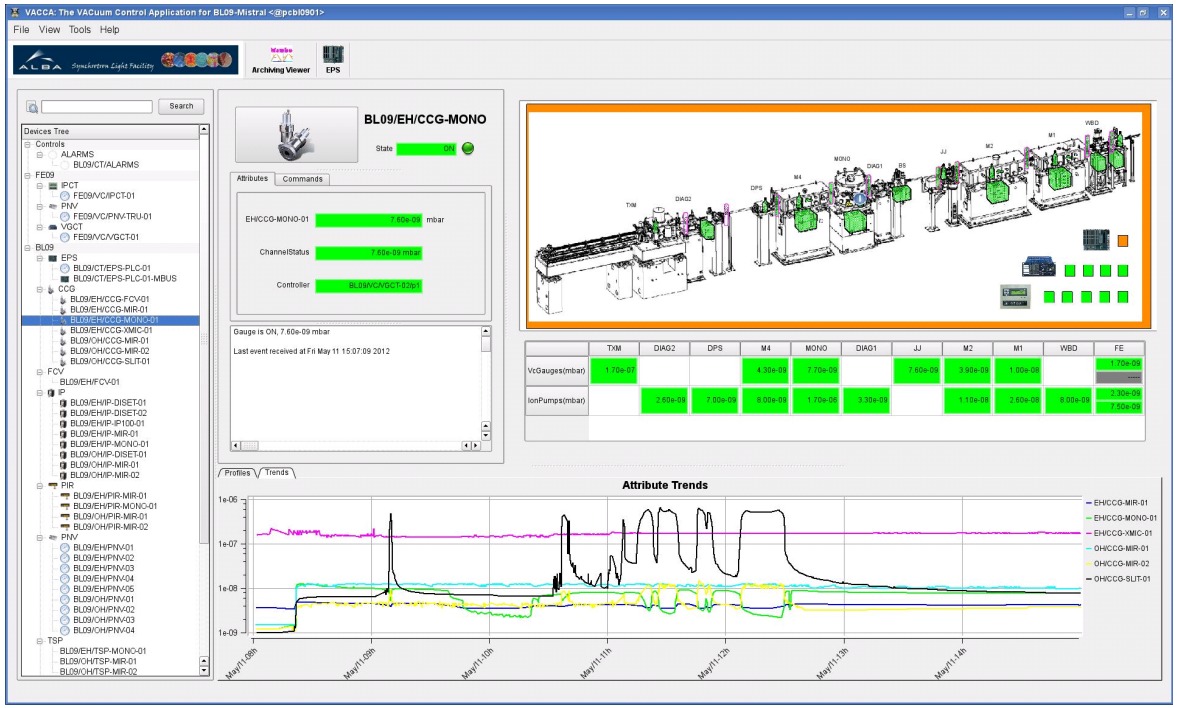
Vacca, the VACuum Controls Application

Vacca is a synoptic-oriented Taurus gui. It is highly customizable, using python files to setup the devices/attributes/icons/commands to be displayed in several grid/tree/profile/synoptic widgets.



Links

Getting Vacca

Source is available from sourceforge:

- svn co https://svn.code.sf.net/p/tango-cs/code/tools/vacca/trunk

Containing 2 python modules to be installed within your $PYTHONPATH :

- **vacca**: it contains the code/widgets specific for vacca.
- **vaccagui**: it is a **taurusgui** application that loads vacca module.

Dependencies of vacca are Taurus and fandango:

- https://pypi.python.org/packages/source/t/taurus/taurus-3.1.0.tar.gz
- svn co https://svn.code.sf.net/p/tango-cs/code/share/fandango/trunk/fandango

Other Links

Vacca is a Tango User Interface:

http://www.tango-controls.org/Members/srubio/vacca

Sources are available at sourceforge within Tango-CS project:

http://sourceforge.net/p/tango-cs/code/HEAD/tree/tools/vacca/trunk

The Vacuum GUIs have a help page in cells intranet

http://www.cells.es/Intranet/Divisions/Computing/Controls/Help/Vacuum/vacca_gui/

The development of Taurus widgets was in Trac:

http://pc148.cells.es:8077/Synoptics

You can find some information regarding devices in the Vacuum project wiki:

http://redmine.cells.es/wiki/vacuum/

Using Vacca

Overview of the application

The Vacuum Control Application for ALBA is based in the interaction between 4 main panels:

The Tree

The device tree in the left-side allows to search for any vacuum device in the machine and to access their attributes values.
Writing just part of a name (e.g. 'eh pnv 02') in the Search field (or Ctrl+f) you can go directly to any element.
Right-button click on any device name open the attribute list, to add attributes to plots or see the archived values.

