GDA QuickStart Guide

Release 9.21

Diamond Light Source

Jul 16, 2021

CONTENTS

Contents:

CHAPTER

ONE

GDA QUICKSTART GUIDE

1.1 Downloading and expanding the zip file ready for compilation

1. Browse http://www.opengda.org/downloads/gda/8.14/, and download `GDA_snapshot_workspace_rnnnnn.zip` (*nnnnn* will be a subversion revision number) (356MB, sorry!)

2. Unzip the downloaded `GDA_snapshot_workspace_rnnnnn.zip` (this will create a directory `GDA_release_8.14/`)

   Here's how to do it from the (Linux) command line:

```
cd ~                                                                    # change to the parent
rm -rf GDA_release_8.14/                                                # delete any previous a
export http_proxy="http://<proxy-url>:<proxy-port>"                     # define proxy (for wge
wget --tries=1 --timeout=5 --no-cache "http://www.opengda.org/downloads/gda/8.14/GDA_snapshot
unzip -q GDA_snapshot_workspace_*.zip                                   # unzip the archive (cr
rm GDA_snapshot_workspace_*.zip                                         # once you are sure eve
```

3. Check that that contents of `GDA_release_8.14/` are as follows:

```
$ ls -A1 GDA_release_8.14/
builder
builders.dawn
example-config
features
GDA_snapshot_workspace_r39673.zip                                       # (it's ok if this has
.metadata
plugins
thirdparty
```

1.2 Setting up the Eclipse workspace

1. GDA 8.14 depends on Java 6 (it has been tested with update 24). Ensure `JAVA_HOME` is set appropriately so that `$JAVA_HOME/bin/java` exists.

2. Start your Eclipse IDE, specifying the new directory `GDA_release_8.14/` as your workspace

3. **Setup PyDev Interpreter** If you have PyDev installed then due to the PyDev natures of certain projects in GDA then when you turn Automatic Build on in the next step you may see error messages of the form:

```
Invalid interpreter: Jython2.5.1 configured for project: uk.ac.gda.core.
```

   To remove these select 'Windows | Preferences | PyDev | Interpreter - Jython' and create a new interpreter:

```
Interpreter Name = Jython2.5.1
Interpreter Executable = <ParentFolder>/GDA_release_8.14/plugins/uk.ac.gda.libs/jython2.5
```

4. Turn automatic building on, by checking Project → Build Automatically.

This will allow you to build the various plugins, features and products for the different GUIs as well as the different java projects used by the server. Note that the example product is defined in `GDA_release_8.14/uk.ac.gda.example.feature/uk.ac.gda.example.product`. The actual devices and scripts for a particular installation are stored in the configuration project `GDA_release_8.14/example-config`.

1.3 Starting the server processes

```
cd GDA_release_8.14/example-config
./bin/gdaservers
```

This will produce a lot of logging output on the console ending with the line:

```
2010-02-24 13:41:42,190 INFO  gda.util.ObjectServer - Server initialisation complete. xmlFile = s
```

When you have finished exploring the client, stop the servers with the commands:

```
./bin/stop_gdaservers
```

1.4 Starting the client process

In production use, the GDA client is build separately and started outside Eclipse. For this QuickStart Guide, we launch the client from within Eclipse.

