easy access to various visual control panels of the specified devices. Three types of panels can be accessed, namely Standard Device Panel, Custom Panel and Device Group Panel. This documentation regards the latter.

The concern of this documentation is strictly just the Device Group Panel for the Solaris Synchrotron Control Program. The documentation regarding the Control Program itself can be found [2]. In the scope of this documentation, we will only describe the functionality of the Control Program that directly concerns the implementation of the Device Group Panels.

# Device Group Panel

The Device Group Panel is a GUI that exposes multiple devices within one GUI. All of them are custom made, but foreseen to be dynamic to a certain extent. Usually, multiple instances of the same Device Group Panel are used within the control program, differing only in the configuration.

For example, the following Device Group GUI instances are created within the Control Program:

* VAC\_K00
* VAC\_K01
* VAC\_K02
* VAC\_K03
* …

They all use the same Device Group Panel implementation, but feed it with different configuration.

Device Group Panels are generally built on a Control Program Device Group template. The Device Group Panel template is provided [3]. Henceforth, we will consider the provided template as a base for Device Group Panel development.

## Device Group Panel start procedure

The Control Program parses instances of the Device Group Panels from the configuration. When running any instance of the panel, the following process takes place:

* The Control Program looks for the name of the python file, corresponding to the Device Group Panel instance that we want to open.
  + The name of the python file is the same as the name of the Device Group Panel instance, omitting a possible integer suffix. For example, Device Group Panel instance with a name “VAC\_K00” would correspond to a python file “VAC\_K.py”.
* The Control Program look for a command called “getGuiWidget” within the python file. If the function exists:
  + Control Program runs the function and passes the entire configuration for the Device Group Panel to the function in a form of an array. For more derail on the configuration, refer to the next chapters.
  + Control Program expects an instance of the QDialog in return. The QDialog is then shown under the control of the Control Program. In such a way, various optimizations are enabled.
* If the function does not exist:
  + The Control Program runs the python file in a separate sub-process. This is useful for integrating certain external applications into the Control Program.

## Directory and file structure

The Control Program uses a dedicated GUI directory for custom and device group GUIs. The python files used for running the Device Group Panels must be located in that directory, on the first level.

For more transparent development, each Device Group Panel is kept in a separate GIT repository. They are included in the dedicated GUI directory as GIT submodules, each in a separate directory. For every Device Group Panel, a dedicated runner script is created and placed in the GUI directory, alongside the Device Group Panel submodules. The runner script is a small python file that is being called by the Control Program. The correct naming of runner scripts is essential. For every Device Group Panel, multiple runner scripts can be created, supporting different names of the Device Group instances within the Control Program. When the Control Program starts the Device Group Panel, it calls a function “getGuiWidget” in the corresponding runner script. The script then imports code from the Device Group Panel submodule, and runs a function “getGuiWidget” there. The QDialog is returned to the Control Program. The runner scripts serve only for forwarding the calls of the Control Program and are regarded as part a of the Control Program, and not as a part of Device Group Panels.

For example, the file structure for the vacuum Device Group Panel:

* Synchrotron\_GUIs (dedicated GUI directory of the Control Program)
  + VAC\_K.py (runner script, contains “getGuiWidget” function)
  + VAC\_R1\_.py (another runner script)
  + app-cosylab-vacuumgroupgui (Device Group Panel repository as submodule)
    - VacuumGroupGUI.py (contains “getGuiWidget” function)
    - VacuumGroup.py
    - IonPumpWidget.py
    - …

## Device Group Panel structure

Device Group Panel structure can be realized in a desired way. The only limitations are:

* A function called “getGuiWidget” must be implemented in one of the files.
* Function “getGuiWidget” accepts an array of configuration parameters.
* Function “getGuiWidget” returns an instance of QDialog.
* Device Group Panel must not initialize a new instance of the Taurus application.

