

---

# **GDA Quick Start Guide**

*Release 8.2*

**Jun Aishima**  
**Mark Basham**  
**Peter Chang**  
**Joachim Diepstraten**  
**Richard Fearn**  
**Matthew Gerring**  
**Paul Gibbons**  
**Karl Levik**  
**Geoff Mant**  
**Vasanthi Nagalingam**  
**Bill Pulford**  
**Eric Ren**  
**Tobias Richter**  
**Duncan Sneddon**  
**Robert Walton**  
**Matthew Webber**  
**Richard Woolliscroft**  
**Fajin Yuan**

March 01, 2010



# CONTENTS

<b>1</b>	<b>Expanding the tarballs ready for compilation</b>	<b>1</b>
<b>2</b>	<b>Setting up the Eclipse workspace</b>	<b>3</b>
<b>3</b>	<b>Starting the server processes</b>	<b>5</b>
<b>4</b>	<b>Starting the client process</b>	<b>7</b>
<b>5</b>	<b>Quick tour of perspectives and views</b>	<b>9</b>
5.1	Scripts Perspective . . . . .	9
5.2	Autocompletion in the Console View . . . . .	10
5.3	Simple Scan, viewing the results and manipulating them in script . . . . .	11
5.4	Nested Scans and XY Plotting . . . . .	12
5.5	Data Perspective . . . . .	13
5.6	Synoptic Displays . . . . .	16



# EXPANDING THE TARBALLS READY FOR COMPILATION

GDA as downloaded from our ftp site consists of 3 tar balls, one containing the GDA plugin projects, one the required third party plugins such as eclipse and other essential items, and the third contains documentation.

1. Put the 3 tar balls into a folder.
2. cd into that folder
3. 

```
>tar -xzf GDA_release_8.2_rxxxxx.tar.gz
>ls GDA_release_8.2
builder COPYING documentation dropins example-config features plugin
```
4. 

```
>tar -xzf GDA_release_8.2_docs_rxxxxx.tar.gz
>ls GDA_release_8.2_docs
GdaAnalysisGuide.pdf GdaDeveloperGuide.pdf GdaInstallationGuide.pdf GdaUserGuide.pdf html
```
5. 

```
>tar -xzf GDA_release_8.2_thirdparty_rxxxxx.tar.gz -C GDA_release_8.2/
>ls GDA_release_8.2/thirdparty
bundles ch.qos.logback.eclipse_1.1.1 CSS-SDS-1.2.0 eclipse pydev
```

*Content of GDA\_release\_8.2/plugins*

- uk.ac.diamond.scisoft.analysis
- uk.ac.diamond.scisoft.analysis.rcp
- uk.ac.gda.client
- uk.ac.gda.client.extensions
- uk.ac.gda.common
- uk.ac.gda.common.rcp
- uk.ac.gda.core
- uk.ac.gda.dal
- uk.ac.gda.epics
- uk.ac.gda.example
- uk.ac.gda.libs
- uk.ac.gda.nexus
- uk.ac.gda.pydev.extension

*Content of GDA\_release\_8.2/features*

- uk.ac.gda.client-feature
- uk.ac.gda.dal-feature
- uk.ac.gda.example-feature

# SETTING UP THE ECLIPSE WORKSPACE

1. GDA 8.2 depends on JDK 1.6 update 16. Ensure JAVA\_HOME is set appropriately so that \$JAVA\_HOME/bin/java exists.
2. Switch to a new workspace
3. Turn Automatic Build off
4. Use “Import Existing Projects into Workspace” to import all projects in GDA\_release\_8.2/plugins, GDA\_release\_8.2/features and the single projects in the folders GDA\_release\_8.2/thirdparty, GDA\_release\_8.2/dropins and GDA\_release\_8.2/example-config.
5. Use Window | Preferences option to set the Plugin Development | Target Platform - select the GDA Build Target defined in the file uk.ac.gda.core/build.target.
6. **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 = <InstallationFolder>/GDA_release_8.2/plugins/uk.ac.gda.libs/jyth
```

7. Turn Automatic Build on

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 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\_8.2/example-config.



# STARTING THE SERVER PROCESSES

1. `cd GDA_release_8.2/example-config`
2. `./bin/GDA_StartServers`

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
```