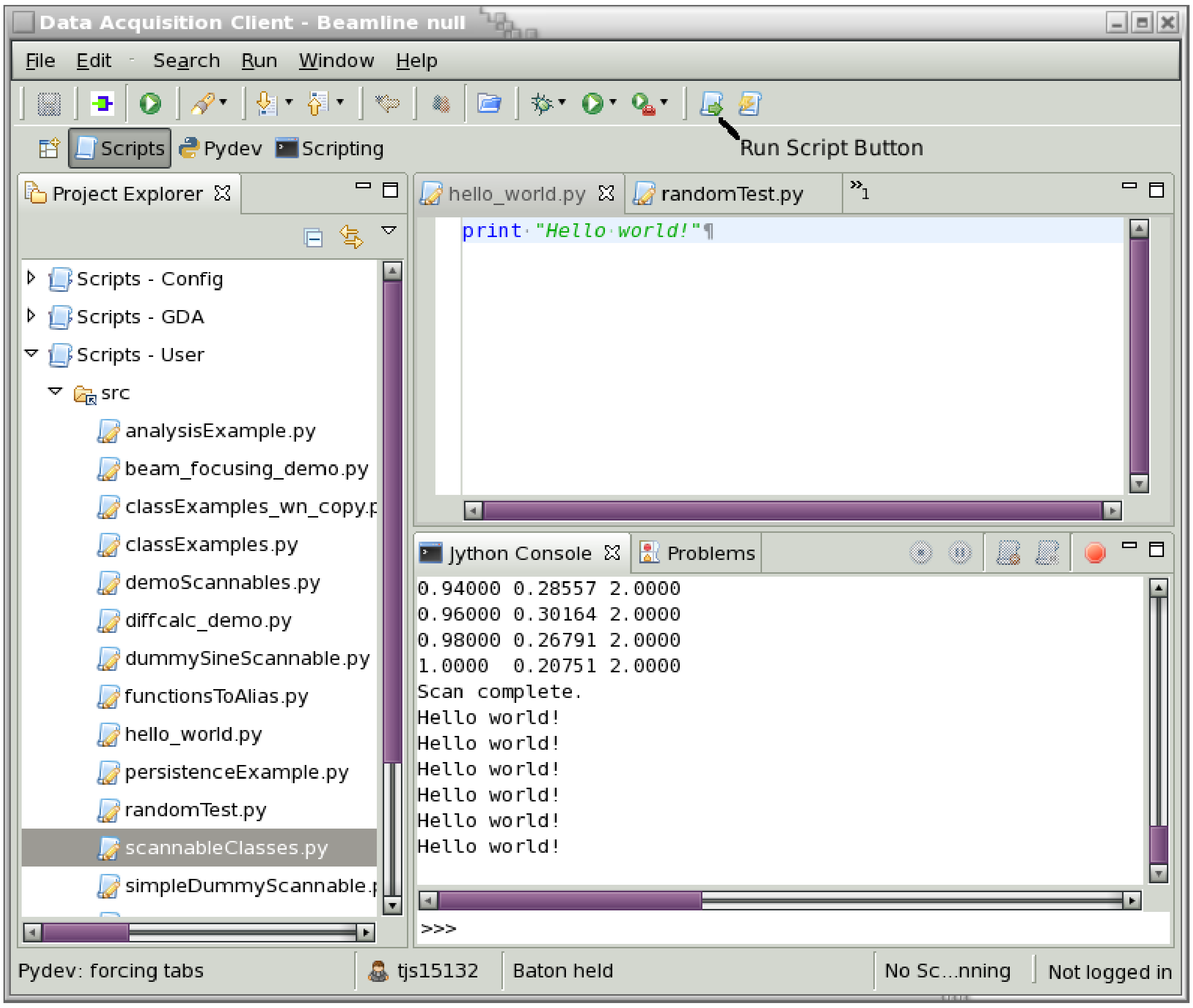
There is a product launch file in feature `uk.ac.gda.example.feature` called `uk.ac.gda.example.product.launch`. This was created using the run configurations dialog with settings:

```
Run a product : uk.ac.gda.example.product

Arguments:
-vmargs
-Dgda.root=${project_loc:uk.ac.gda.core}/../
-Dgda.config=${project_loc:example-config}
-Dgda.data=${project_loc:example-config}/data
-Dgda.var=${project_loc:example-config}/var
-Dgda.logs.dir=${project_loc:example-config}/logs
-Dgda.propertiesFile=${project_loc:example-config}/properties/java.properties
-Djacorb.config.dir=${project_loc:example-config}/properties
-Dgov.aps.jca.JCALibrary.properties=${project_loc:example-config}/properties/JCALibrary.propertie
-Xms256m
-Xmx1024m
-XX:PermSize=128m
-XX:MaxPermSize=256m

Plugins:
    (Use Add Required Plugins)
    Add uk.ac.gda.dal manually
```

1. Open the context menu (right-click) for the launch file `uk.ac.gda.example.feature/uk.ac.gda.example.prod` and select *Run As → Run Configurations → uk.ac.gda.example.product*.
2. On the *Plug-ins* tab, click *Deselect All*, then click *Add Required Plug-ins*.
3. Click *Apply* to save your changes.
4. Click *Run* to start the client running (make sure that the servers have been started).

1.5 Quick tour of perspectives and views

1.5.1 Scripts Perspective



```
print "Hello world!"
```

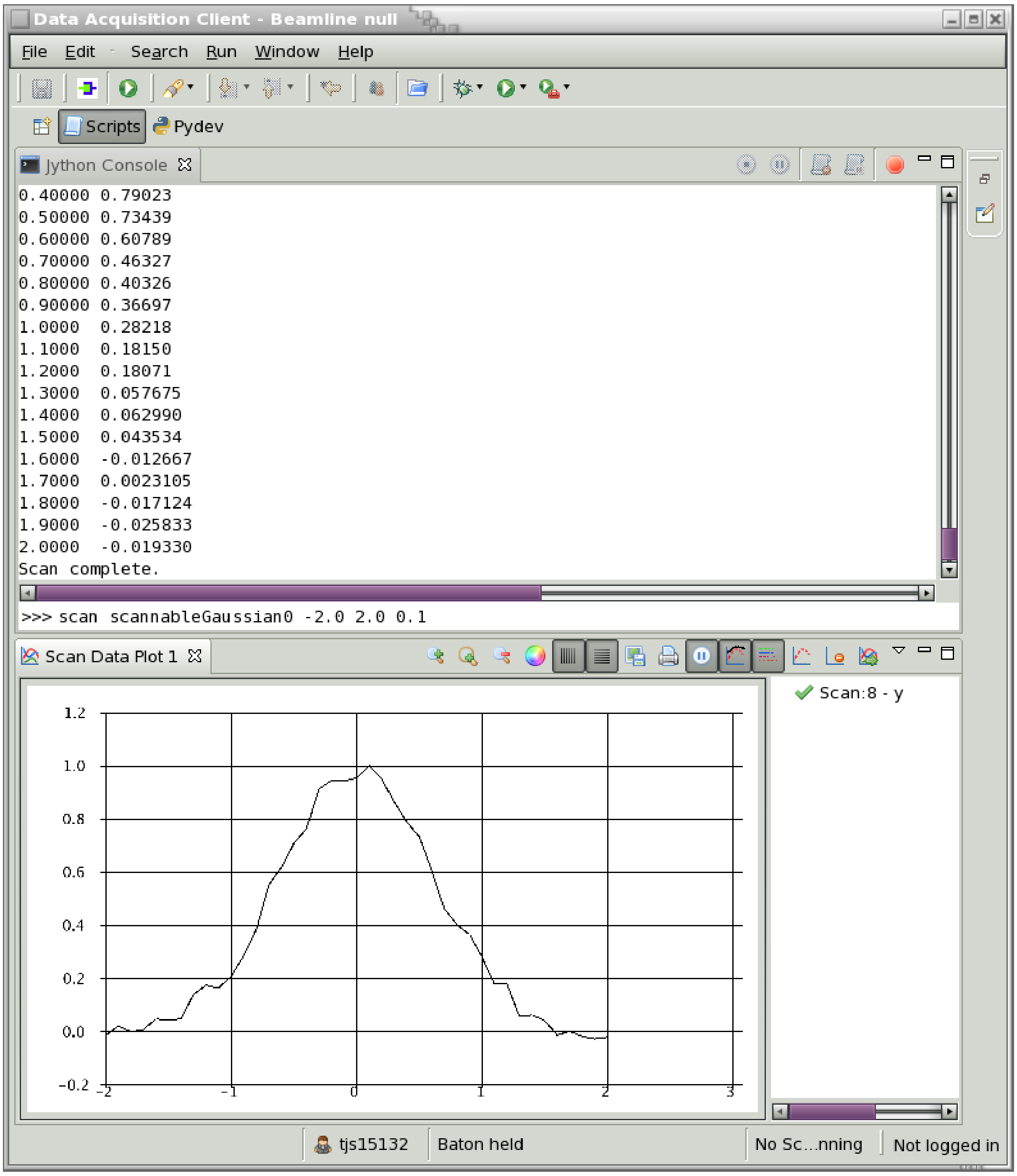
```
0.94000 0.28557 2.0000
0.96000 0.30164 2.0000
0.98000 0.26791 2.0000
1.0000  0.20751 2.0000
Scan complete.
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
>>>
```