The central panel

When a device is selected in the tree or in the synoptics its most common attributes and commands will be accessible here.
To visualize more than one element, you can push in the device image to move it to a new window.

The Synoptic

It shows the state of all the devices and its pressure readings; clicking on any device it will be selected in both tree and central panel.
The synoptic can be hidden at any time going to Menu->View->ShowSynoptic , that will reduce the size of the application to fit in small screens.

Plotting area

There are different tabs showing the pressure profiles from all the sectors of the machine and a Trends (taurus) tab that allows to plot any variable from the tree.
The Trends tab will display last values and also archived values if past dates are selected in the "Plot Configuration" dialog.
Attributes can be added to "Trends" tab using righ-click on attribute nodes in the tree or doing drag-and-drop from the attribute names in the central panel.

Device Naming

Device Name Structure

Every device is identified by a code like:

```
DOMAIN / FAMILY / EQUIPMENT_TYPE - [RACK] - Number
```

- Domain is a cabling system: LI, LT, BO*, BT, SR*, FE*, ...
- Family is a cabling subsystem: VC, RF, CT, ...
- Equipment type defines a code for each type of device: Gauge=CCG, Controller=VGCT, ...
- RACK and Number allow the operator to identify a unique device in the machine.

All devices related to the vacuum control system are available in the left-side tree of the application, and can be easily searched.

Searching for a device

The "search" button allows to find any name in the Vacca device list. Any space in the search field acts as a "*" wildcard.

It means that a search like "bo vgct" will return any name containing "bo" and "vgct" codes in the given order (all the gauge controllers of the booster).

Equipment Types Glossary

Those are the names commonly used, although it can be customized to fit any need.

- CCG: Cold Cathode Gauges
- PIR: Pirani Gauges
- VGCT: MKS 937 Gauge Controller
- IP: Ion Pump
- IPCT: Varian DUAL Ion Pump Controller
- SPBX: Ion Pump Splitter Box unit
- PNV: Pneumatic Valve (Booster/FrontEnds)
- SPNV: Pneumatic Valve (Storage Ring)
- EPS-PLC: Equipment Protection System PLC
- RGA/MicrovisionIP: Residual Gas Analyzer
- Modbus/Serial/Socket: Communication devices

- Composer/ALL/ALL-IPs: Control system devices calculating pressure profiles for each sector
- Alarms: Alarm servers used to send emails/SMS or perform automatic actions on certain conditions.

Showing attributes and archived values

- Clicking on a device in the tree or in synoptic panel will show the device status in the central panel.
- Most common attributes and commands from the device will appear in two tabs.
- Other attributes can be obtained by clicking with the right button over the device name in the tree.
- The menu option "Show Attributes" will display the available attributes.

The attribute values history can be visualized in two ways:

- Dragging the attribute name from the tree to the Taurus Trend tab, if the attribute is readable it will be plot in the graph.
- Right button on the attribute name node in the tree will display the "Show History" action if the attribute is archived. That action will show archived values in a table and will allow to export them into .csv.

Setup of Vacca

Installing

Get vacca and vaccagui modules from sourceforge:

> *svn co* *https://svn.code.sf.net/p/tango-cs/code/tools/vacca/trunk*

Copy vacca folder to a folder in your PYTHONPATH
Copy ./vaccagui script to a folder in your PATH
Copy default.py.ini to default.py and edit your default options (follow comments in the file)

The idea is to have general config in default.py (ORGANIZATION_LOGO, EXTRA_APPS, ...) and use another file to customize to your beamline/system.
Copy default.py to my_beamline.py; modify variables like JDRAW_FILE, COMPOSER, GRID, DEVICE, GAUGES, ...
Remove all variables you don't need to modify (if ORGANIZATION_LOG is the same for all do not have it declared in many files)

Running

Launch the gui as:
./vaccagui path/to/mybeamline.py

Path to your file must be absolute or relative from "vacca" module location.
If no config file is passed as argument the application will try to open $VACCA_CONF or $TANGO_HOST.py if exists.

The VACCA_CONFIG environment variable can be used to set the Vacca config file to load.

Editing Vacca default.py file

The widgets that are part of Vacca are defined at 3 levels:

- **vacca/config.py** : it configures the **Taurus GUI** creating the widgets that are *permanent* in Vacca. Those widgets are created from those variables defined in the next configuration files. Therefore, **config.py doesn't need to be modified**.
- **vacca/default.py** : it defines the default value to all vacca instances. Typically you will add here your logo, icons and default tools to be added to Vacca toolbars; and then you will use a **local_configuration_file.py** to customize vacca to each beamline or subsystem.