To stop the servers use the command:

```
#. ./bin/GDA_StartServers --stop
```



# STARTING THE CLIENT PROCESS

There is a product launch file in plugin `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.users=${project_loc:example-config}/users  
-Dgda.propertiesFile=${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.properties  
-Xms128m -Xmx1024m
```

Plugins:

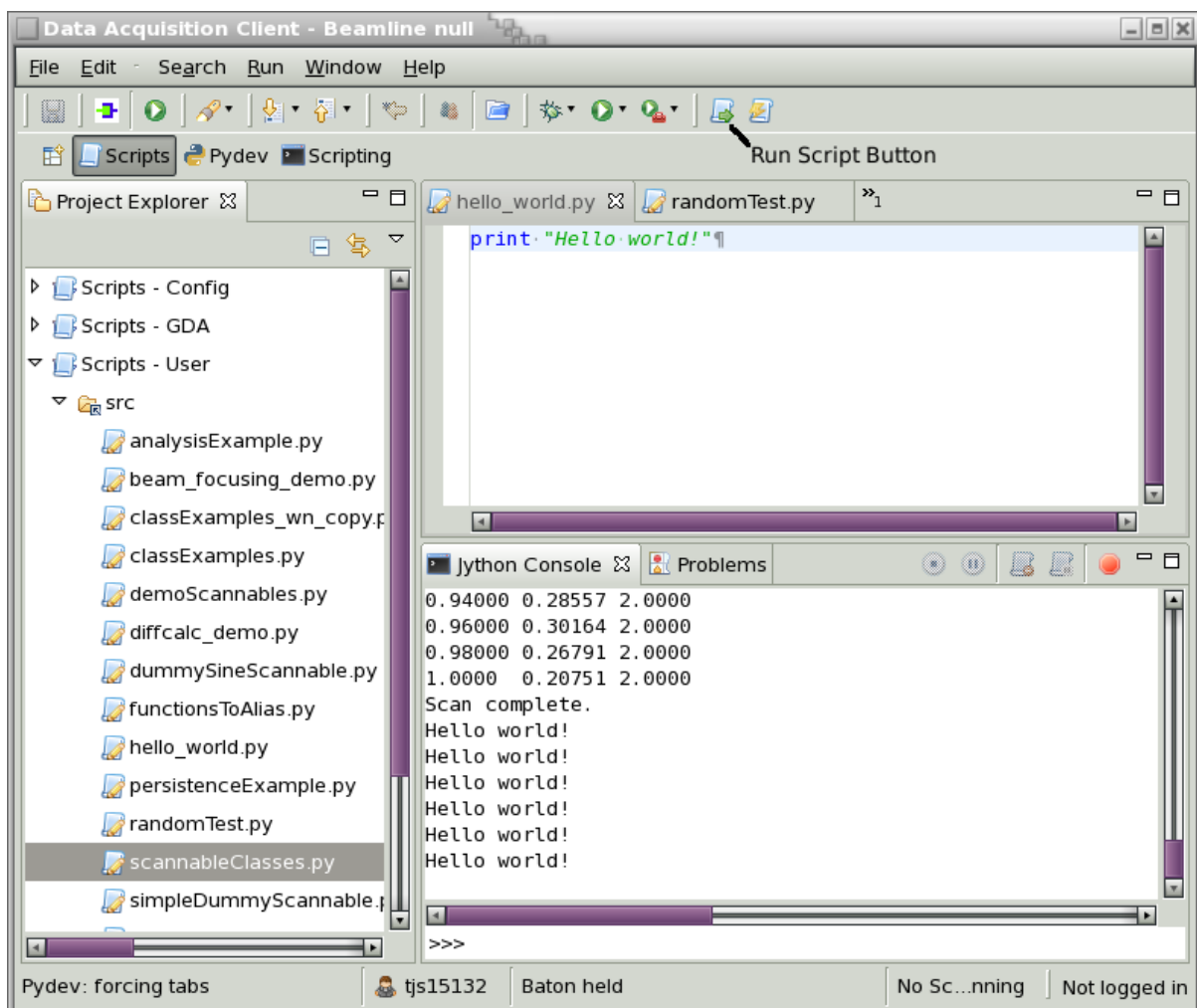
```
(Use Add Required Plugins)  
Add uk.ac.gda.dal manually