The view on the left of the Scripts Perspective shows the various collections of pre-existing scripts. If you do not see the ‘Scripts - GDA’ and ‘Scripts - Config’ projects go to the preferences on the ‘Project Explorer’ and show those two projects.

The editor shows the simple script hello_world.py. To run the script simply press the Run Script toolbar icon . Similarly type jython directly into the input pane of the Console view (marked with >>>).

Open the XYScanPlot view from the collection named ‘Data Acquisition - General’. Then enter the command in the Console view:

```
scan scannableGaussian0 -2.0 2.0 0.1
```


Data Acquisition Client - Beamline null
File Edit Search Run Window Help
Scripts Pydev
Jython Console

```
0.40000 0.79023
0.50000 0.73439
0.60000 0.60789
0.70000 0.46327
0.80000 0.40326
0.90000 0.36697
1.0000  0.28218
1.1000  0.18150
1.2000  0.18071
1.3000  0.057675
1.4000  0.062990
1.5000  0.043534
1.6000  -0.012667
1.7000  0.0023105
1.8000  -0.017124
1.9000  -0.025833
2.0000  -0.019330
Scan complete.
>>> scan scannableGaussian0 -2.0 2.0 0.1
```

1.5.2 Autocompletion in the Jython Editor View

When GDA Client is deployed (i.e. *not* a debug/run session in eclipse as above but built into the deployed product), the Jython Editor is configured to see the compiled GDA classes for auto-completion. It uses a jar file called ‘gda-script-interface.jar’ in the ‘client’ folder.

So then when entering the text:

```
from gda.
```

and pressing CTRL-SPACE, the editor shows the auto-completions (see ‘Autocompletion in the Console View’ below).

In a run/debug session this jar file does not exist. You may set the property ‘gda.debgug.client.interface’ to specify the location of this file or edit the Jython Interpreter directly when using the client. (Note that if you set 'gda.debgug.client.interface' it is only used when the client is first started on a new workspace.)

The jar file 'gda-script-interface.jar' can be generated using the build-main.xml the task 'make_script_interface_jar' does this.

1.5.3 Autocompletion in the Console View

In the input Console view type (with the final dot):

```
from gda.
```

and press CTRL-SPACE. This will bring up a list of possible completions. Select configuration and press enter. Type a further `.` and press CTRL-SPACE again to reveal further completions; select properties and press enter. Complete the command so that it becomes:

```
from gda.configuration.properties import LocalProperties
```

Now simply type LocalPr and press CTRL_SPACE for autocompletion of LocalProperties to be performed.

1.5.4 Simple Scan, viewing the results and manipulating them in script

Now create a simple 1d scan using the commands:

```
from scannableClasses import *
ss = ScannableSine("ss",0.0, period=0.5)
scan ss -4. 4. .05
```

The contents of a data file can be read into an object called a ScanFileHolder. To get the name of the name of the last file created programmatically and use that to load the data in a ScanFileHolder enter the commands:

