Herschel Data Processing HowTo Documents

version 0.4, Document Number: HERSCHEL-HSC-DOC-1199
14 November 2008

Herschel Data Processing HowTo Documents:

Table of Contents

Chapter 1. HowTos Preface

1.1. Introduction

This document is intended to provide a general overview of the main interface to the Herschel Data Processing (DP) software referred to as HIPE (Herschel Interactive Processing Environment). HIPE provides a GUI plus command-line access to the data processing capabilities of the Herschel Common Science System (HCSS).

The intended audience is for the general astronomer new to the DP system who wishes to start with HIPE for doing standard data analysis. More specialist analysis is possible and scripting, allowing for batch processing, can be done within the system.

For those interested in becoming more advanced in the system the `"DP Basic User's Manual"` is also available from within the DP help system and on-line as PDF and HTML documents.

1.2. Change Log

Substantial changes were made in going from version 0.3 to version 0.4. The following items have been changed. Changes are with respect to user release 0.6.6 of the HCSS and DP system.

- HIFI pipeline chapter substantially updated
- SPIRE pipeline chapter updated to include sample output products
- HowTo chapters on spectral display, image display and arithmetic , spectral arithmetic, spectral fitting and image analysis all added.
- HowTo Access Data substantially changed to include updates in access to the Herschel Science Archive.
- HowTo Save and Read data to and from FITS and ASCII files -- information expanded.

The following items have been changed in version 0.3 from version 0.2. Changes are with respect to user release 0.6.5 of the HCSS and DP system.

- HIFI pipeline chapter added
- PACS pipeline chapter added
- SPIRE pipeline chapter updated to include sample output products
- HowTo plot chapter updated.
- HowTo save chapter updated, including missing image.

The following items have been changed in version 0.2 from the original version (v0.1). Changes are with respect to user release 0.6.4 of the HCSS and DP system.

- Update of HIPE overview to new views/capabilities of the HIPE environment.
- How to Save data section added.

Chapter 2. HIPE Introduction

-- Using the Herschel DP Interface

2.1. Introduction

The data processing application for Herschel Data Processing, Herschel Interactive Processing Environment (HIPE), strives to provide an integrated suite of graphical interfaces that can interact with each other. It allows for interactively choosing your active data in your session, visualizing that data in various ways and selecting tools that can operate on the data. Both command-line and GUI interfaces to the user are available. High-level interactions, which can involve GUIs, are also echoed as commands on the command-line that allow the saving of commands used in a session and the generation of scripts from these interactions.

This section of the Data Processing (DP) HowTos manual provides a brief overview of the fundamental elements of the user interaction framework, HIPE. Hopefully this enables you -as a user- to make the most efficient use of HIPE.

2.2. HIPE Philosophy

HIPE tries to embrace several aspects which affects both users as well as developers:

```
    * an integrated application - giving access to all data
    processing functionality in a unified graphical interface
    * one look-and-feel - where window layout, toolbars, buttons,
    and menus are alike
    * a customizable layout - allow the user to decide which windows are
    relevant and how these windows are layed-out on screen
    * user guidance - including command-line echoing of main graphical
    functionality, allowing the user to learn the scripting language by
    interacting with the system interactively
    * extendible application - allow the developer to add new bells-and-whistles
    which are automatically integrated
```

2.3. Installation and Startup of HIPE

HIPE is part of the Herschel Data Processing system and can be installed with the software installer (see Herschel Science Centre website, or installer script). Software can be run on a server or individual workstation running Windows XP, Linux or Solaris. The minimum recommended system is Windows/ Linux 32-bit w/1GB RAM or 64-bit W/Lin/Mac w/1GB RAM; Browsers for use with the systemm (including download) IE 6+ , Netscape 7+, Mozilla (Firefox) 1.5+, Safari (Mac). The system is Java based and requires Java 1.6. General Java scripts can be run on the system. Installation instructions are provided at the bottom of the FTP page.

Once the software is installed, HIPE can be started by several means. Using Windows, Herschel software can be started under the "Start" menu after a standard installation. Alternatively, HIPE can be started from a command line.

```
hipe
```

While HIPE is being launched a splash screen is shown (see Figure 2.1).

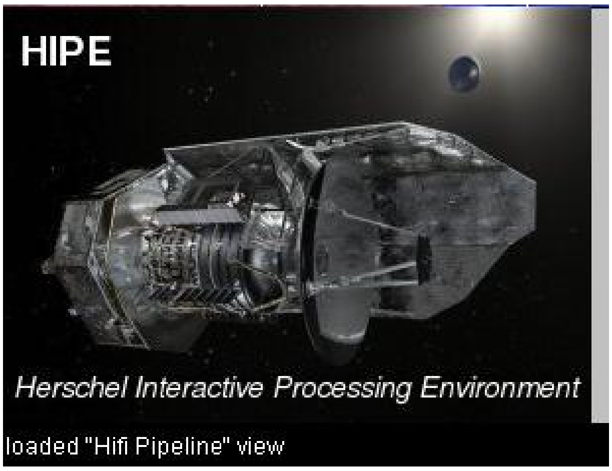


Figure 2.1. HIPE startup splash screen.

2.4. Obtaining Help from within HIPE

Help can be obtained via the Help pulldown menu available at the top of the HIPE screen at any time and with any view the user has. There are three types of Help available (see Figure 2.2").

Figure 2.2. HIPE help available.

- **Contents:** This provides a general view of the help information available to the user. Selection provides a new tab in the default browser of the user showing a hierarchical set of complete DP documentation with more advanced documentation appearing towards the bottom of the screen (see Figure 2.3).
- **Working in HIPE:** This provides access to similar documentation, also from within the user's default browser. In this case, however, Chapter 2 of the HowTos manual which provides an overview of how to interact with HIPE, is opened (see Figure 2.5).
- **Release Notes:** This provides a similar view again, except this time the page automatically opened is that containing the news of the most recent additions/changes to the system (see Figure 2.4).

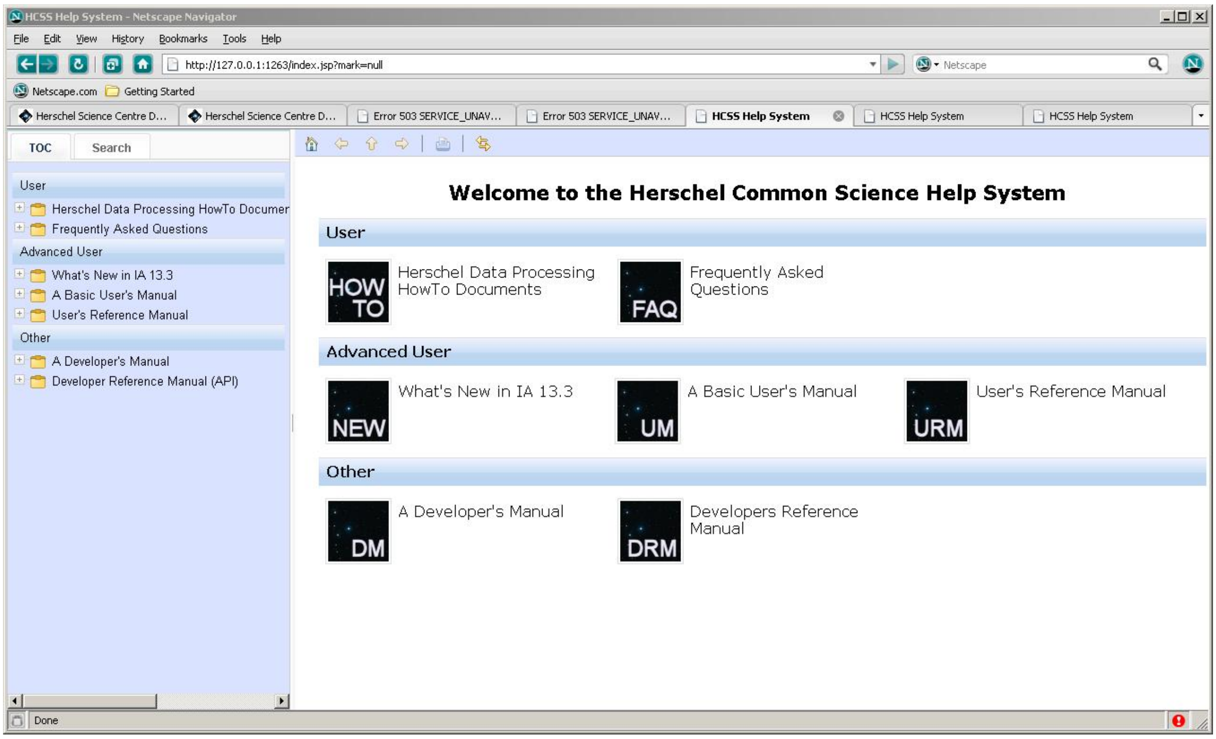


Figure 2.3. HIPE general help contents.

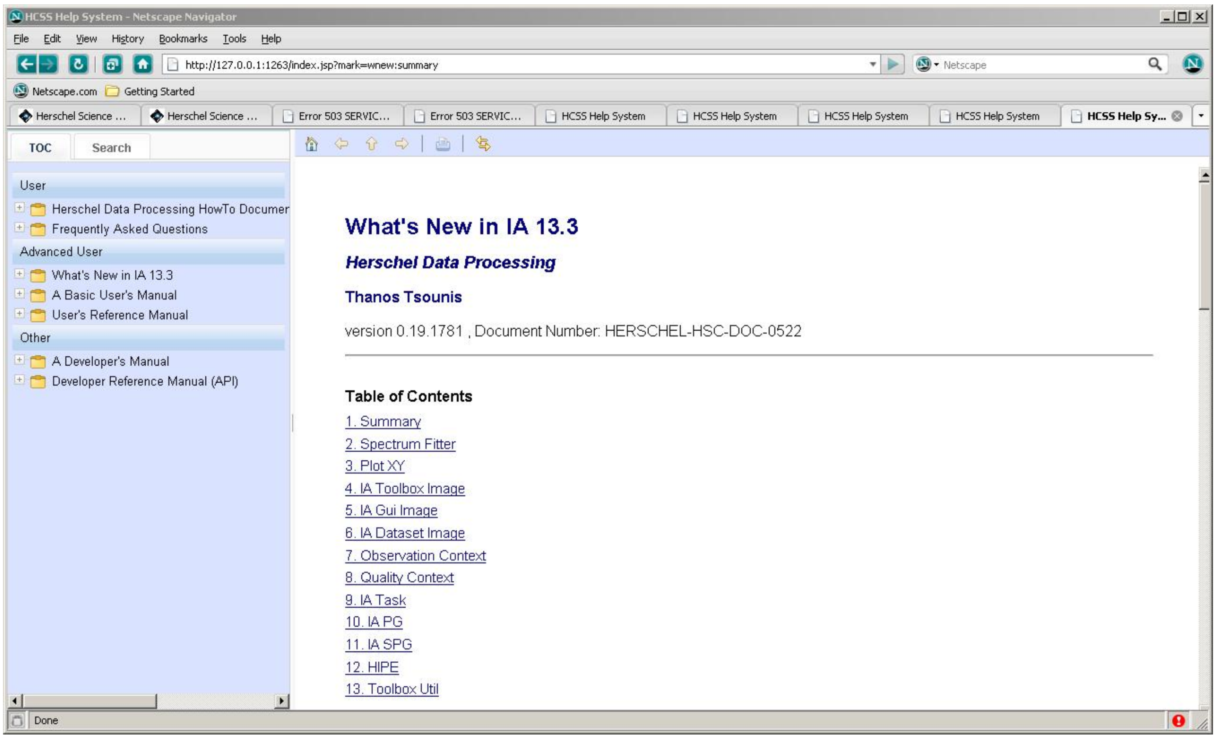


Figure 2.4. Help with HIPE interactions.