```

Open the launch file using the context menu command 'Run As | `uk.ac.gda.example.product`'. This will display a simply login dialog. Press the OK button to display the default perspective.



# QUICK TOUR OF PERSPECTIVES AND VIEWS

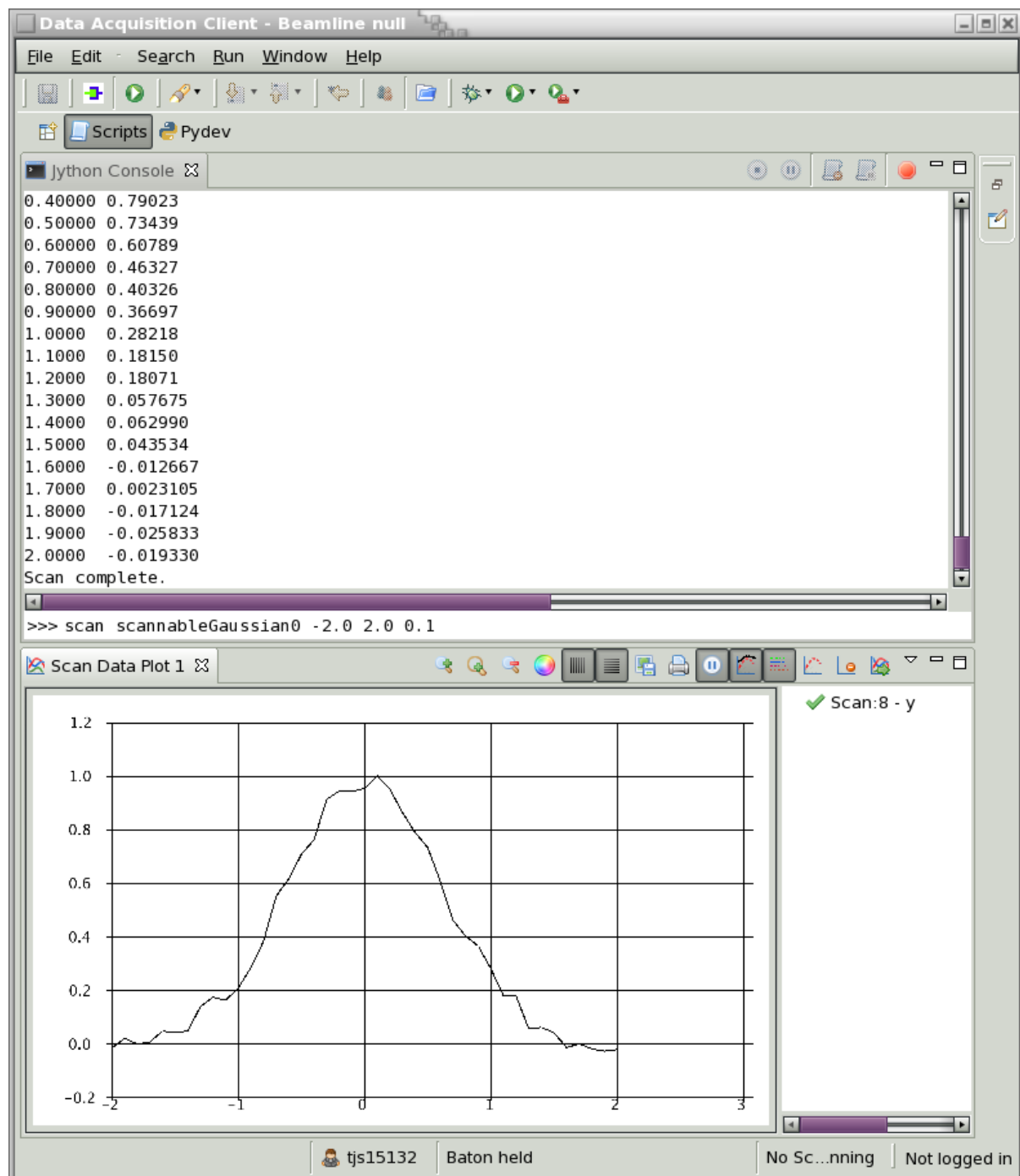
## 5.1 Scripts Perspective



The view on the left of the Scripts Perspective shows the various collections of pre-existing scripts. 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
```