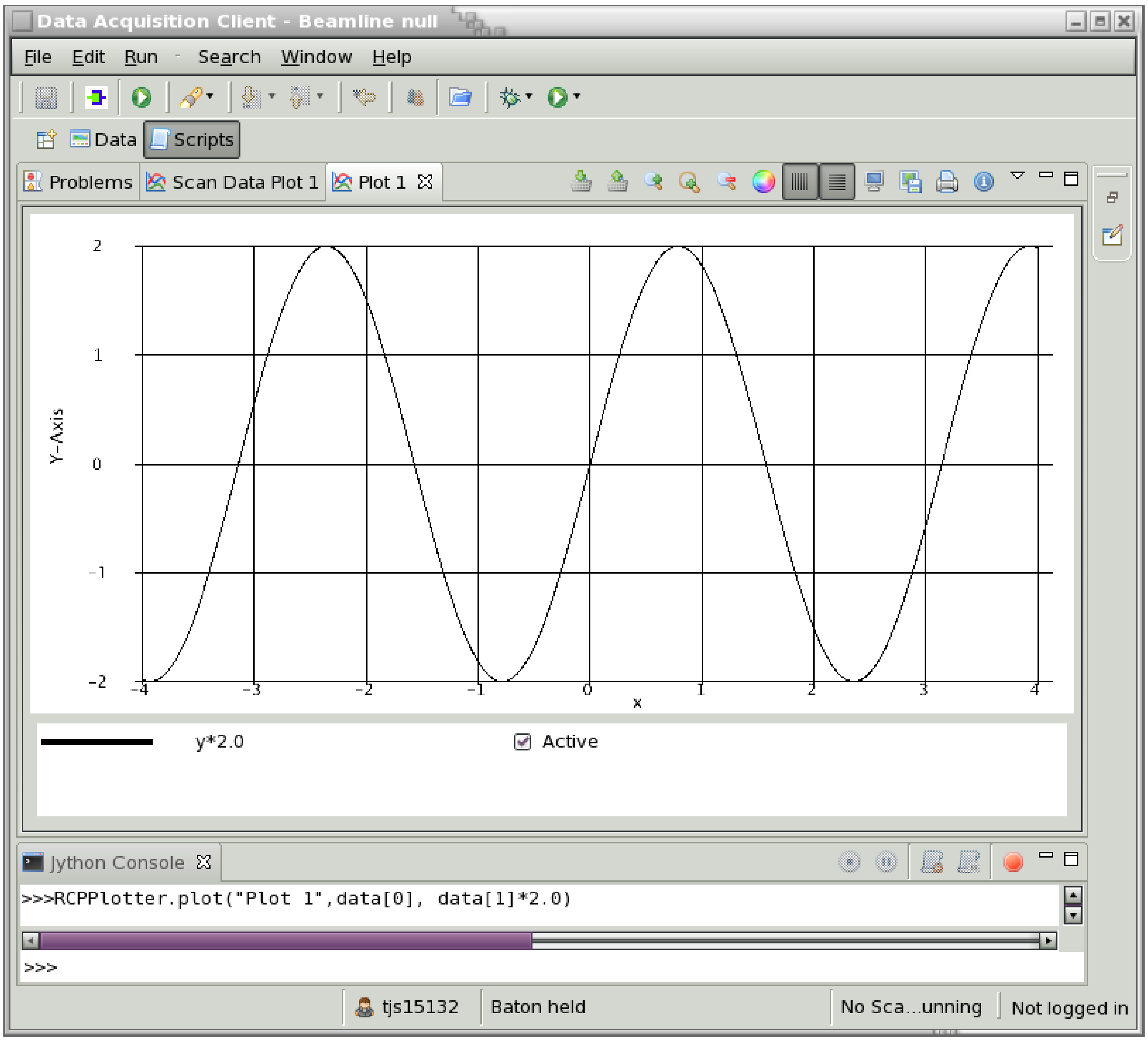
```
from gda.data import NumTracker
from gda.data import PathConstructor
from gda.analysis.io import SRSLoader
numTracker = NumTracker("tmp")
file = PathConstructor.createFromDefaultProperty()
file = file + "/" + `int(numTracker.getCurrentFileNumber())`+".dat"
data = ScanFileHolder()
data.load(SRSLoader(file))
data.getHeadings()
```

Now open the view called "Plot 1" from the group named 'Data Analysis - General'. To display the data just loaded in this view enter the commands:

```
RCPPlotter.plot("Plot 1",data[0], data[1])
```

You can plot the effects of various mathematical operators on the data such as:

```
RCPPlotter.plot("Plot 1",data[0], data[1]*2.0)
```


Data Acquisition Client - Beamline null
File Edit Run Search Window Help
Data Scripts
Problems Scan Data Plot 1 Plot 1
2
1
0
-1
-2
Y-Axis
-4 -3 -2 -1 0 1 2 3 4
x
y*2.0
Active
Jython Console
tjs15132 Baton held No Sca...unning Not logged in

```
>>>RCPPlotter.plot("Plot 1",data[0], data[1]*2.0)
>>>
```

1.5.5 Nested Scans and XY Plotting

First we want to change the data format to from the ASCII format used above to Nexus. The format is controlled by a property called gda.data.scan.datawriter.dataFormat that can be read and set using the static methods LocalProperties.get and LocalProperties.set.

To get the current value type:

```
LocalProperties.get("gda.data.scan.datawriter.dataFormat")
```

This should return SrsDataFile which indicates that currently data files are written in an ASCII format originating from SRS. Now type:

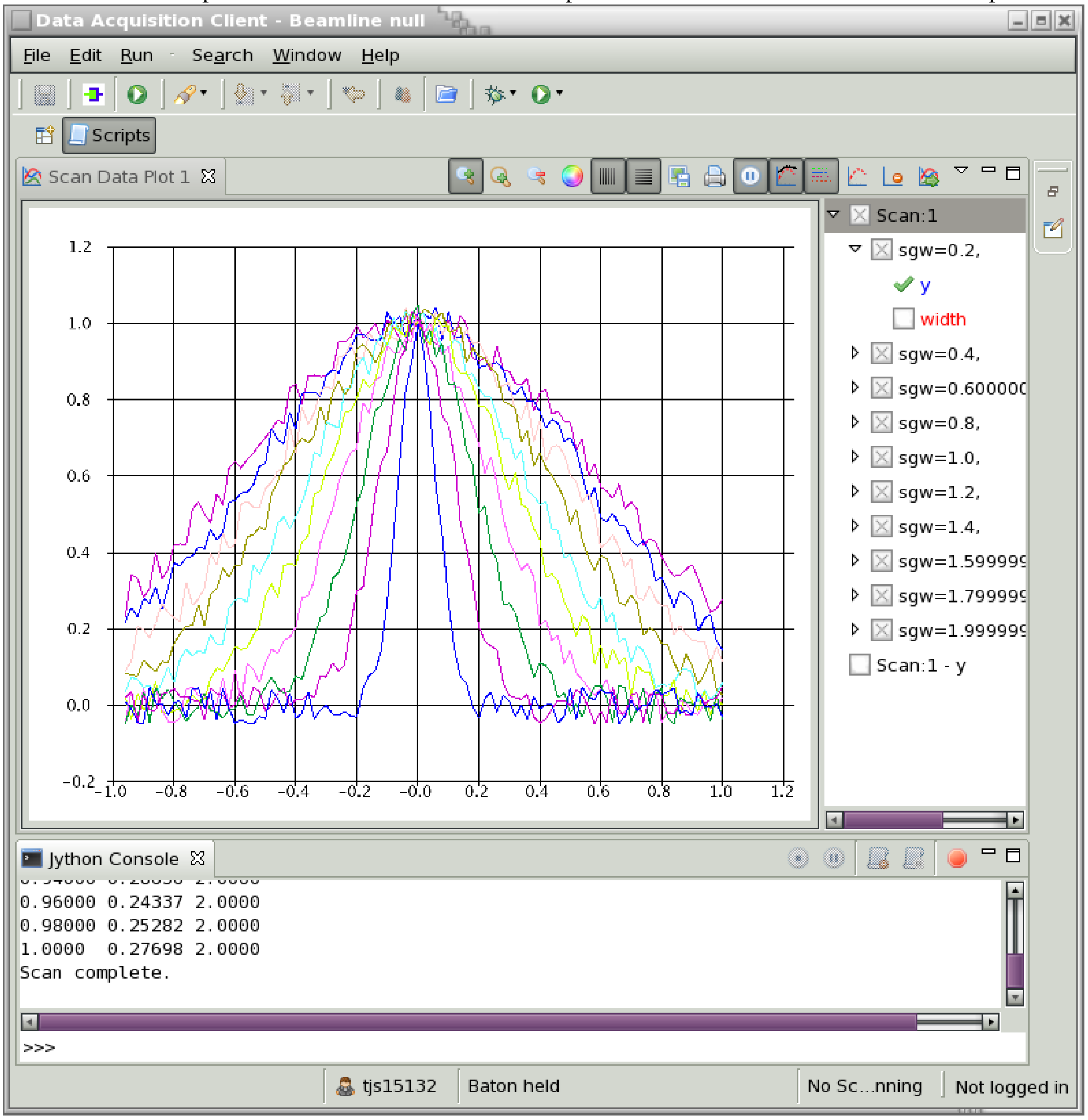
```
LocalProperties.set("gda.data.scan.datawriter.dataFormat","NexusDataWriter")
```

to change the file format to Nexus.

Now to create an interesting multidimensional Nexus file enter the commands:

```
import scannableClasses
from scannableClasses import *
sgw = ScannableGaussianWidth('sgw', scannableGaussian0)
scan sgw 0.2 2.0 0.2 scannableGaussian0 -1.0 1.0 0.02
```

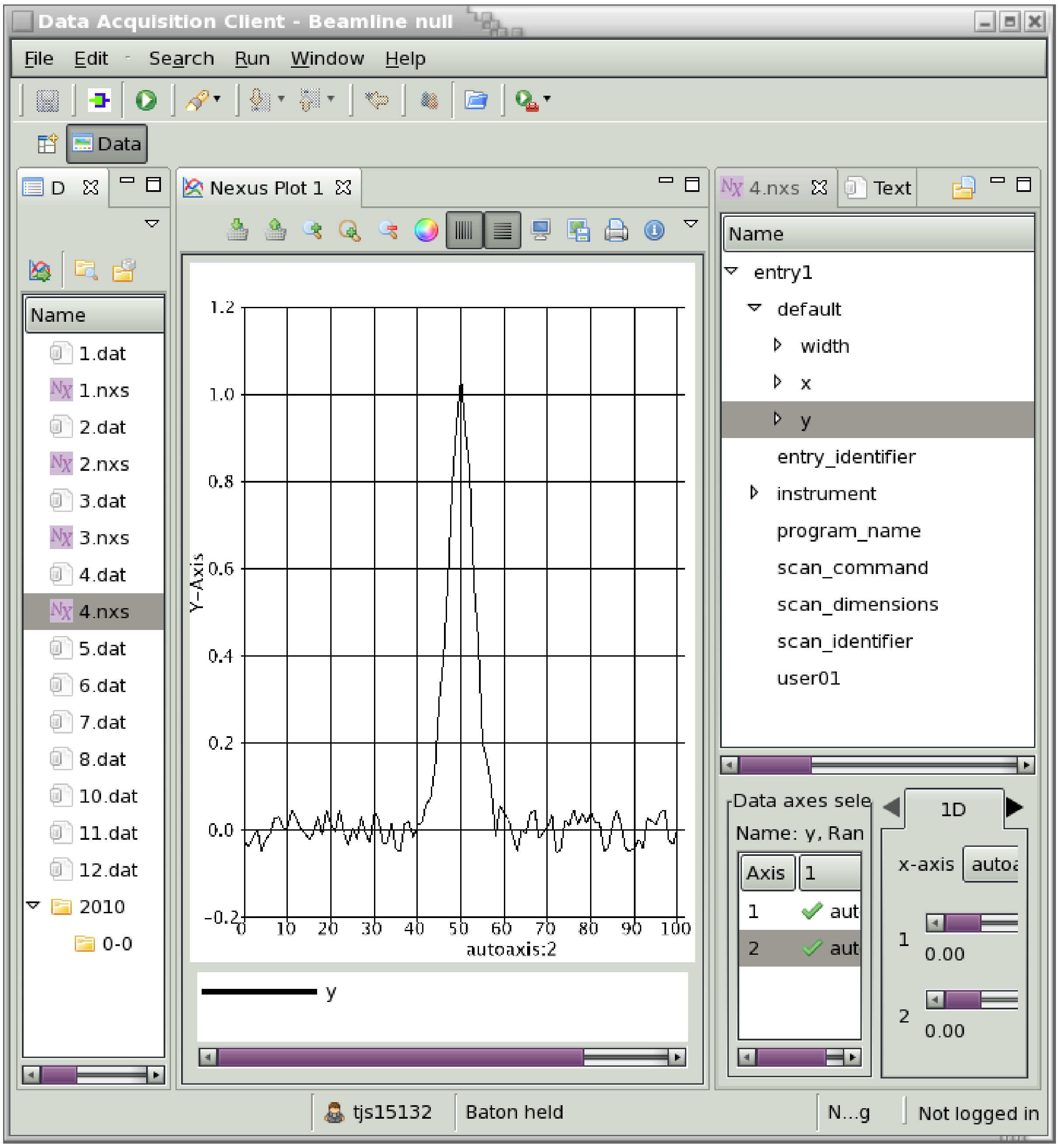
This nested scan has an outer scan which sets the width of the contained scannable Gaussian to different values from 0.2 to 2.0 in steps of 0.2. The inner scannable is then plotted for each width from -1.0 to 1.0 in steps of 0.02