In this section, we will describe a structure of a standard implementation of the Device Group Panel. The Device Group Panel template provides such a structure. TODO

## Additional Device Group Panel functionality

To enhance the Device Group Panel, the following additions were implemented:

* Chaining Device Group Panel instances
  + For every instance of the Device Group Panel, one can define a set of preceding and succeeding instances of the Device Group Panels.
  + For every preceding or succeeding element in the chain, the Control Program can call a function of the Device Group Panel and provide a callback for opening the preceding/succeeding instance. The Device Group Panel can handle this in a desired way.
  + A simple example would be to generate buttons on the left and the right side of the Device Group Panel. When a user would click on a button, the Control Program would open a preceding/succeeding instance of the Device Group Panel, and close the current instance.
* Communication between Device Group Panel and the Control Program
  + The Control Program can provide an instance of itself to the Device Group Panel. In such a way, a Device Group Panel has access to the functionality of the Control Program.

### Chaining Device Group Panel instances

TODO

### Communication between Device Group Panel and the Control Program

TODO

## Device Group Panel configuration

Device Group GUI receives the entire configuration from the Control Program in a form of the input parameters. However, in order for the user to fully understand the configuration procedure to tackle the development of Device Group GUIs, one needs to be familiar with how the Control Program reads and forwards the configuration.

Two sources are used for configuring Device Group Panels. The former is referred to as the main device CSV configuration file. The latter is referred to as the group configuration file. In the following chapters we explain the syntax and provide the description for both.

### Device configuration

The device configuration CSV is the main configuration file for the Control Program. It is device oriented. Below is provided a device configuration BNF format. Here we only provide the details of the configuration that is relevant for the Device Group Panels. For more information, refer to the Control Program documentation [2].

|  |  |
| --- | --- |
| **Device Configuration BNF Format** | |
| CSV configuration: | 1\*(DEVICE\_SPECIFICATION) |
| DEVICE\_SPECIFICATION | ELEMENT\_NAME, “,”,  TYPE, “,”,  L, “,”, S, “,”, X, “,”, Y, “,”, Z, “,”, SECTION, “,”,  SUBSYSTEM, “,”,  MANAGED\_IN\_CS, “,”,  DEVICE\_SERVER\_NAME, “,”, DEVICE\_SERVER\_INSTANCE, “,”,  DEVICE\_CLASS, “,”,  FULL\_TANGO\_DEVICE\_NAME, “,”,  DEVICE\_ALIAS, “,”,  TRIGGERED\_BY\_TTL, “,”,  CUSTOM\_GUI, “,”,  DEVICE\_GROUP\_GUIS, “,”,  DESCRIPTION, “,”,  COMMENT; |
| DEVICE\_GROUP\_GUIS | “” | (DEVICE\_GROUP\_GUI, 0\*(“|”, DEVICE\_GROUP\_GUI)); |
| DEVICE\_GROUP\_GUI | DEVICE\_GROUP\_INSTANCE, “-”, DEVICE\_GROUP\_ID; |
| DEVICE\_GROUP\_INSTANCE | SCRIPT\_NAME, INTEGER\_SUFFIX; |

Table 2: Device Configuration Format

Upon the device configuration, the Control Program parses all different Device Group Panel instances and all devices that concern it. The Control Program assembles a configuration array that is sent to the Device Group Panel runner function upon opening the panel. The process is described below on a simple example.

**Example**

Assume the following configuration:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ELEMENT\_NAME** | **…** | **FULL\_TANGO\_DEVICE\_NAME** | **…** | **DEVICE\_GROUP\_GUIS** | **…** |
| SOME\_VALVE1 |  | SOME/VALVE/1 |  | VAC\_K00-VGM\_R|VAC\_K01-VGM\_L |  |
| SOME\_ION\_PUMP1 |  | SOME/IONPUMP/1 |  | VAC\_K00 -IPC1 |  |
| SOME\_ION\_PUMP2 |  | SOME/IONPUMP/2 |  | VAC\_K01 -IPC1 |  |

The Control Program would define the following:

|  |  |  |
| --- | --- | --- |
| **Device Group Panel instance** | **Devices** | **Device IDs** |
| VAC\_K00 | SOME/VALVE/1 | VGM\_R |
| SOME/IONPUMP/1 | IPC1 |
| VAC\_K01 | SOME/VALVE/1 | VGM\_L |
| SOME/IONPUMP/2 | IPC1 |

For every Device Group Panel instance, the Control Program prepares its configuration. The configuration is in a form of an array. When a user opens an instance of the Device Group Panel, this configuration is passed to the “getGuiWidget” function of the corresponding python file.