## 5.2 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.

## 5.3 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:

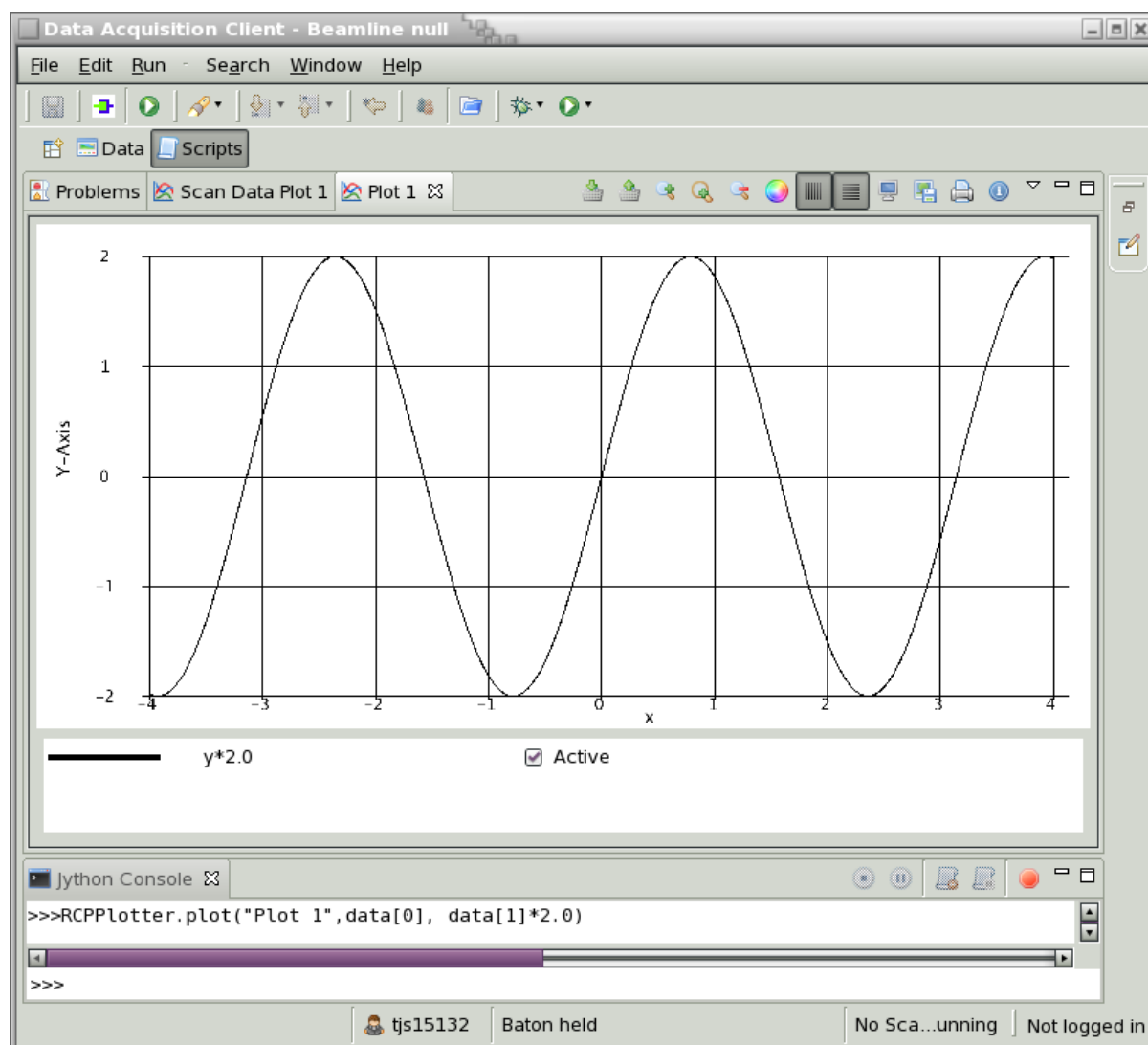
```
from gda.data import NumTracker
from gda.data import PathConstructor
numTracker = NumTracker("tmp")
file = PathConstructor.createFromDefaultProperty()
file = file + "/" + `int(numTracker.getCurrentFileNumber())`+".dat"
data = ScanFileHolder()
data.loadSRS(file)
data.getHeadings()
```