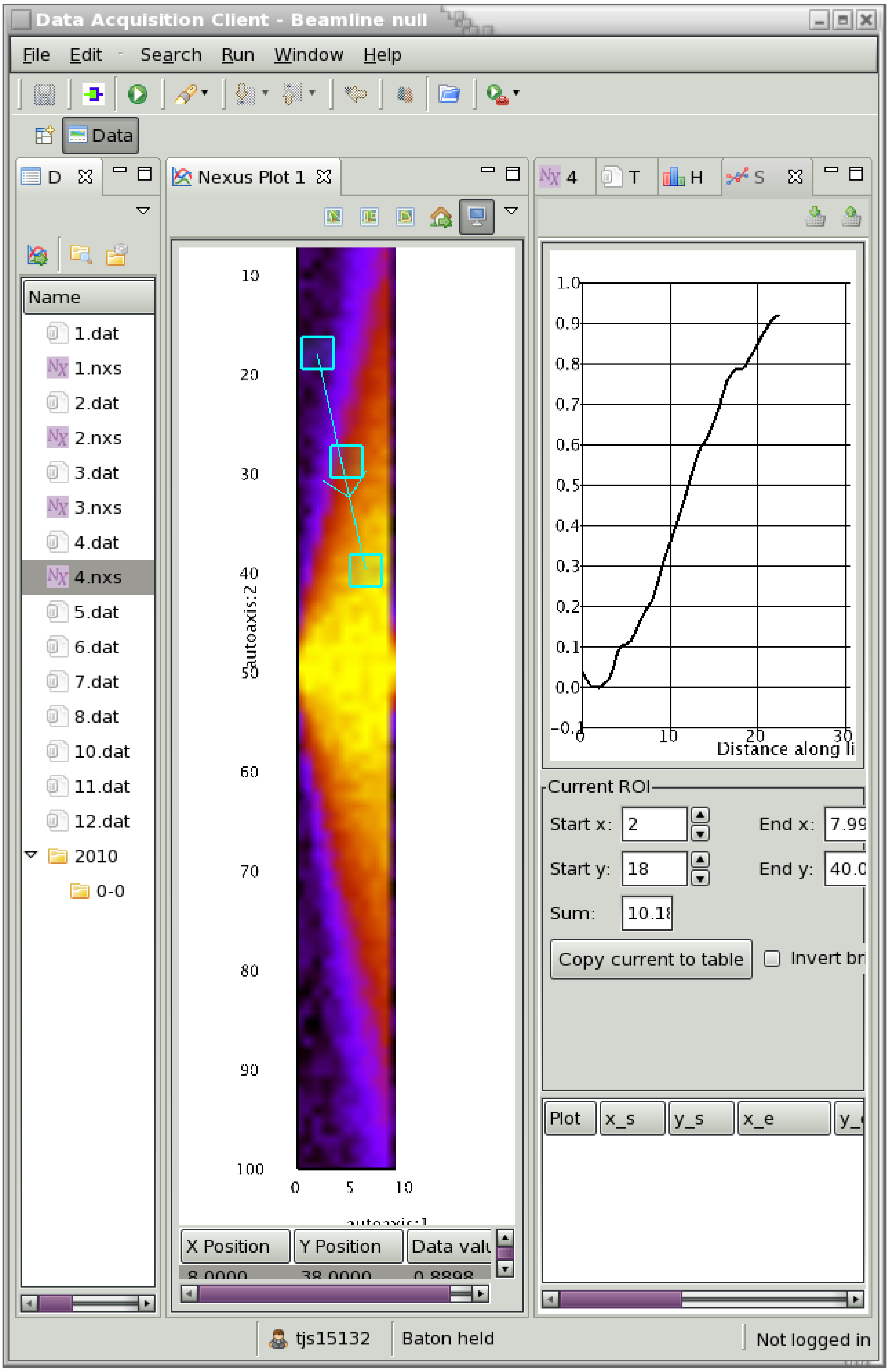


```
0.96000 0.24337 2.0000
0.98000 0.25282 2.0000
1.0000  0.27698 2.0000
Scan complete.
>>>
```