The Device Group Panel configuration array is assembled following as follows:

* A configuration array contains the flag for label (“--LAB”), followed by the Device Group Panel instance name.
  + E.g. CONF\_ARRAY = [“--LAB”, “VAC\_K00”, …]
* For every device that corresponds to the Device Group Panel instance, three items are added: device ID flag(--DEVICE\_ID), full Tango device name, device description. They are added consecutive, in the above order.
  + E.g. CONF\_ARRAY = [… , “—VGM\_R”, “SOME/VALVE/1”, “DESCRIPTION”, …]

For the above example of device configuration, Control Program would generate the following Device Group Panel configuration arrays:

|  |  |
| --- | --- |
| **Device Group Panel instance** | **Configuration array** |
| VAC\_K00 | [“--LAB”, “VAC\_K00”,  “—VGM\_R”, “SOME/VALVE/1”, “DESCRIPTION”  “—IPC1”, “SOME/IONPUMP/1”, “DESCRIPTION”] |
| VAC\_K01 | [“--LAB”, “VAC\_K01”,  “—VGM\_L”, “SOME/VALVE/1”, “DESCRIPTION”  “—IPC1”, “SOME/IONPUMP/2”, “DESCRIPTION”] |

### Group configuration

Group configuration is a separate configuration CSV file, used only for configuring the Device Group Panels. Three configuration elements can be placed for every Device Group Panel instance, namely a set of ascending Device Group Panel instances, a set of descending instances, and an additional input parameter. Bellow we provide a BNF format of this configuration file.

|  |  |
| --- | --- |
| **Group Configuration BNF Format** | |
| CSV configuration: | 1\*(DEVICE\_GROUP\_SPECIFICATION) |
| DEVICE\_GROUP\_SPECIFICATION | DEVICE\_GROUP\_PANEL\_INST\_NAME, “;”,  PRECEEDING\_INSTANCES, “;”,  SUCEEDING INSTANCES, “;”,  ADDITIONAL\_PARAMETER, “\n” |
| PRECEDING\_INSTANCES | “” | (PRE\_DEV\_GROUP\_PANEL\_INST\_NAME, 0\*(“|”, GROUP\_GUI\_NAME)) |
| SUCEEDING\_INSTANCES | “” | (SUC\_DEV\_GROUP\_PANEL\_INST\_NAME, 0\*(“|”, GROUP\_GUI\_NAME)) |
| ADDITIONAL\_PARAMETER | Any string. It can contain “\n”, if entire string is enclosed in quotes. |

#### Additional parameter

The Control Program according to the configuration updates the configuration array for every Device Group Panel instance as follows:

* If an additional parameter is not empty:
  + A flag for additional parameter(“--ADD”) is added to the configuration array, followed by the actual additional parameter.
    - CONF\_ARRAY += [“--ADD”, ADDITIONAL\_PARAMETER]

With this, we achieve a way of sending additional information to the Device Group Panel, separately for every instance. It is up to the Device Group Panel implementation to handle this additional input. A simple example when this would be useful is when PLC signals showing temperatures have to be specified for each instance of the vacuum Device Group Panel.

#### Preceding and succeeding instances

For every Device Group Panel instance, the Control Program notes all the preceding and succeeding instances.

When the Control Program acquires the QDialog of a certain Device Group Panel instance, it executed the following:

* For every succeeding instance, it calls a function of the returned QDialog called “addNext” (if it exists). As arguments to the function, the Control Program first provides the callback function, which has to be called, when a current Device Group Panel instance wants to open that particular succeeding instance. As the second argument, the control program provides an index, which has to be passed to the callback function. As the third argument, the Control Program provides a short description of the succeeding instance.
* For every preceding instance, it calls a function of the returned QDialog called “addPrev” (if it exists). The arguments are the same as for the “addNext” function.

The Device Group Panel can handle these calls in a desired manner.