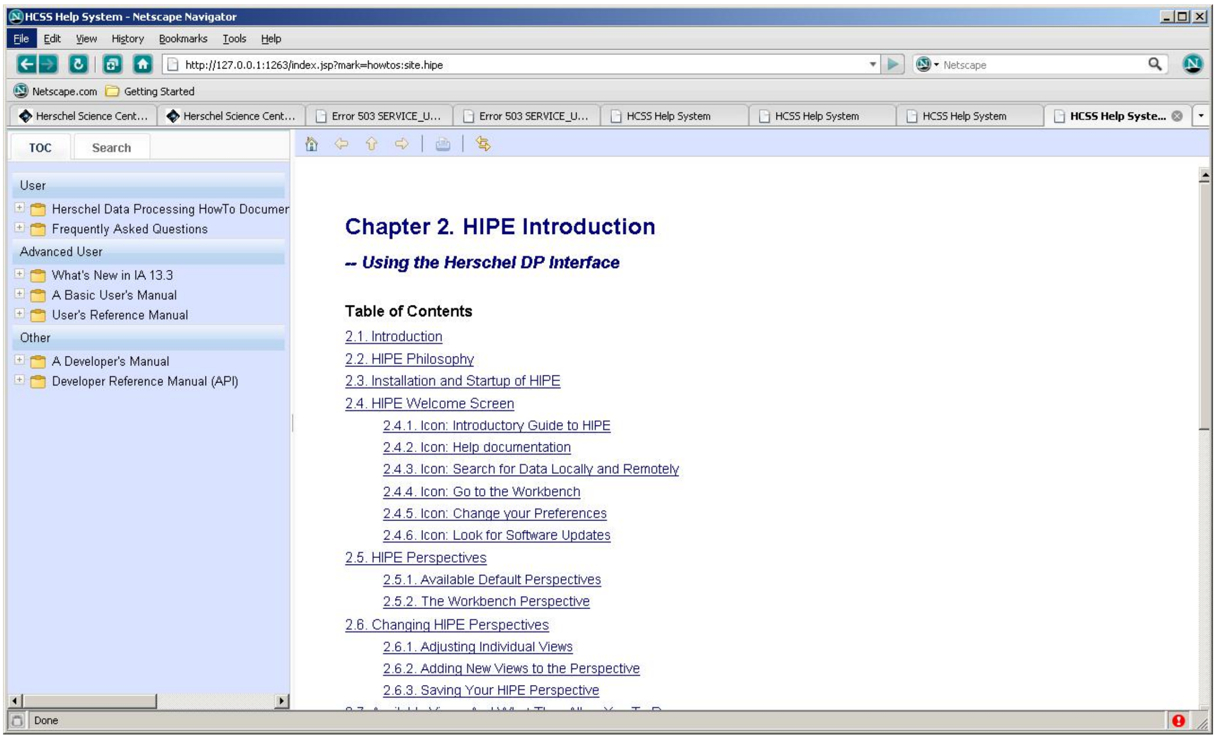


Figure 2.5. HIPE release notes.

Navigation through the help documentation is via standard mouse clicks on the links that appear in the browser window.

2.5. HIPE Welcome Screen

Following launch a Welcome screen is displayed which includes six icons appearing on the HIPE startup screen. Access to the setup for analysis (Work Bench), a data access area (Access Data), and general help (Documentation) is currently available. Later editions will allow for software update searches (Updates) and preference selection (Preferences) and interactions with external tools (External Tools). Placing the mouse over each of the icons on the screen provides a small description that appears along the bottom of the HIPE window (see Figure 2.6).

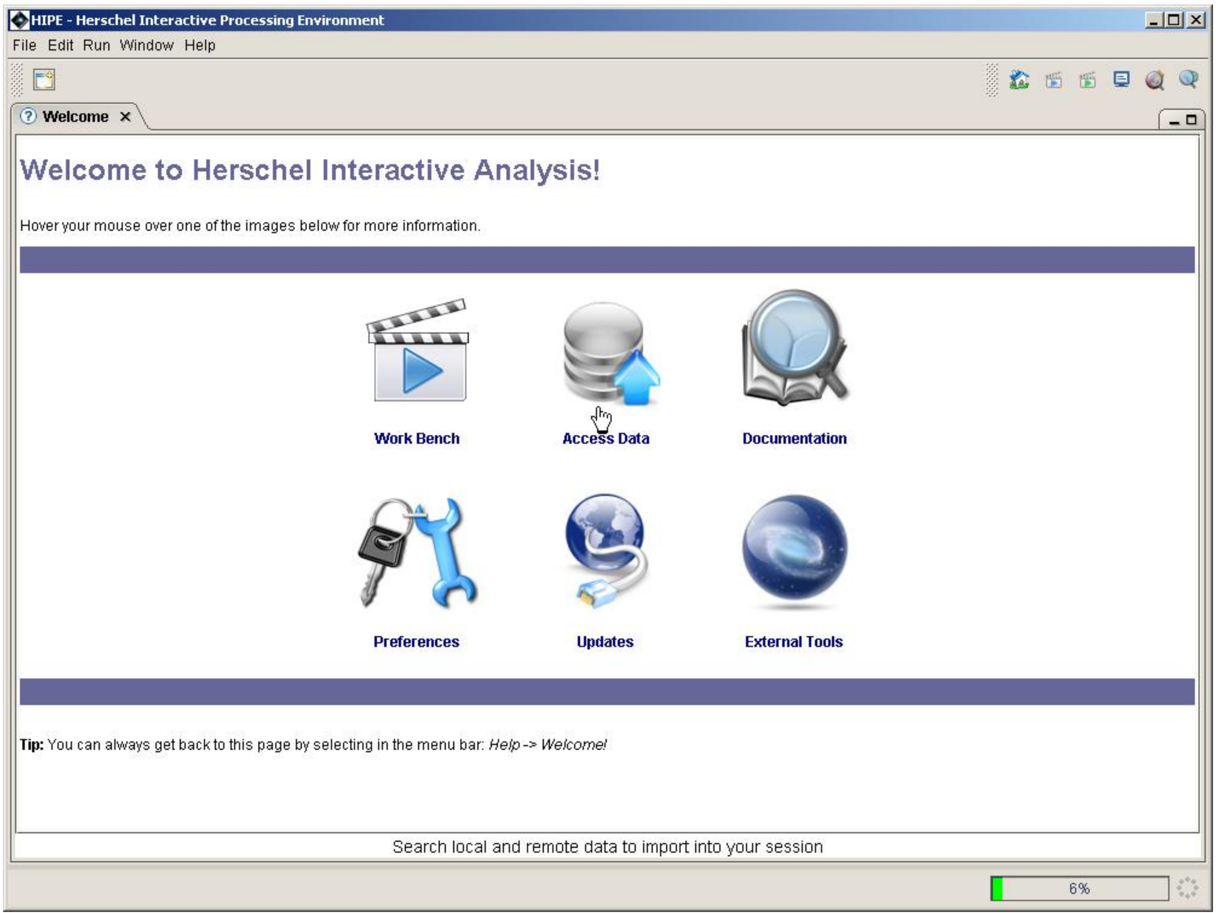


Figure 2.6. Information on 'Welcome' screen icons. See bottom strip of the HIPE screen for the explanation of each icon the mouse is placed over. In this case the Access Data view is stated as being accessible via the icon the mouse is hovering over.

Note that a tool bar exists at the very top right of all window displays of HIPE. This tool bar and its uses are discussed in the section on HIPE perspectives (see Section 2.6). However, just to note here that the Welcome screen can always be returned to by using the 'Help' pulldown menu to 'Welcome' (see Figure 2.7)
. The Welcome screen is also available using the first icon in th elist to the top right of the HIPE screen ().

Figure 2.7. HIPE Welcome screen access.

2.5.1. Icon: Work Bench

Clicking on the  icon takes the user to the workbench perspective (for information on perspectives in HIPE see Section 2.6). The default view of the workbench is shown in Figure 2.8. This is the main work area for doing data analysis. Here we can look at data values, plot spectra and images, create scripts for batch processing and run analysis tools. The contents of the workbench can be updated with various "Views" available under the Window pull-down menu (see Section 2.8 on available Views).

The current default work bench is a somewhat slimmed-down version of the full work bench. Either perspective on the system can also be be provided via use of the "Window" menu. Selection of the "Show Perspectives" and either "Full work bench" or "Work bench" provides the two main default perspectives for when doing work in HIPE.

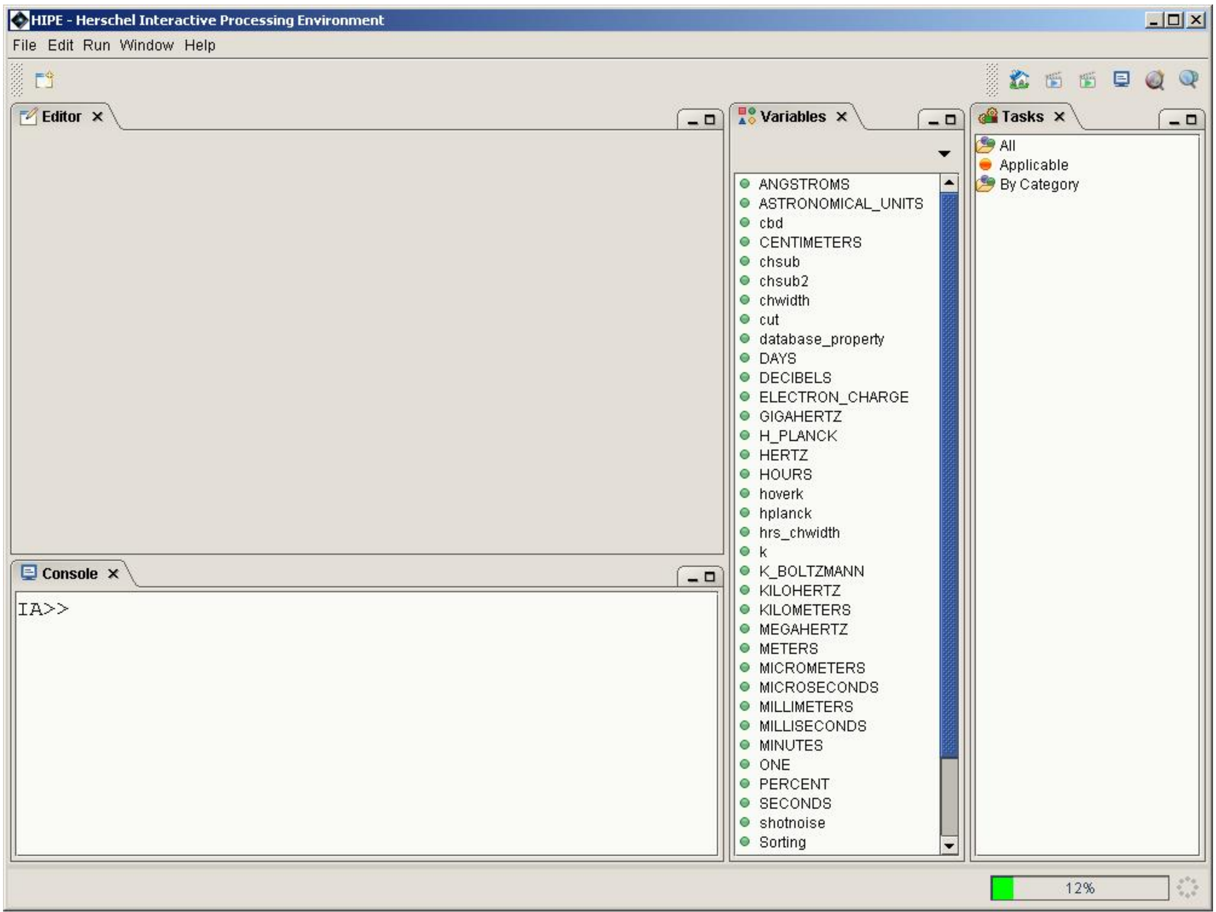


Figure 2.8. HIPE default view of the workbench perspective.