> *In case you don't need different configurations you can use directly default.py to customize vacca.*

- **vacca/local_configuration_file.py** : customization of vacca to your beamline / subsystem. Define your synoptic, device tree, default device, attributes in trend, etc ...

default.py.ini

default.py.ini is a configuration file example of how to specify system-wide defaults for VaccaGUI

After copying it to default.py this configuration file determines the default, permanent, pre-defined
options for the GUI; modify them to match your default gui options.

Every CONFIG file loaded by vaccagui will override those options redefined on it;
use default.py for your common options and whatever.py for your in-place customization.

Several methods are imported from **fandango** module to allow easy use of caseless regular expressions while searching in the Tango database:

```
from fandango import matchCl,searchCl,replaceCl,CaselessDict,CaselessList,get_matching_devices,get_matching_attributes
```

A typical usage of **fandango** methods for searching devices:

```
#This will return all valve-related devices but excluding Tango admin devices
all_valves = [d for d in get_matching_devices('*(pnv|valve)*') if not searchCl('dserver',d)]
#This will return all ion pump, cold cathode or pirani attributes if name matches P1,p2,P3. or pressure.
all_pressures = get_matching_attributes('*/*/(ip|ccg|pir)*/(pressure|p[0-9])')
```

Logo and title

```
#ALL these variables can be re-defined in CONFIG FILE
GUI_NAME = 'VACCA'
WDIR = imp.find_module('vacca')[1]+'/'
```

```
URL_HELP = 'http://www.tango-controls.org/Members/srubio/vacca'
URL_LOGBOOK = 'http://logbook.cells.es/'
VACCA_LOGO = WDIR+'image/icons/nggshow.php.png'
ORGANIZATION_LOGO = WDIR+'image/icons/AlbaLogo.png'
```

Synoptic

```
#Synoptic file
JDRAW_FILE = '' #WDIR+'%s/%s.jdw'%(TARGET,TARGET)
#Enables/disables loading jdraw objects into tree
JDRAW_TREE = True
#A method that does transformation on signals sent to other widgets.
JDRAW_HOOK = None #lambda s: apply_transform(get_coordinates(s))
# Enable multiple selection of objects in Synoptic
#from taurus.qt.qtgui.graphic import TaurusGraphicsScene
#TaurusGraphicsScene.ANY_ATTRIBUTE_SELECTS_DEVICE = True
```

Setup of the tree

```
#Domain/Target used to generate grids/trees ; domain may be regexp, target should not
DOMAIN = 'BL*'
TARGET = DOMAIN
USE_DEVICE_TREE = True

#Devices not in JDraw or regular expression to be added to the tree
EXTRA_DEVICES = [DEVICE] #map(bool,set(['%s/VC/ALL'%TARGET,'%s/CT/ALARMS'%TARGET,DEVICE,COMPOSER]))

#Custom tree branches are built using nested dictionaries and regular expressions (if empty a jive-like tree is built).
CUSTOM_TREE = {}
# {'CT':'BL*(CT|ALARMS|PLC)$',
#  'FE':'FE*/VC/*',
#  'Valves': {'.OH':'*OH/PNV*',
#             'EH01':'*EH01/PNV*',},
#  'BAKEOUTS':'BL*(BAKE|BK)*',}
```

Device Panel setup

```
#PyStateComposer to get Vacuum Profile curves
COMPOSER = '' #'%s/vc/all'%DOMAIN
```

```
#Default device to appear in the DevicePanel
DEVICE = 'sys/tg_test/1'
USE_DEVICE_PANEL = True
PANEL_COMMAND = 'taurusdevicepanel --config-file='+WDIR+'default.py'
```

Setup of Attribute / Icon filters for Panel and Tree

The AttributeFilters and CommandFilters dictionary will control which attributes are shown for each equipment tree in the tree and device panel.