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)
```



## 5.4 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")
```

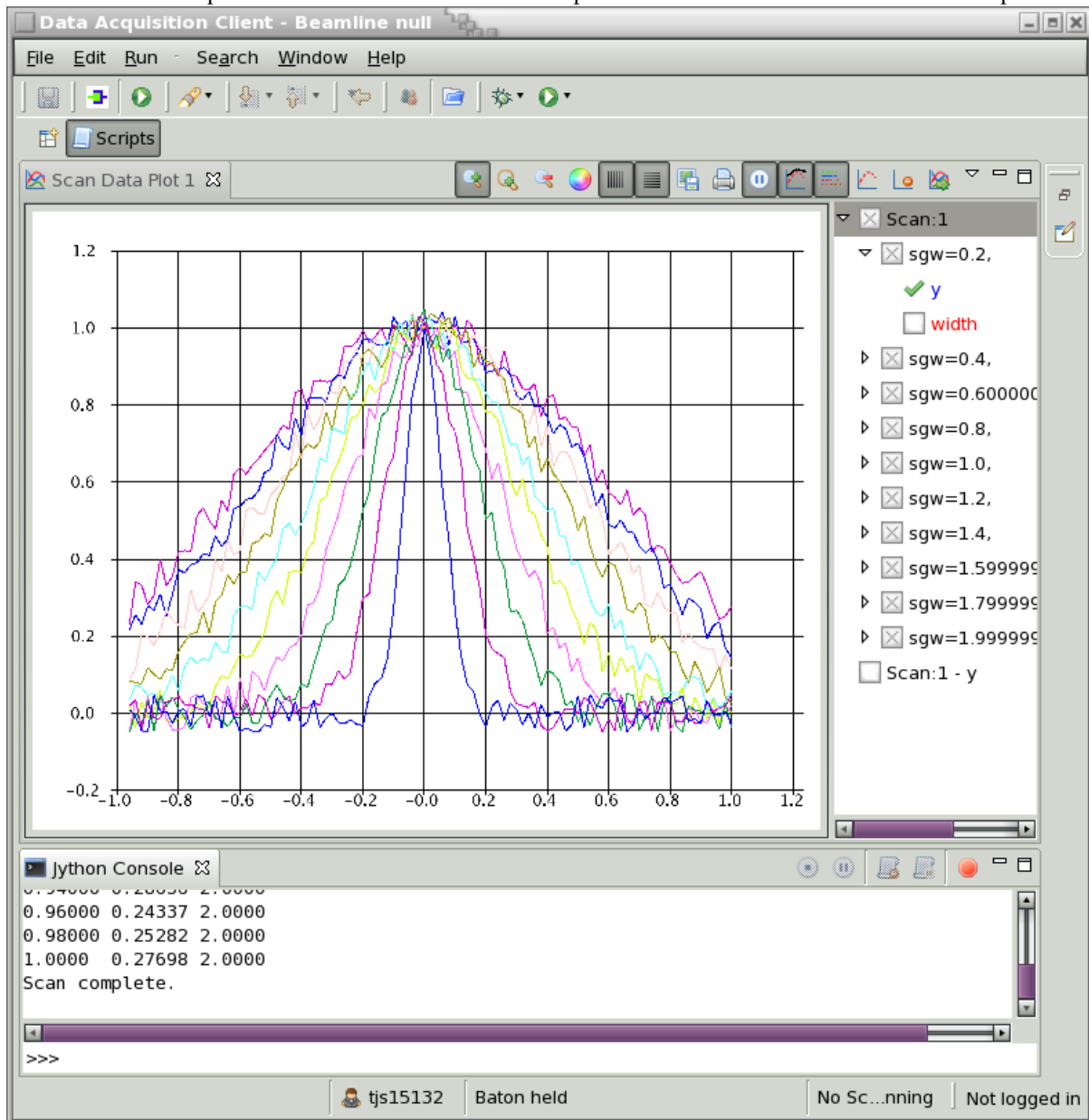
to change the file format to Nexus.