2.5.2. Icon: Access Data

The  icon opens up a replacement window in HIPE that provides access to data held in databases both locally or at a remote site (for example the Herschel Science Archive). It also allows the import of FITS and ASCII table files into and out of a DP session.

The access tools allow the user to search and do queries on stored data and its attributes in order to make it accessible within the processing session.

Four icons appear that allow import/export to databases, direct access to the Herschel Science Archive or import/export of FITS or ASCII table files (see Figure 2.9).

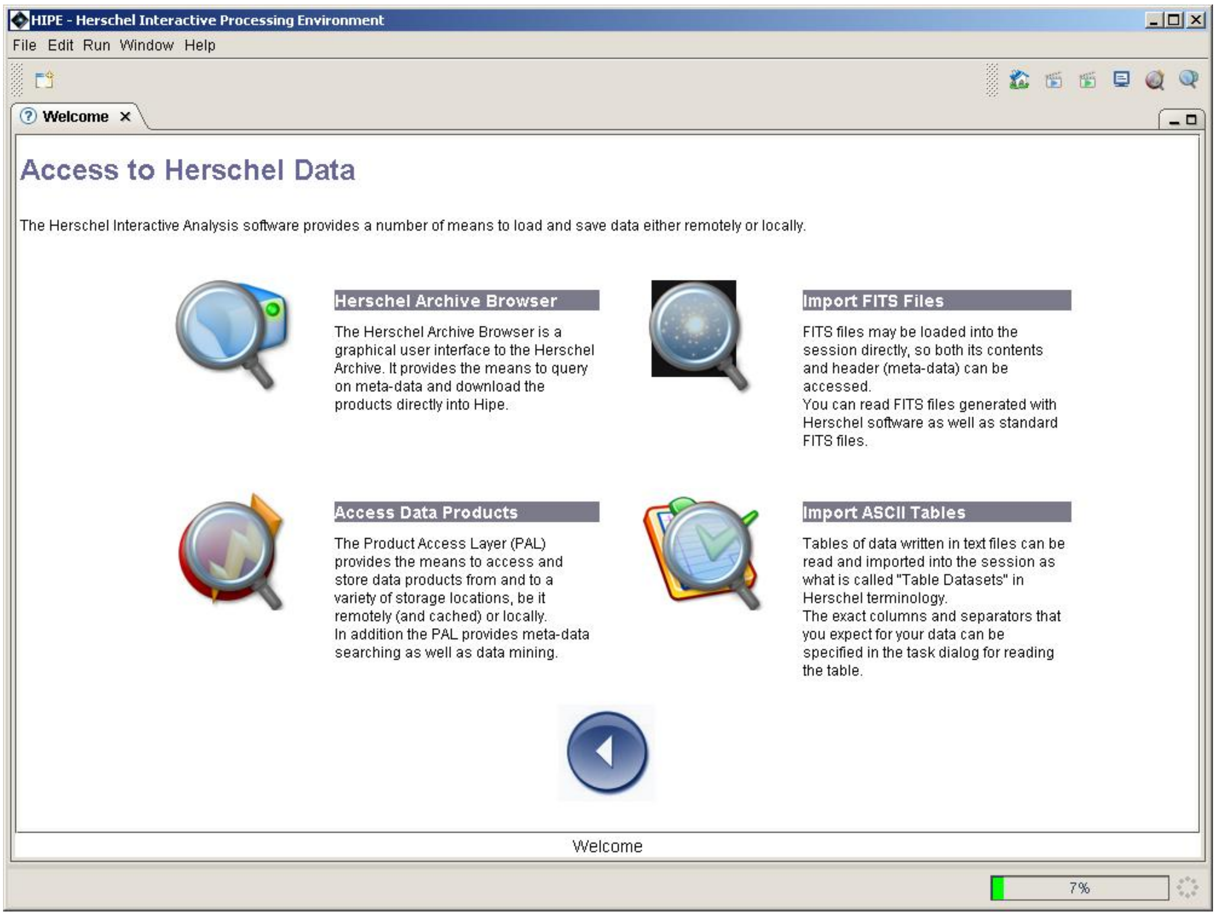


Figure 2.9. HIPE default view of the workbench perspective.

Clicking on the "Herschel Archive Browser" icon opens up the Herschel Archive perspective, while the Data Access icon takes the user to the Product Access Layer perspective (see Section 2.6 for information on these perspectives). The means for actually bringing data into the system is described in detail in the *HowTo Access Data*.

For FITS and ASCII I/O the other icons produce perspectives that allow for this which are based on the default Work Bench plus the simple FITS archive tool or ASCII archive tool respectovely. These are discussed more thoroughly in the *HowTo save and restore data (ASCII and FITS)*.

2.5.3. Icon: Documentation

The  icon allows access to the complete DP release documentation tree. After clicking on this icon, documentation is provided via a web browser and uses the Eclipse system which comes with the HCSS build. The user is able to get the top-level How-To information that explains such basic functionality as accessing data in a database, displaying images and spectra, plus basic image and spectral analysis for Herschel (also see Section 2.4).

Links are also provided to documentation that explain the scripting capabilities and use of the commands on the command-line of a console window. This allows the user access to the full power of the system as well as the creation of his/her own batch mode processing. The scripting language has great similarity to the Jython scripting language and borrows many of the items from that language. This is contained in the "Basic User's Manual".

User task commands, numerical package and product storage information is also available in the User's Reference Manual (URM). The URM provides a short inrtroduction to any of the commands available. Help for a given command displays the URM contents for that command.

Developer documentation for the complete system is available. These are in JavaDoc format (described in Chapter 9 of the "Basic User's Manual"). Any of these commands may be used at the console command line or within scripts produced for the DP system.

2.5.4. Icon: Preferences

The icon  functionality is NOT IMPLEMENTED YET. Clicking on this item will change Herschel DP system preferences for the user.

2.5.5. Icon: Updates

The icon  functionality is NOT IMPLEMENTED YET. Clicking on this item will (in future) allow the user to search for software updates available from the Herschel Science Centre.

2.5.6. Icon: External Tools

The icon  functionality is NOT IMPLEMENTED YET. Clicking on this item will allow access to certain tools external to the Herschel DP system.

2.6. HIPE Perspectives

When going to the workbench or using the welcome icon link to the data access capabilities of HIPE, the user is presented with a "perspective". A "perspective" is a presentation of the system that is made available to the user through a set of "views" (basically separate windows within the environment that provide particular capabilities). The following section discusses the views the user can have, but in this section we describe perspectives and in particular the default workbench perspective. We also discuss how the user can control a perspective to make it as simple or as complex as wished.

HIPE is built-up from several graphical elements, of which the fundamental ones are shown in Figure 2.10, which provides the full work bench. A perspective is a collection of graphical windows ("views") organized in a way to focus the user on doing a specific job within the whole suite of jobs that a user can and will do within the system. It may consists of one or more views and, optionally, the editor area; these windows are then organized in tabbed panes and split panes. Many of the views also contain their own toolbars. These toolbars are in addition to the toolbars for HIPE displayed at the top left (editor capabilities for editor window view) and right of the HIPE screen (icons providing access to full set of defaults perspectives -- hover mouse over icon to view perspective name).

Figure 2.10. A single element (view) for a HIPE perspective.

2.6.1. Available Default Perspectives

There are five perspectives that come pre-packaged in the system. *These can always be obtained by using the toolbar at the top of the HIPE window. Click on "Window" and pull down to "Show Perspectives", which provides the list.*

2.6.1.1. Product Access Layer Perspective

The **Product Access Layer** perspective provides a convenient means of getting and briefly viewing data from databases and data stores -- both locally and remotely stored. This is illustrated in Figure 2.11. There are 4 windows ("views") including an editor where DP scripts can be created (see the DP User's Manual).

Data can be queried from a locally stored database (default is under the ~/.hcss directory) or remotely registered database using the "Data Access" view seen to the left of the perspective (see ???). More information on how to get data from databases and the Herschel Science Archive is available from the chapter *HowTo Access Data*.

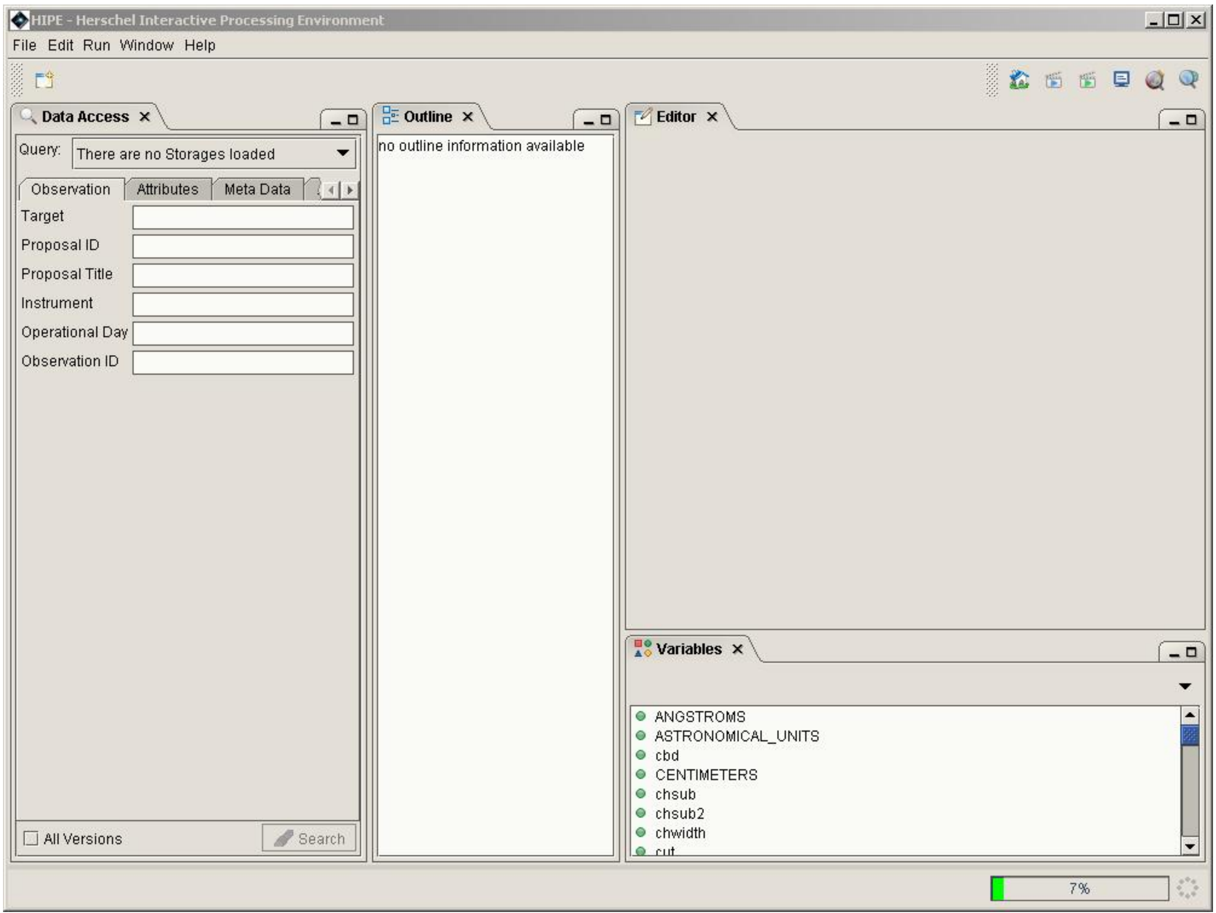


Figure 2.11. HIPE Product Access Layer (PAL) perspective. This provides access to data stores both on-line and on the user's own computer.