1.5.6 Data Perspective

Open the Data Perspective and select the latest created Nexus file (ends in extension nxs). The structure of the Nexus file can be viewed in the NexusTree view. The selected data , normally within entry1 | default | name, can then be viewed in various ways from 1d to 2d surfaces.


Data Acquisition Client - Beamline null
File Edit Search Run Window Help
Data
D
Nexus Plot 1
4.nxs
Text
Name
1.dat
1.nxs
2.dat
2.nxs
3.dat
3.nxs
4.dat
4.nxs
5.dat
6.dat
7.dat
8.dat
10.dat
11.dat
12.dat
2010
0-0
Y-Axis
autoaxis:2
y
entry1
default
width
x
y
entry_identifier
instrument
program_name
scan_command
scan_dimensions
scan_identifier
user01
Data axes sele
Name: y, Ran
Axis
1
aut
2
1D
x-axis
autoa
0.00
tjs15132
Baton held
N...g
Not logged in

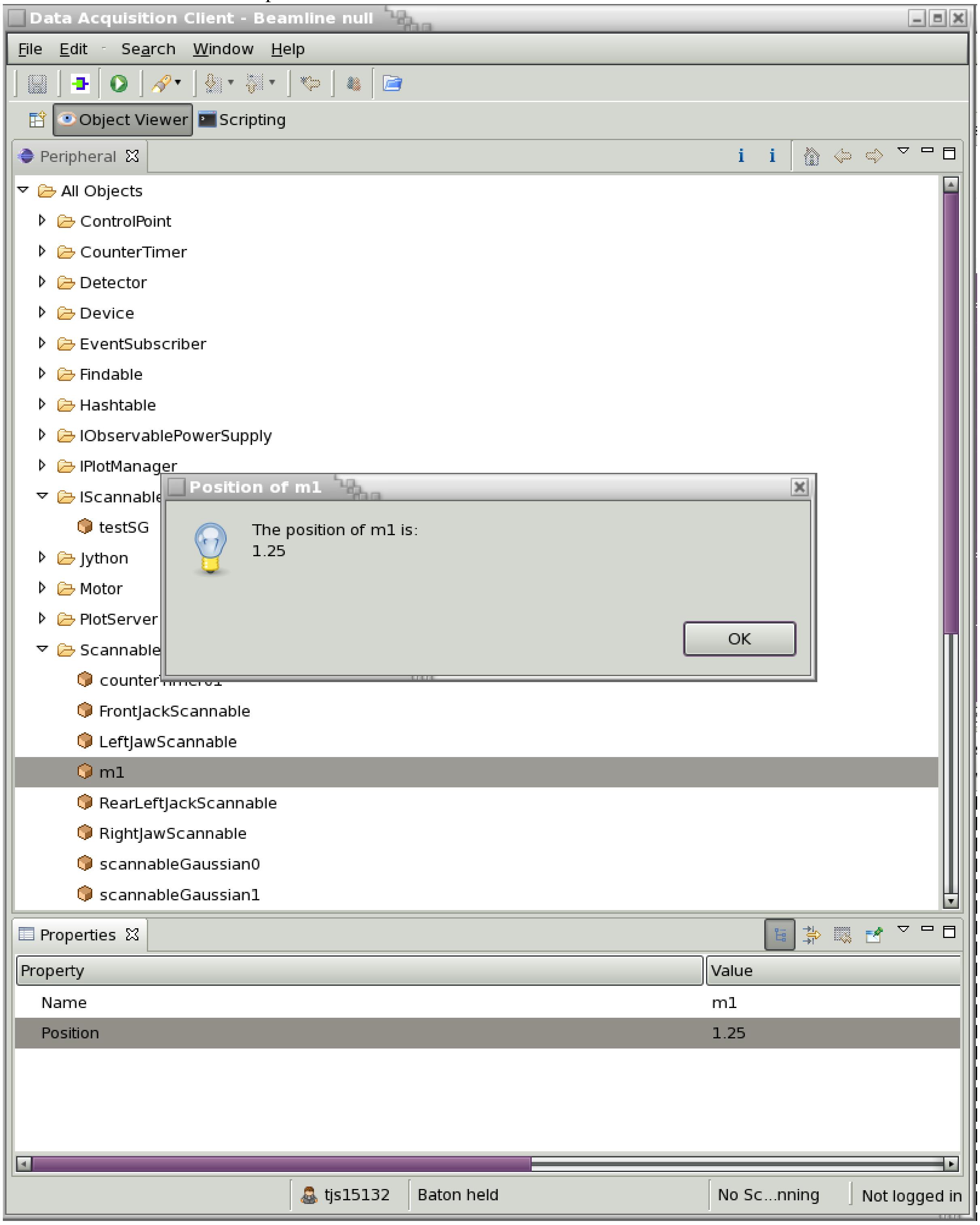


Start x:	2	End x:	7.99
Start y:	18	End y:	40.0
Sum:	10.1		

X Position	Y Position	Data valu
8.0000	38.0000	0.8898

1.5.7 Object Viewer Perspective

The Object Viewer Perspective displays a tree view of the objects on the server grouped by interface, if you select an item its property is shown in the Properties view. The context menus on the tree are item type dependent and can be extended via extension points.