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

```
import scannableClasses
from scannableClasses import *
```

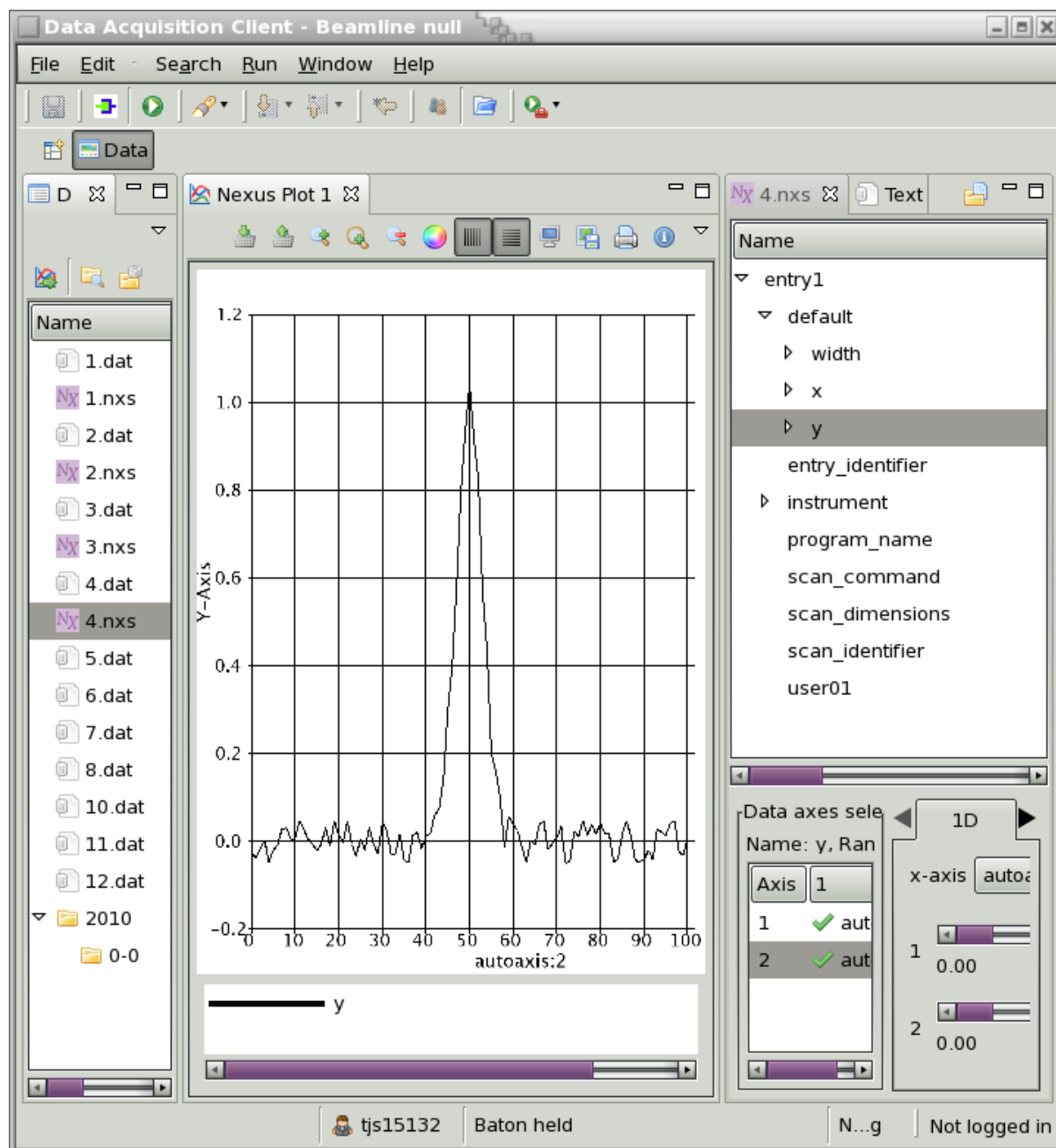
```
sgw = ScannableGaussianWidth('sgw', scannableGaussian0)
scan sgw 0.2 2.0 0.2 scannableGaussian0 -1.0 1.0 0.02
```