2.6.1.2. Classic(JIDE) Perspective

The **Classic JIDE** perspective provides a scripting environment with 3 windows that provide an editor/debugger window, a console window and a log window. This is the basic view of the system used during earlier development of the DP system (see Figure 2.12). A new Jython (DP) script window can be added by clicking on the  icon at top left of the Editor window. More information on the Editor view can be found at Section 2.8.4. The DP User's Manual, available under the Help menu, also provides significant further help on JIDE itself and the HIPE/JIDE view in creating user scripts.

The same perspective can be obtained by clicking the  icon to the top right of the HIPE window.

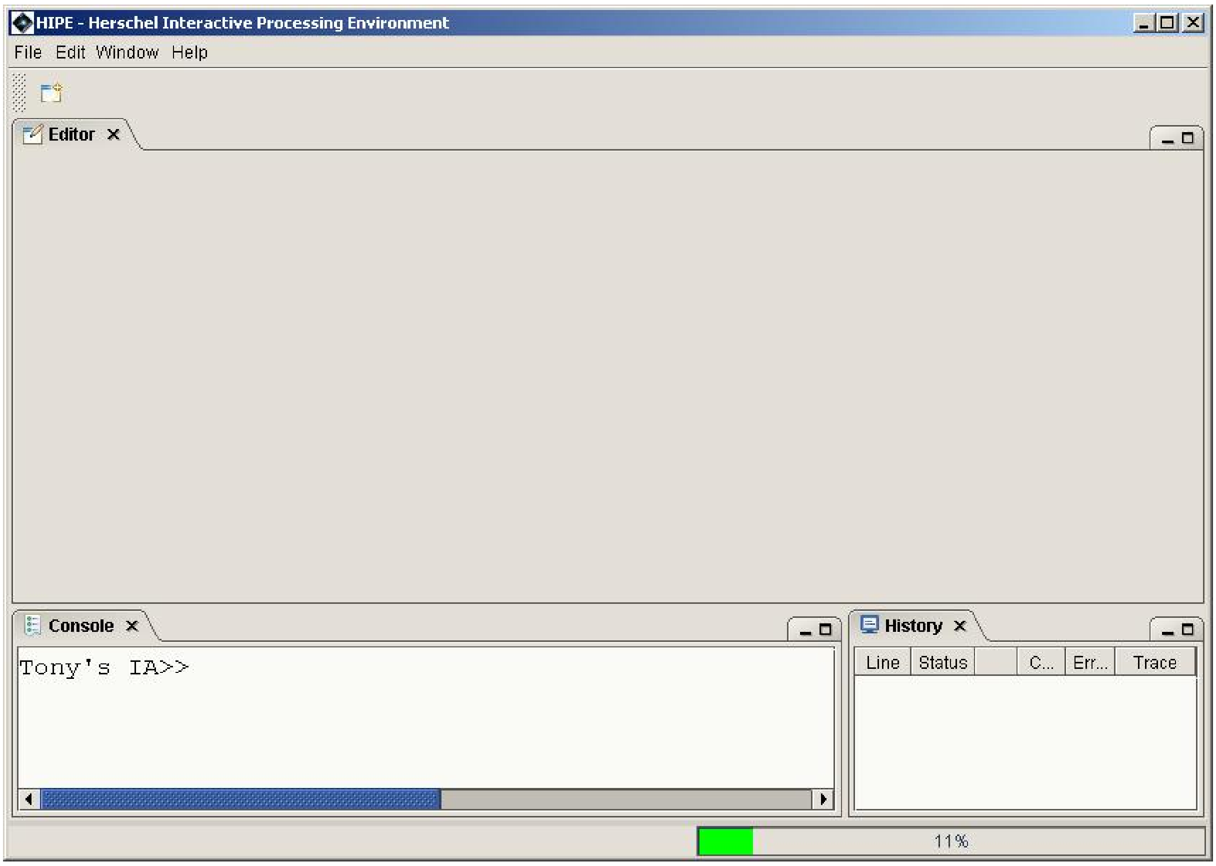


Figure 2.12. HIPE's 'classic' JIDE perspective.

2.6.2. The Full Work Bench Perspective

The **Full Work Bench** perspective provides a general environment with multiple windows, five of which are prominent (editor, console, variable list, outline, run tools). Other windows/views are available by clicking on the tabs, e.g., Navigator, Classes (see Figure 2.13).

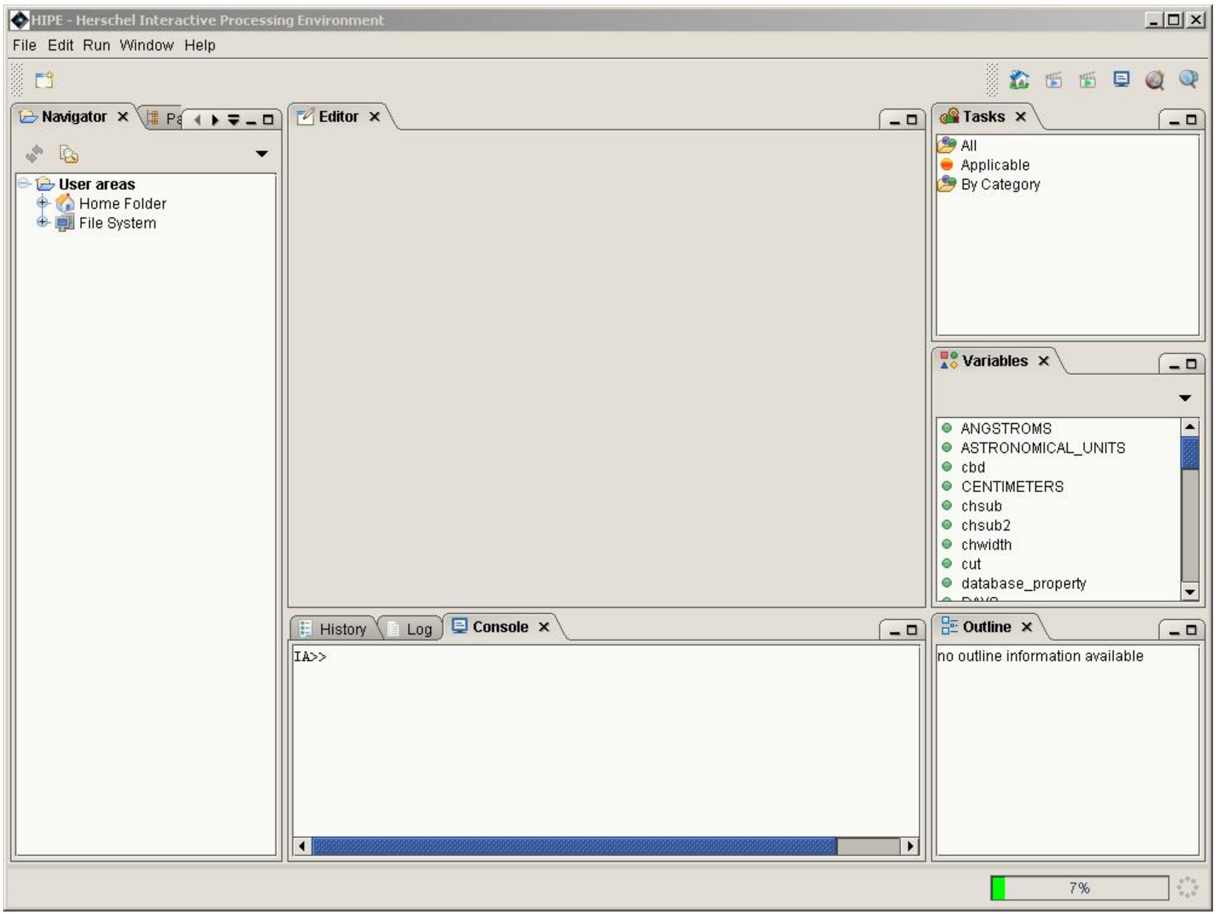


Figure 2.13. HIPE view of the full work bench perspective.

2.6.3. The Work Bench Perspective

The **Work Bench** perspective provides a slimmed-down general environment similar to the work bench but with only with four windows (views). The editor, console, variable list, outline, tasks views are available (see Figure 2.14).

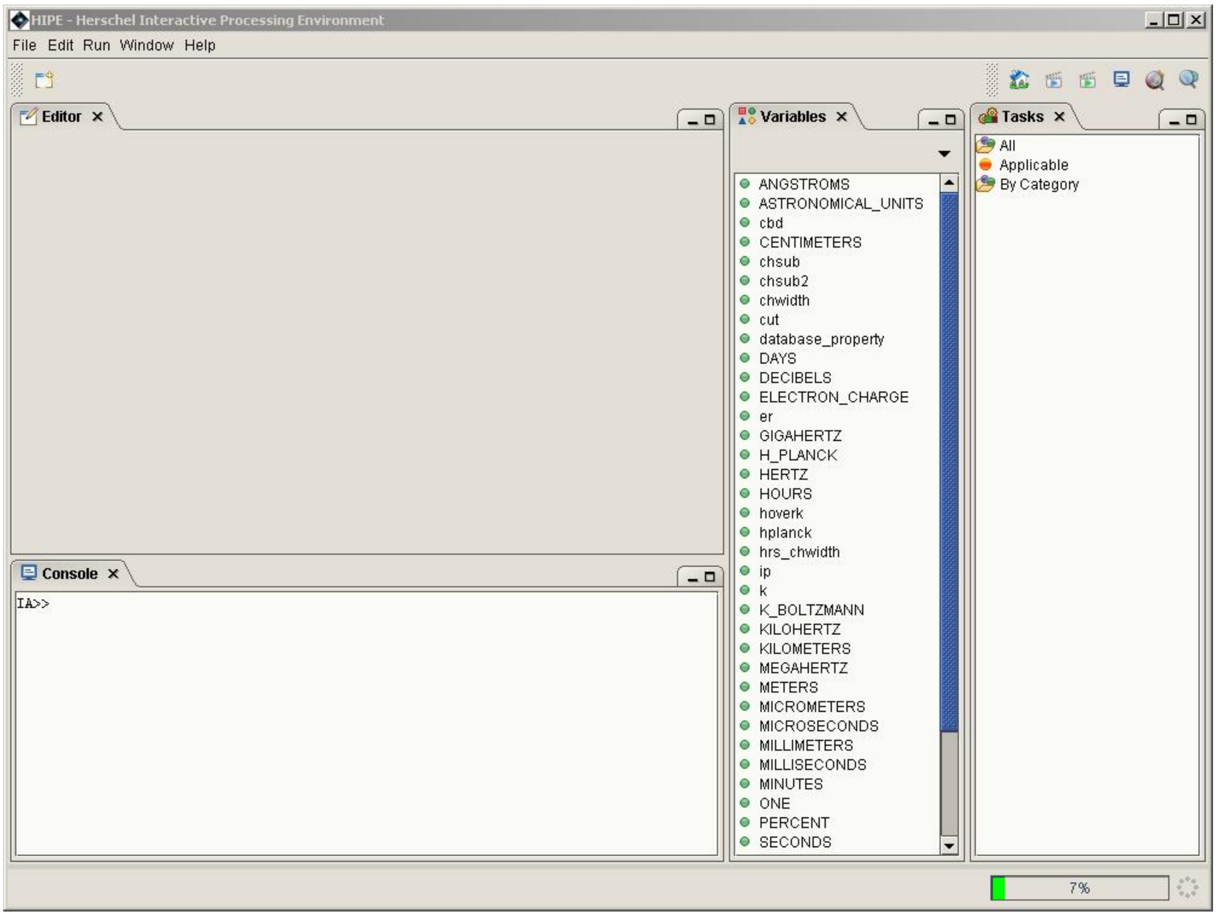


Figure 2.14. HIPE default view of the work bench perspective.