The position of m1 is:
1.25

Property	Value
Name	m1
Position	1.25

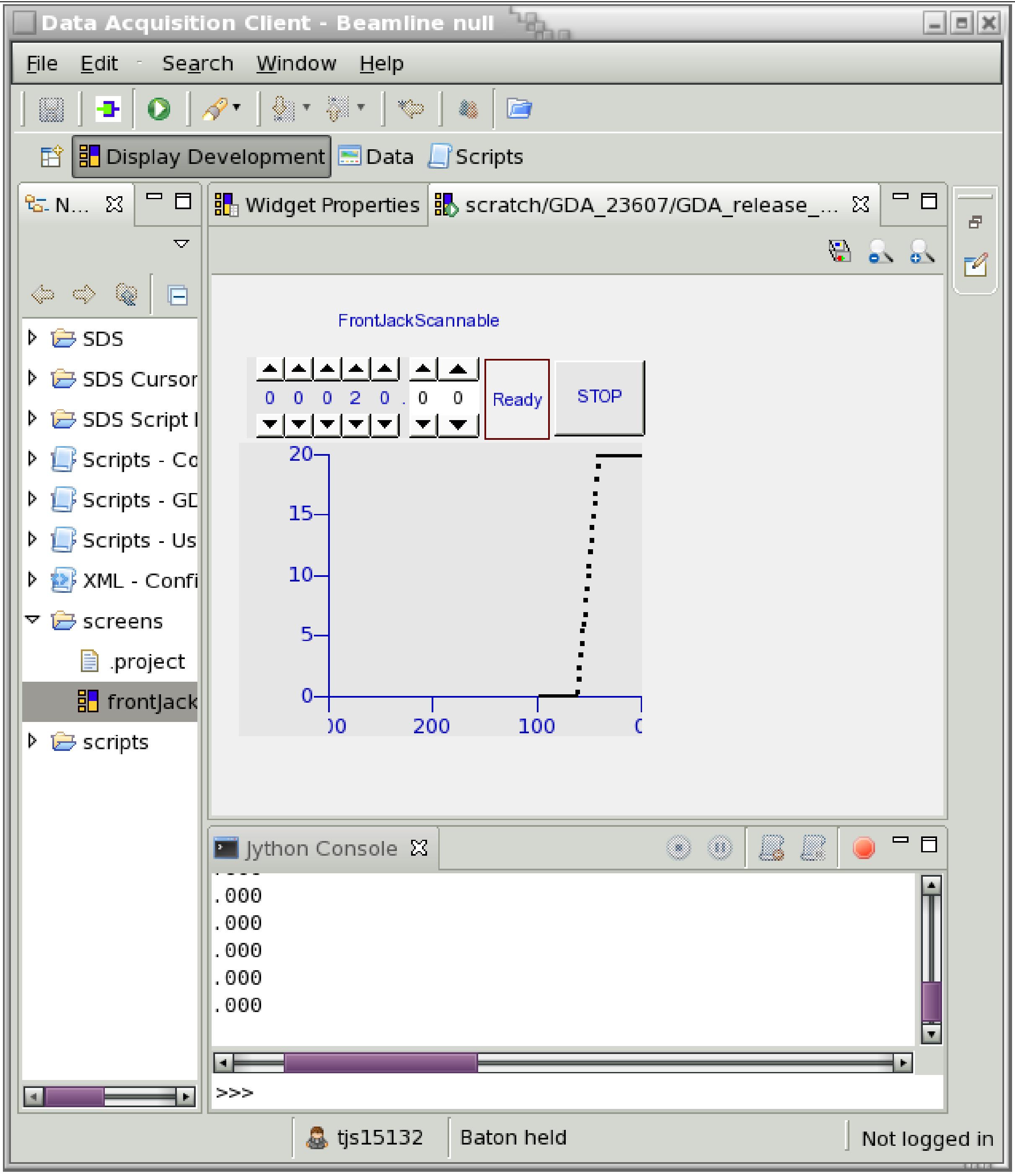
1.5.8 Synoptic Displays

For information on CSS see http://css.desy.de/content/index_eng.html

GDA contains a plugin called uk.ac.gda.dal that contains a DAL to allow GDA scannables to be displayed in a CSS SDS window.

Open the Display Development perspective. Refresh the Navigator view. You should see a project called screens containing a file called frontJackScannable.css-sds which allows monitoring and control of a scannable named FrontJackScannable. (Note that this scannable has a range of 0 to 20). Select this file and open it using the command "Run as View". With the view displayed scan the FrontJackScannable using a command of the form:

```
scan FrontJackScannable 0. 20. 2.
```



```
.000
.000
.000
.000
.000
>>>
```

The view was created using the following instructions:

1. Select the project called screens and add a new Synoptic Display.
2. Add a Linking Container to the view large enough to hold a number of widgets.
3. Add an alias to the linking container:

```
    name = channel
    value = FrontJackScannable
```

4. Set the Primary PV of the linking container to $channel$
5. **Add various widgets to within the linking container and use the Configure** Dynamic Aspects to link the value of FrontJackScannable to the widget properties. For example add a ThumbWheel and configure the dynamic aspect of the value property so that the Value input channel is associated with $channel$.