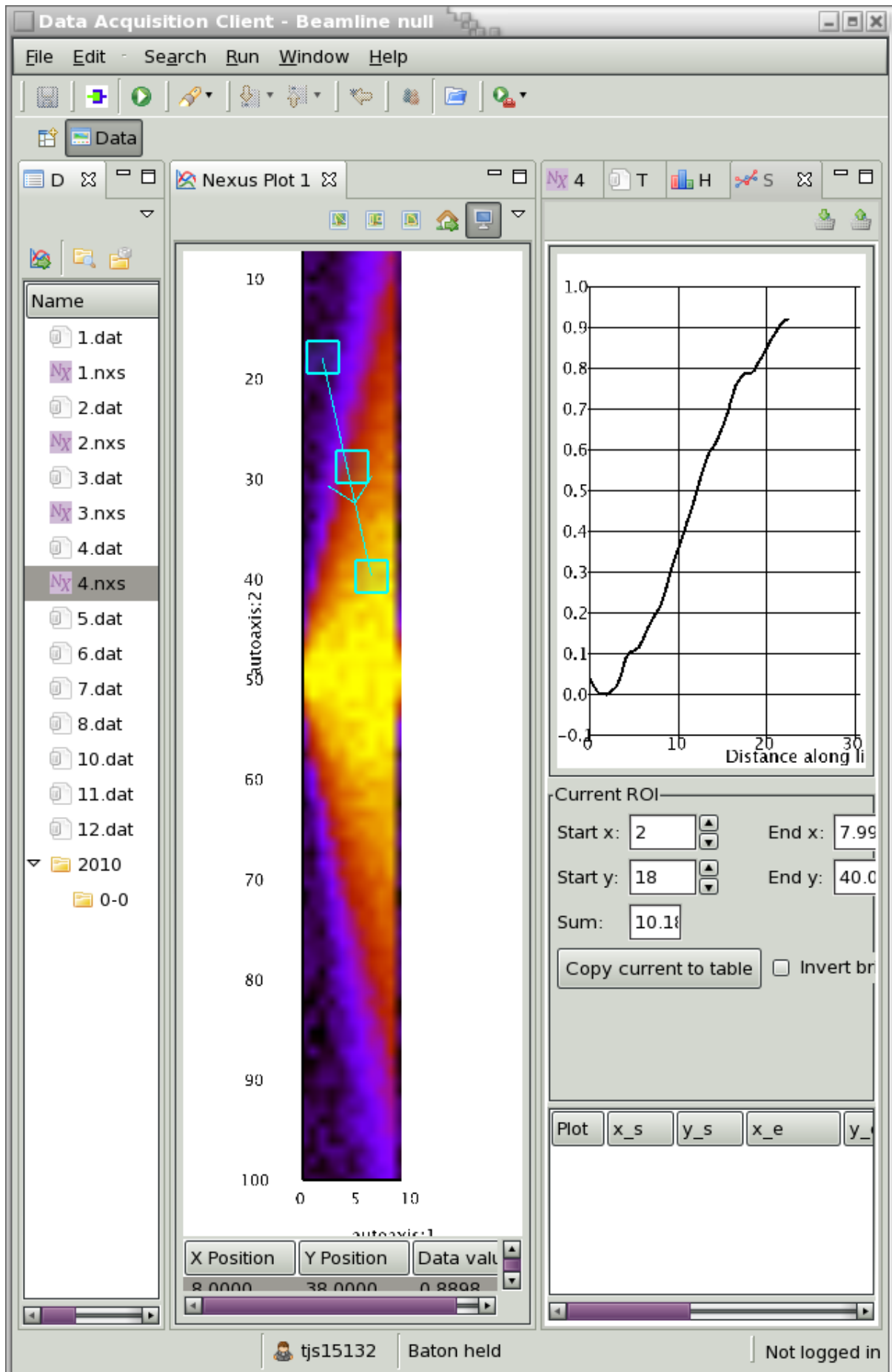
This nested scan has an outer scan which sets the width of the contained scannable Gaussian to different values from 0.2 to 2.0 in steps of 0.2. The inner scannable is then plotted for each width from -1.0 to 1.0 in steps of 0.02