2.6.4. Archive Browser

The **Archive Browser** perspective provides a convenient means of querying and obtaining data from the Herschel Science Archive (HSA). There are three views related to providing log-in information for the HSA, the connection to the HSA (via plastic VO protocol) and the loading of selected data from the archive (see Figure 2.15).

Queried data appear under a single, selectable variable in the DP session (under Variables view) and a click on the variable allows its outline to be provided in the Outlines view. These two views are described in more detail later in this chapter. Further information on how to get data from databases and the Herschel Science Archive is available from the chapter *HowTo Access Data*.

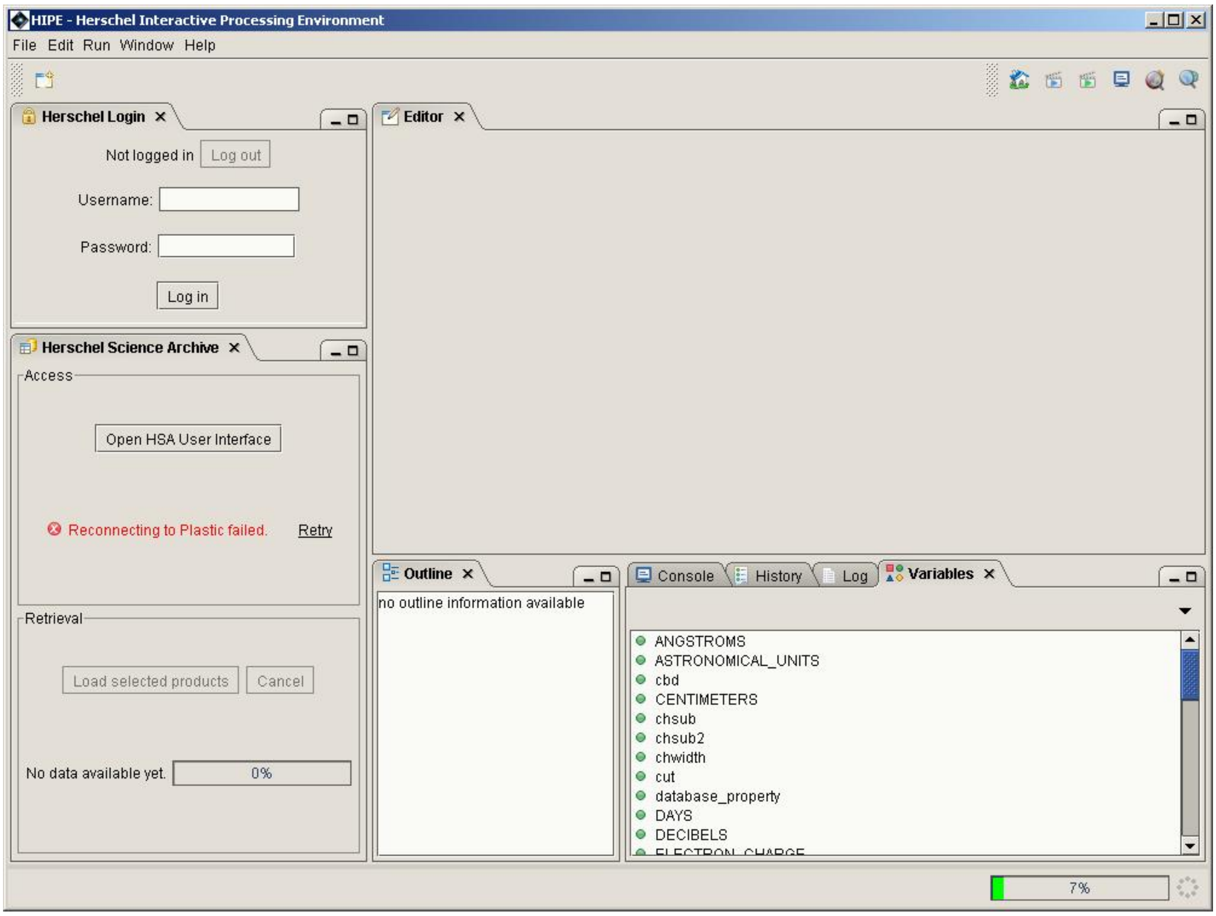


Figure 2.15. HIPE Archive Browser perspective. This provides access to the Herschel Science Archive (HSA). In this case an error is shown in red indicating that access to the archive server has not been established in the DP session.

2.7. Changing HIPE Perspectives

Changing a perspective to a worksurface that a user prefers can be done in various ways. Each window can be resized or dragged to different areas of the workspace. Also, new views can be added to a perspective.

2.7.1. Adjusting Individual Views

Each individual window can be adjusted in the following ways.

- **Window resizing**. These can be adjusted in a standard way. To the top of each window, the cross (X)on the tab being clicked removes the window/view. The underscore line minimizes a window (_) while the window can be maximized or returned to its original size by clicking of the box figure in the tab at top right. Minimized windows appear to bottom left of the workspace (see Figure 2.16). Holding the right mouse button down while on the window tab also provides a menu which includes the same options.

  Clicking and dragging borders of each of the windows allows for expansion in any direction of any of the views.

- **Window Tab Placement**. A right click on the view tab provides a pull-down menu that allows some default window resizing and also tab placement and direction of written label (see Figure 2.17).

- **Moving Views**. Windows can be moved inside the HIPE workspace by clicking on the window itself and dragging to another part of the worksurface. Outline black boxes appear on the screen indicating where the window would be if the mouse button was released at that point.

  It is also possible to completely *Undock* a window view by holding the right mouse button down while on the window tab. Pulling down on the menu to "Undock" allows the view to become a separate window that can be moved completely off of the HIPE surface (see Figure 2.18 for an example). To move this undocked window the user need only click on the top, blue part, of frame of the window and drag to wherever he/she wishes on the screen surface.

- **Moving Between Windows in a View**. Windows can be moved inside a view using the arrow buttons to the top right of the view. The left and right arrows toggle through the windows available in a view, while the down arrow allows window selection from a list (see Figure 2.19).

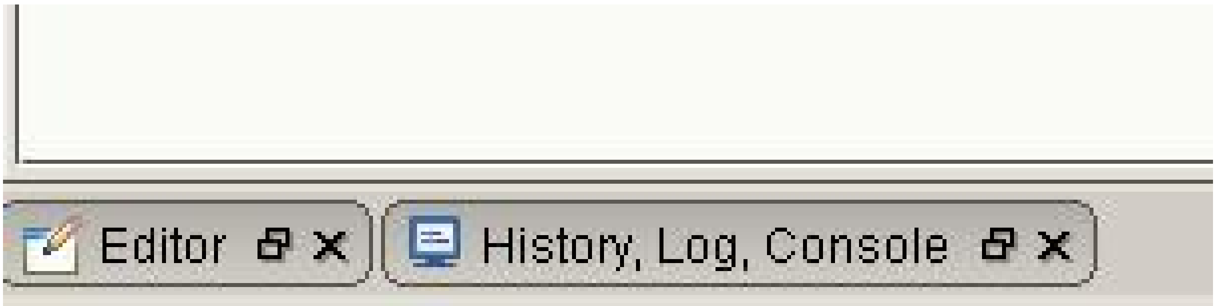


Figure 2.16. Minimized window appearance at the bottom of the HIPE window.

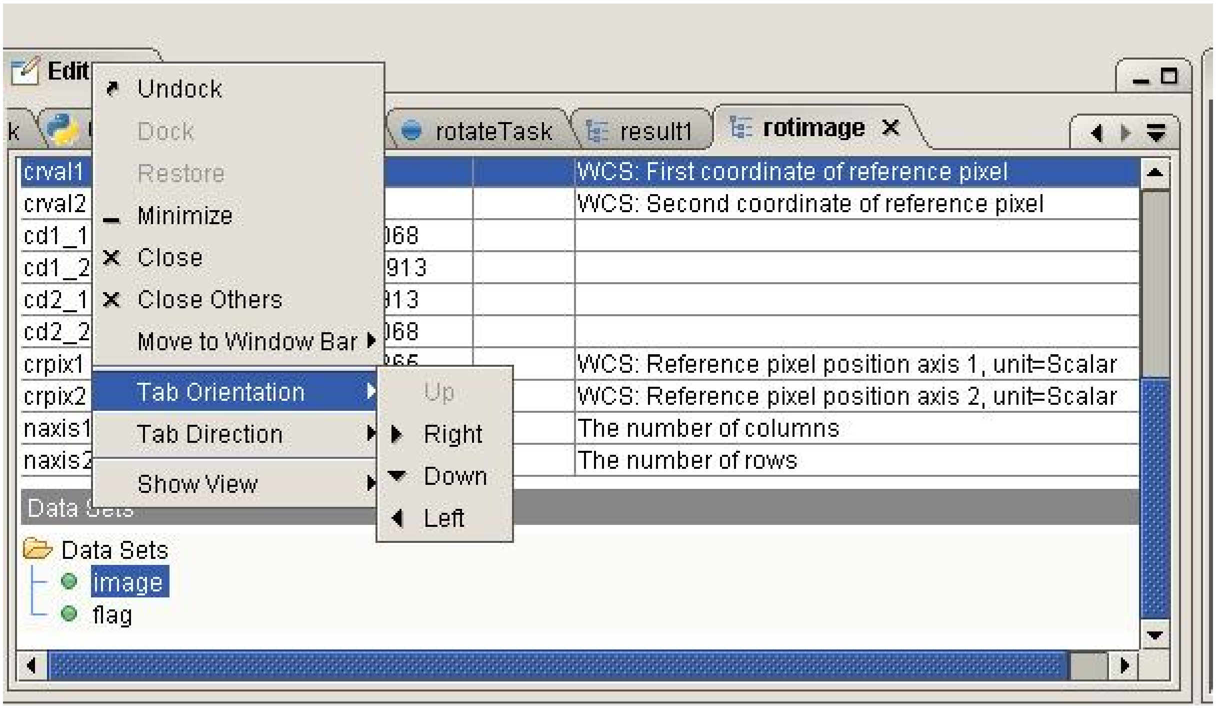


Figure 2.17. Changing tab positions in a HIPE view.