Key of the dictionary should be part of the device member, you can distribute attributes or commands in several tabs if you use a list of tuples.

```
#Examples of Attribute filters to be applied to DevicePanel
AttributeFilters = CaselessDict('V-PEN': ['pressure','channelstatus','controller'],)
AttributeFilters['EPS']=[ #You can distribute attributes in different tabs using tuples
    ('Status',['_READY','OPEN_','CLOSE_']),
    ('Signals',['.*_PT.*','was_','paas_','*RGA*']),
    ]
CommandFilters = CaselessDict('V-PEN': (('on',()),('off',())),('setMode',('START','PROTECT')),) #Second argument of tuple is the li
```

The IconMap will link each equipment type to the icon to be shown in tree and panel. It is recommended to use .gif files with transparent background.

```
IconMap = CaselessDict('v-pen':WDIR+'image/equips/icon-pen.gif') #Will be used in both panel and tree
```

```
## Optional:
## If you put filters in a separate file you can load a taurusdevicepanel.py with --config-file=filters.py option
## Then replace previous lines by:
# from vacca.filters import *
```

Plot Setup

```
#Pressure values to be added to the trend
GAUGES = [] #['bl13/vc/vgct-01/p1','bl13/vc/vgct-01/p2']
#sorted(fandango.get_matching_attributes('%s/*/vgct*/p[12]'%DOMAIN.replace('BL','(BL|FE|ID)')))
```

Grid setup

```
#Grid showing all pressures in rows/columns
```

```
GRID = {
    'column_labels': '',
    'delayed': False,
    'frames': False,
    'model': ['%s/VC/(IPCT|VGCT|CCGX)*/(P[12]|Pressure|State)$'%DOMAIN,'%s/V-[^/]*/[0-9]*/(Pressure|State)'%DOMAIN],
    'row_labels':'VcGauges(mbar):(VGCT|PEN), IonPumps(mbar):(IPCT|VARIP)',
    }
```

Setup of external tools

```
#Extra widgets to appear in the NewPanel dialog
EXTRA_WIDGETS = [] #('vacca.VacuumProfile',WDIR+'image/ProfilePlot.jpg'),
EXTRA_PANELS = [] #('vacca.VacuumProfile',WDIR+'image/ProfilePlot.jpg'),
TOOLBARS = [] #[(name,modulename.classname)]

#============================================================================
# Define which External Applications are to be inserted.
# To define an external application, instantiate an ExternalApp object
# See TaurusMainWindow.addExternalAppLauncher for valid values of ExternalApp
#============================================================================

from taurus.qt.qtgui.taurusgui.utils import PanelDescription, ExternalApp, ToolBarDescription, AppletDescription

xvacca = ExternalApp(cmdargs=['konqueror',URL_HELP], text="Alba VACuum Controls Application", icon=WDIR+'image/icons/cow-tux.png')
xtrend = ExternalApp(cmdargs=['taurustrend','-a'], text="TaurusTrend")
xjive = ExternalApp(cmdargs=['jive'], text="Jive")#, icon=WDIR+'image/icons/cow-tux.png')
xastor = ExternalApp(cmdargs=['astor'], text="Astor")#, icon=WDIR+'image/icons/cow-tux.png')
#logbook = ExternalApp(cmdargs=['konqueror %s'%URL_LOGBOOK], text="Logbook", icon=WDIR+"image/icons/elog.png")

#============================================================================
# Define custom applets to be shown in the applets bar (the wide bar that
# contains the logos). To define an applet, instantiate an AppletDescription
# object (see documentation for the gblgui_utils module)
#============================================================================

#Each Applet can be described with a dictionary like this:
#   (name, classname=None, modulename=None, widgetname=None,
#       sharedDataWrite=None, sharedDataRead=None, model=None, floating=True, **kwargs)
#For ExternalApp/VaccaActions use:
# {'$VarName':{'name':'$AppName','classname':'VaccaAction','model':['$Test','$/path/to/icon.png','$launcher']}}

EXTRA_APPS = {
    #'xrga':{'name':'RGA','classname':'VaccaAction','model':['RGA',WDIR+'image/equips/icon-rga.gif']+['rdesktop -g 1440x880 ctrga01
```

```
    }

#from vacca.panel import VaccaAction

xmambo = AppletDescription('Mambo',classname = 'vacca.panel.VaccaAction',model=["Archiving",WDIR+'image/icons/Mambo-icon.png','mamb
xalarms = AppletDescription('Alarms',classname='vacca.panel.VaccaAction',model=['Alarms',WDIR+'image/icons/panic.gif','panic'])
xsnap = AppletDescription('xSnap',classname='vacca.panel.VaccaAction',model=['Snap',WDIR+'image/icons/Crystal_Clear_app_kedit.png',
```

vacca-mistral-2012.jpg (327.723 KB) Sergi Rubio, 17/10/2013 16:17

VaccaBanner.jpg (17.441 KB) Sergi Rubio, 18/10/2013 18:08