## 5.5 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.





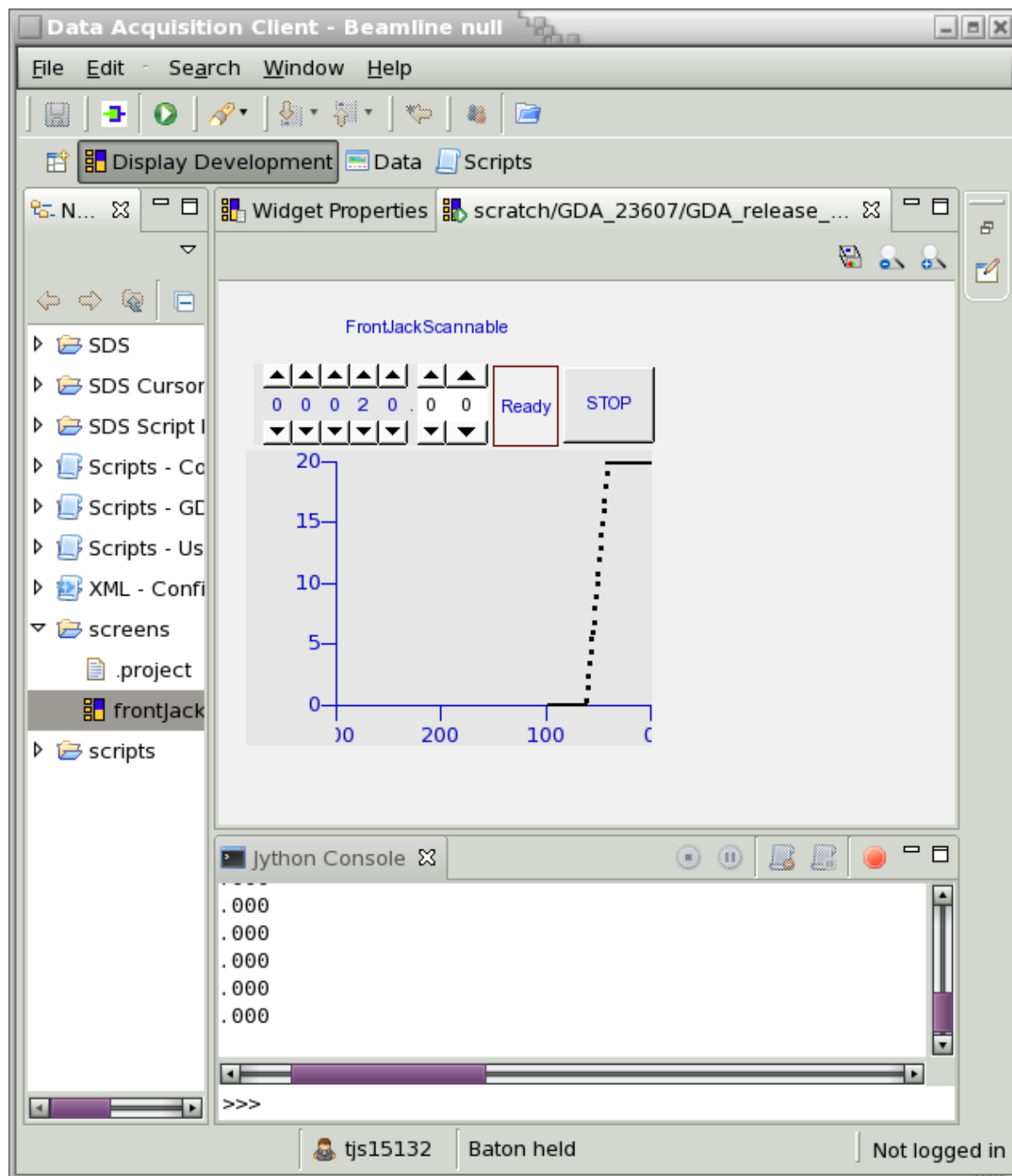
## 5.6 Synoptic Displays

For information on CSS see [http://css.desy.de/content/index\\_eng.html](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.
```



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 Front** 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\$.