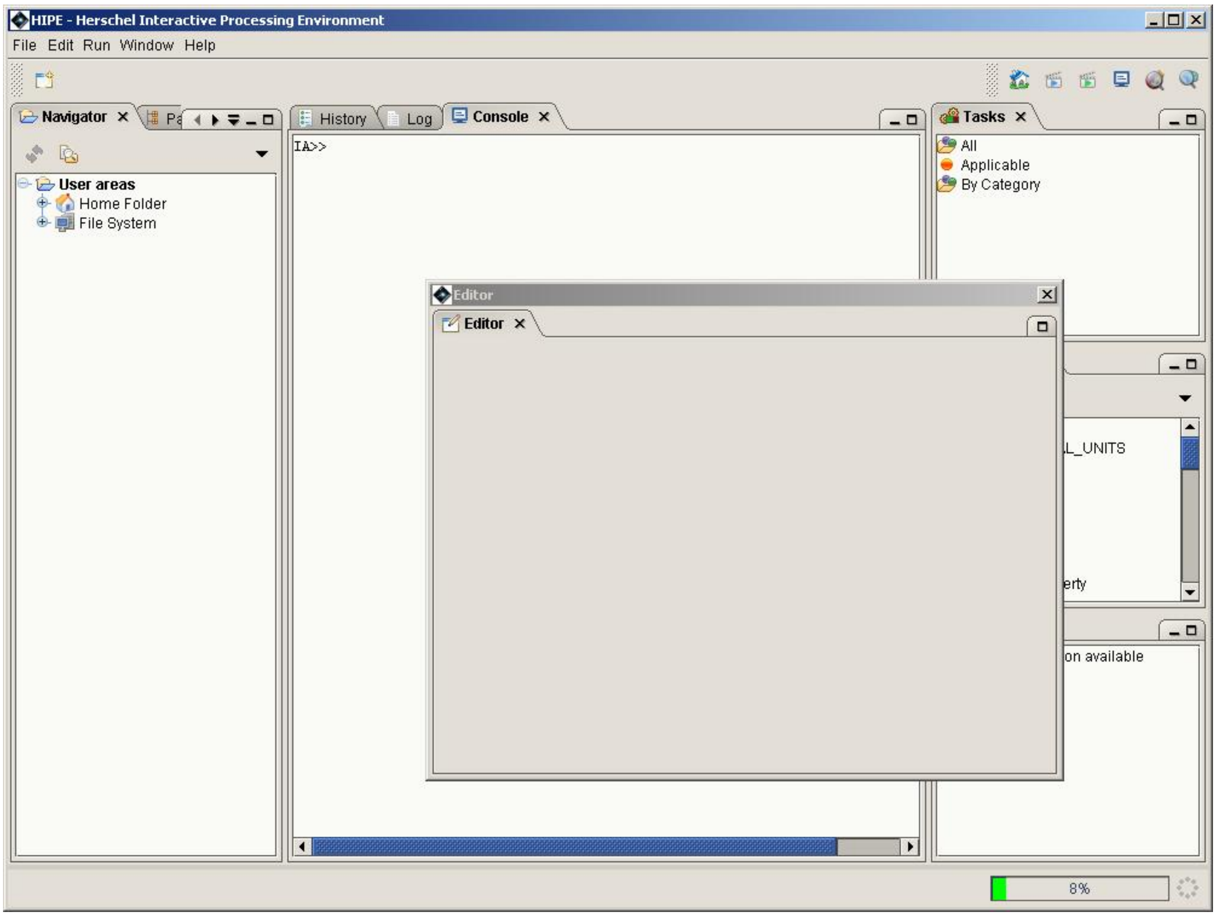


Figure 2.18. Example of undocking a view using HIPE. In this case the Editor view has been undocked and now sits "over" the HIPE worksurface and can be dragged to anywhere on the user's computer screen.

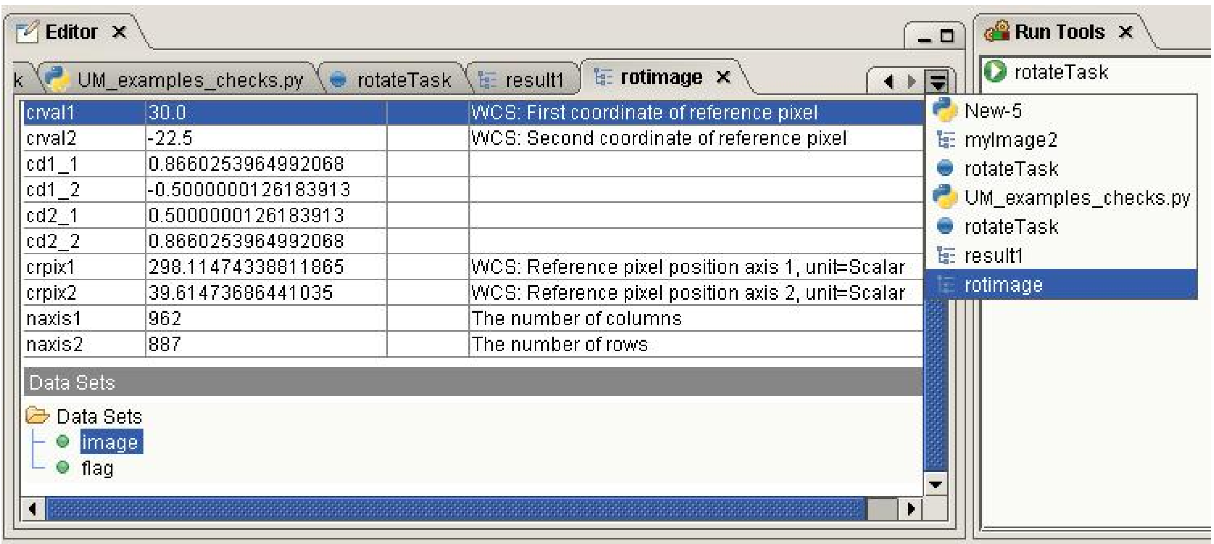


Figure 2.19. Selecting windows within a view. The down arrow shows the list of windows available in the Editor view that the user can move to.

2.7.2. Adding New Views to the Perspective

Several additional views can be added to a perspective. The complete list is obtained from the Windows menu on the toolbar at the top of HIPE. Pulldown to "Show Views" to show the available views in the system. Click on one to add that view to the current worksurface (see Figure 2.20).

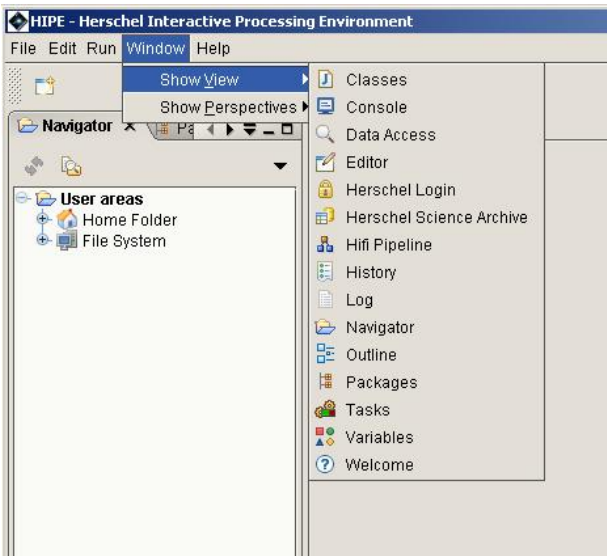


Figure 2.20. The 'Show Views' selection from the Windows pulldown menu lists the views that can be added (note: if the view already exists then a new one is not added).

2.8. Available Views And What They Allow You To Do

Each view has particular capabilities that can be combined to provide a powerful interactive environment. However, the environment can be simplified to a few windows to make a perspective, as noted above. . The views available under the HIPE "Window" pulldown menu on the toolbar can be added to any perspective.

2.8.1. Classes

This view allows the user to see all the classes (routines) currently available in the session. These can include scripts written and loaded into the system by the user. Help information for any of the classes can be obtained by use of a right mouse button click. This brings up a small menu which provides access to Help.

Help information on a class appears in the "Topic Help" view.

Both these views are available in the default "Workbench" perspective (see Section 2.6.1).

2.8.2. Console

The Console view is also available in the default workbench window. It provides a terminal-like input for the DP system where command-line DP inputs can be made. A prompt (editable in a user's properties) is provided.

Re-running commands. It is possible to cut and paste command lines into the window. It is also possible to rerun commands by clicking on the window then hitting the up arrow key until the command that requires repeating is reached. Editing of the command line can then be done before hitting return again to rerun the (edited) command.

Note that the console inputs are the same as for the classic JIDE case and its full use is described in the *Basic User's Manual*. Outputs such as plots or images will appear as separate tabbed windows within the editor view (see below for more information on the Editor view).

The console window is also where printed output from routines appears. So a routine that involves a print output will provide that printed output to the Console view (see Figure 2.21).

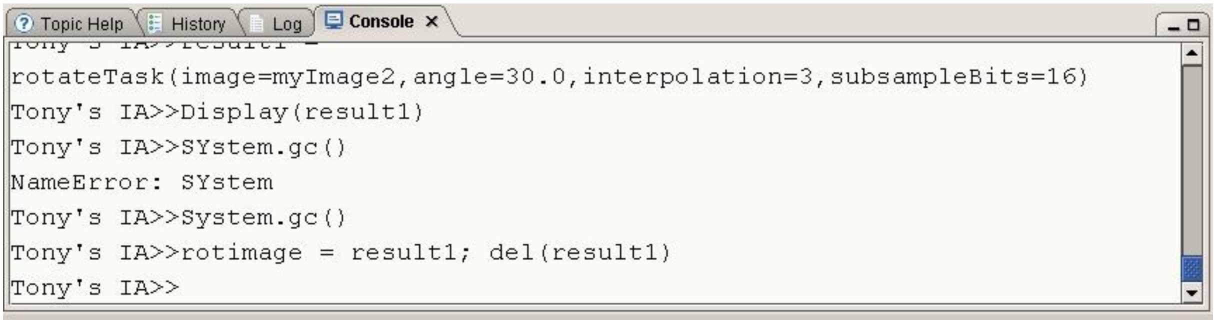


Figure 2.21. The Console view is where command-line input can be made and where feedback command-lines appear following the use of a GUI.

2.8.3. Data Access

This view brings up the interface for downloading data into a session (see Figure 2.22). This provides a mechanism for interacting with a set of data on a user's machine or data contained in remote databases, including the Herschel Science Archive (HSA). The data can be accessed by several means;

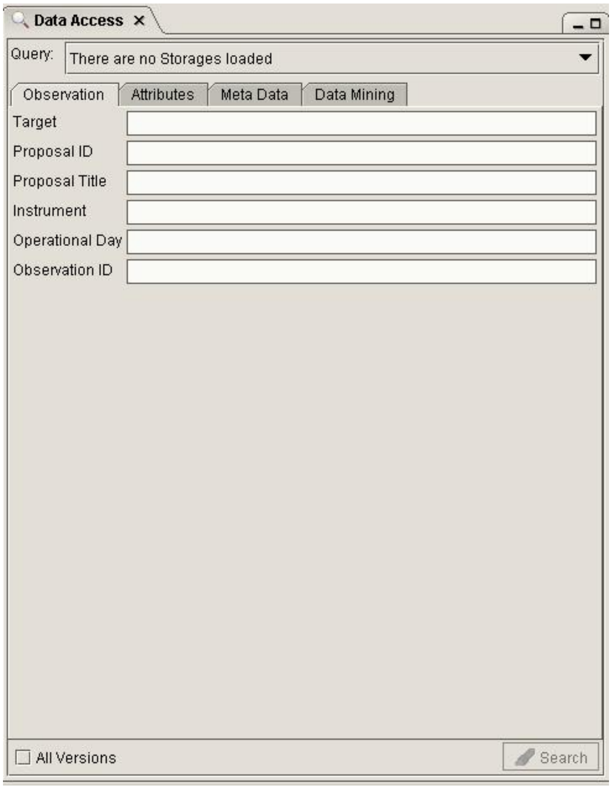


Figure 2.22. Outline of a variable in the DP session is shown in the Outline view.

- **Observation:** which allows querying for observations by target, proposal information, instrument or observation id/day of observation.
- **Attributes:** which allows data selection via attributes in the data products such as creation date and instrument model.
- **Meta Data:** which allows selection based on metadata associated with the products in the database (TBD).
- **Data Mining:** which allows selection based on information contained within the science data themselves (TBD).

2.8.4. Editor

The Editor view is where scripts are displayed and can be edited (see Navigator view information). When scripts are viewed a complete editing environment is made available allowing for standard editing capabilities. These are available under the "Edit" toolbar at the top of the HIPE work area (see Figure 2.23). A number of edit capabilities are also shown by icons (
) which can be clicked to perform certain tasks. Hovering the mouse over each of the icons indicates the edit function it performs, e.g. the torch icon provides find/replace edit capabilities and the double arrow icon allows the loading of everything showing in the current editor window. The run icon will run the highlighted lines shown in the currently shown edit window. If no lines are highlighted then the one line will be run from where the cursor is presently placed at the side of the script. An arrow appears marking this position when the mouse is clicked next to a line -- in left hand side grey region -- in a script window (see Figure 2.24).

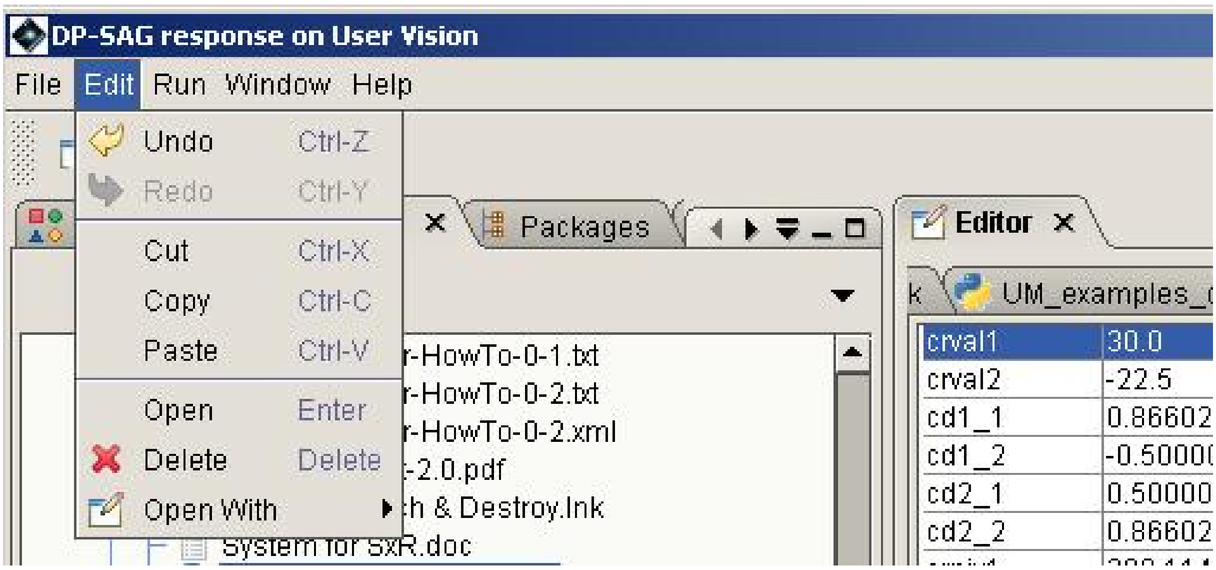


Figure 2.23. The Edit toolbar.

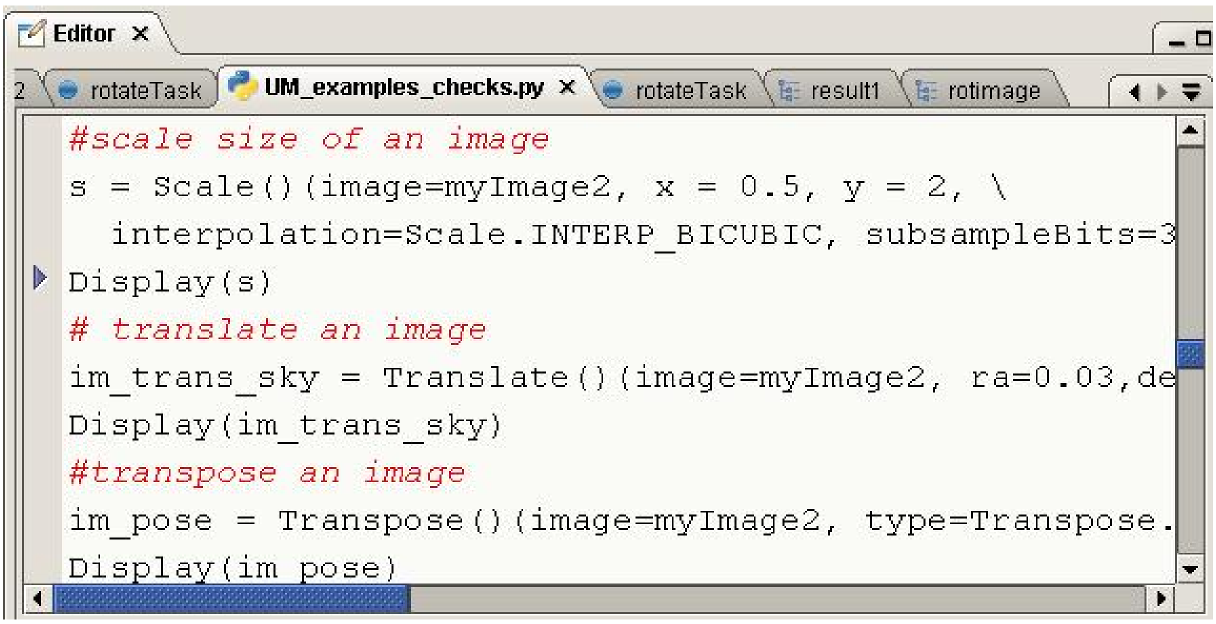
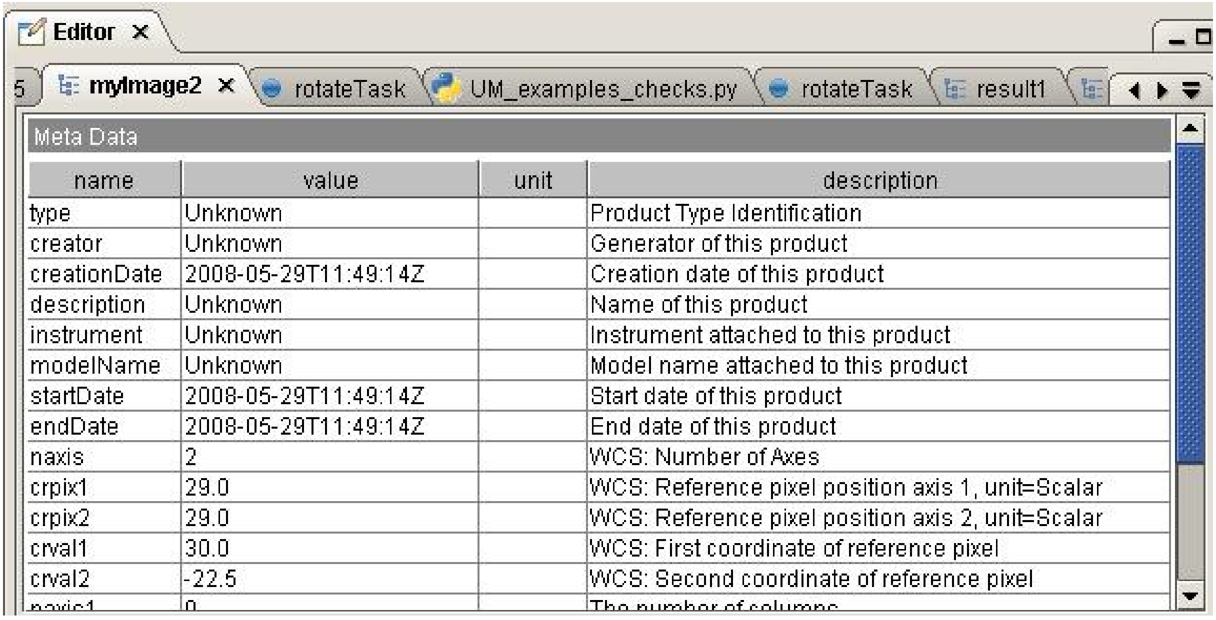


Figure 2.24. The Edit arrow is placed next to the line the user wishes to execute next. In this case, the Display task would be called once the Run button was clicked.

Once a script is initiated, it can be halted by clicking the red highlighted square (Stop) icon. NOTE: the current line of the script will be completed before the script stops running. This can lead to a delay before coming to a halt.

The Editor view is also where informational overview or the contents of a DP file type are displayed -- when requested. It is also the area where plots -- which are in themselves editable, e.g. zoom, pan, change of labeling, task dialogs etc. -- are placed. Examples are shown in Figure 2.25 and Figure 2.26.



name	value	unit	description
type	Unknown		Product Type Identification
creator	Unknown		Generator of this product
creationDate	2008-05-29T11:49:14Z		Creation date of this product
description	Unknown		Name of this product
instrument	Unknown		Instrument attached to this product
modelName	Unknown		Model name attached to this product
startDate	2008-05-29T11:49:14Z		Start date of this product
endDate	2008-05-29T11:49:14Z		End date of this product
naxis	2		WCS: Number of Axes
crpix1	29.0		WCS: Reference pixel position axis 1, unit=Scalar
crpix2	29.0		WCS: Reference pixel position axis 2, unit=Scalar
crval1	30.0		WCS: First coordinate of reference pixel
crval2	-22.5		WCS: Second coordinate of reference pixel

Figure 2.25. A window shows metadata associated with an image within the Editor view.

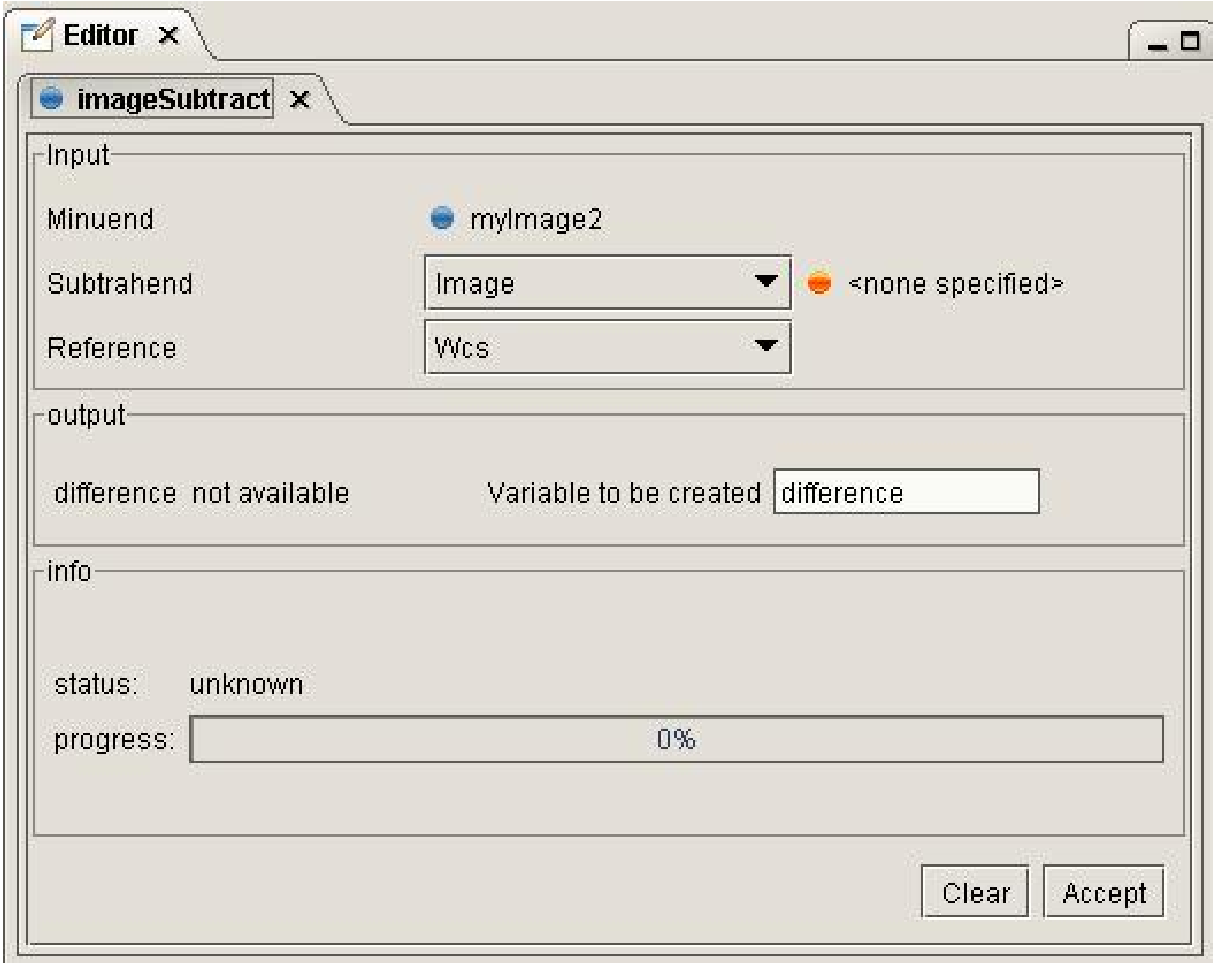


Figure 2.26. Window showing a task dialog associated with an image rotation within the Editor view.

The area can hold several (tabbed) windows so multiple plots/scripts/file contents can be open at one time.

2.8.5. Herschel Login

This view allows the user to login to the Herschel Science Archive (HSA). It is also available as part of the Access Data perspective noted previously. See Figure 2.27. The user enters username and password which allows certain priviliged access to the archive system.

Figure 2.27. The Herschel Science Archive login screen provided by the Herschel Login view.

2.8.6. Herschel Science Archive

This view allows the user to access the Herschel Science Archive (HSA). It is also available as part of the Access Data perspective noted previously. See Figure 2.27. The user can get the HSA interface by clicking the "Open HSA User Interface" button. Once selection is done then the "Load Selected Products" button will bring selection into the HIPE session. More information is provided in the HowTo chapter on Data Access.

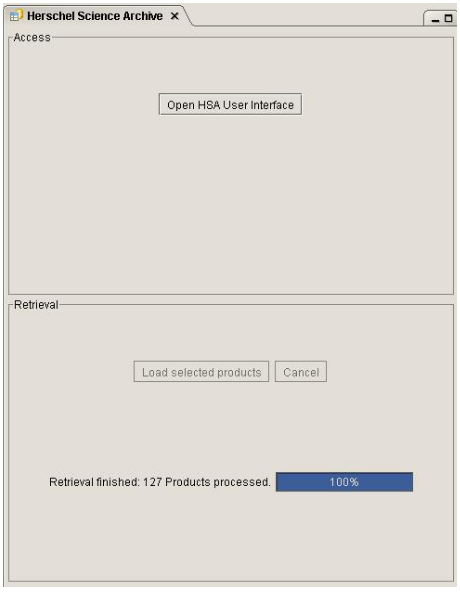


Figure 2.28. The Herschel Science Archive interface view.

2.8.7. HIFI pipeline

This has not been fully implemented yet, but will be a specific view from within which it will be possible to run HIFI pipelines (in part or full).

2.8.8. History

The History view provides a listing of the commands executed at the console or lines executed from the Jython script window of the Editor. This also shows whether the command was successful or not. A tick () indicates the command supplied was successfully executed. A white cross in a red circle () indicates that there was a problem when performing the command. A click on the small plus sign in a circle () next to this will expand out the error information including a complete traceback (see Figure 2.29).

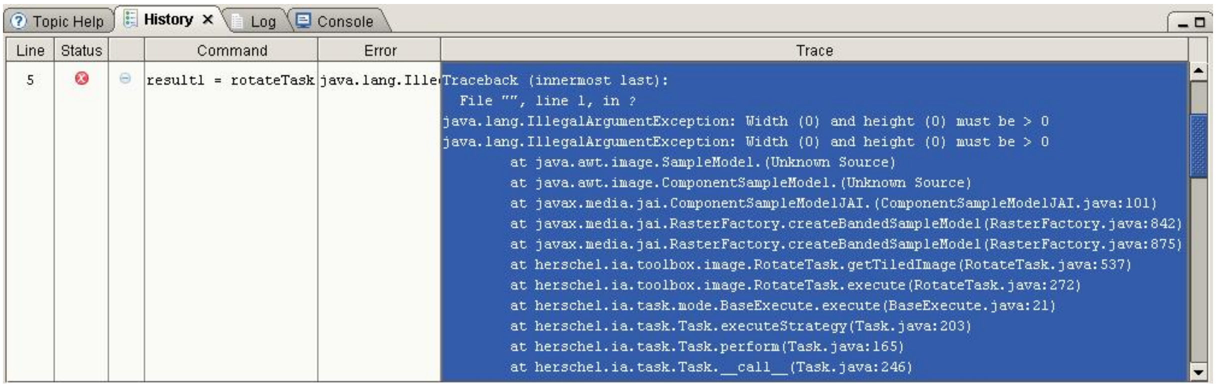


Figure 2.29. A traceback of errors is available from the History window.

History can be saved and used for later batch processing (see ???).

2.8.9. Log

The log screen provides a logging of the commands that have been exectured from the command-line or the equivalent from dialog interactions in HIPE. It also indicates warnings generated in the system. The warning system level can be adjusted by the pulldown menu available at the arrow to top right of the window, from FINE to SEVERE warning levels (see Figure 2.30).

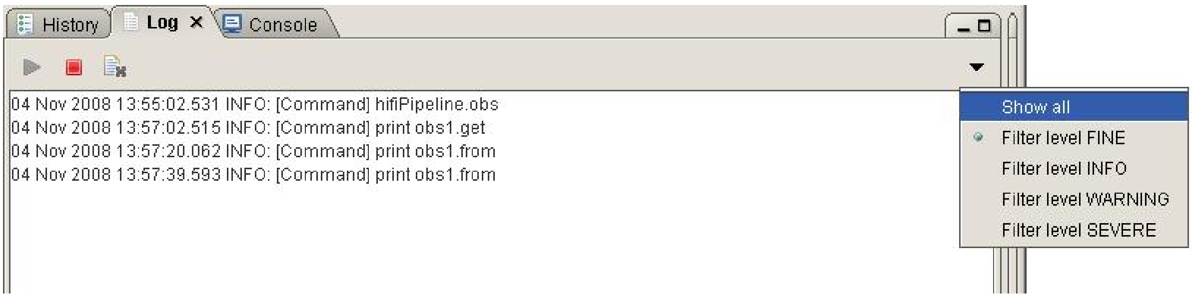


Figure 2.30. The Log screen with pull-down menu showing warning levels that are to be filtered and displayed in the Log view.

2.8.10. Navigator

The Navigator view provides access to the user's directory environment. By default it provides a listing of the user's home directory. Certain types of stored information can be brought into the session and displayed. A right click on an item in the Navigator list provides items indicating what may be done with the particular file (see Figure 2.31).

A prime example of using the Navigator tool is in loading a Jython script (file ending with .py). A right click and pull down to "Open With..." then "Jython Script Editor", will open the script up in an Editor view window (the Editor view can hold several, tabbed, windows). Scripts can also be run directly from the same menu, with the "Run Script" item appearing on the menu. Although the scripts need to be self-contained requiring no parameter inputs.

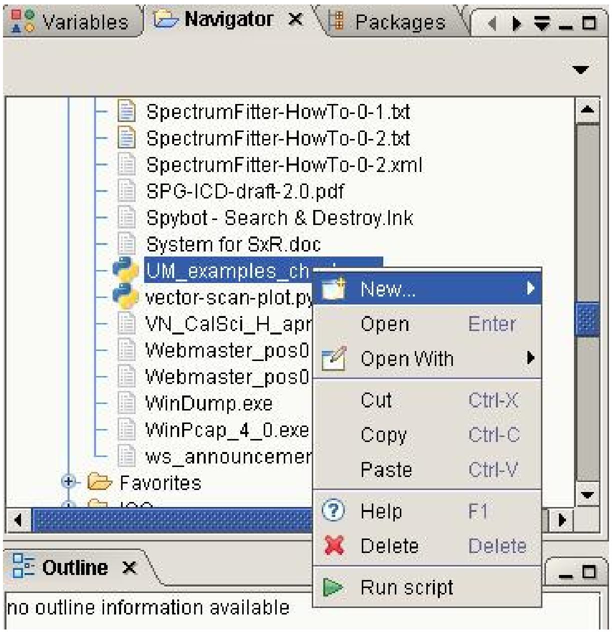


Figure 2.31. The Navigator view for HIPE showing the options available for the selected item on the user's system. Double-clicking on a ".py" script will open the script in a new Editor view window.

2.8.11. Outline

The outline information on a given variable is placed in this uneditable view. Clicking on the variable in the "Variables" view (see Section 2.8.14)provides an output of its name, variable type (class) and the herschel package in which this variable type is defined. In Figure 2.32, the DP session variable `myImage2` is shown to be an image dataset which could be viewed using the available DP Display task (for example).

Figure 2.32. Outline of a variable in the DP session is shown in the Outline view.

NOTE: This window provides information only, and its contents can not be manipulated by the user.

2.8.12. Packages

This view brings up a panel that provides access to the packages that are currently available to the session (see Figure 2.33. In order to get further information on what is available in a given package the user can double-click on one of the folders displayed. Package documentation associated with the available commands () can be obtained by clicking on the command or right click on the item in the Package view and pulldown to "Help". Documentation appears in the "Topic Help" view. Note that the documentation provided at this level is not for the general user but more for those wishing to use to use package elements to develop scripts etc. within the HCSS.

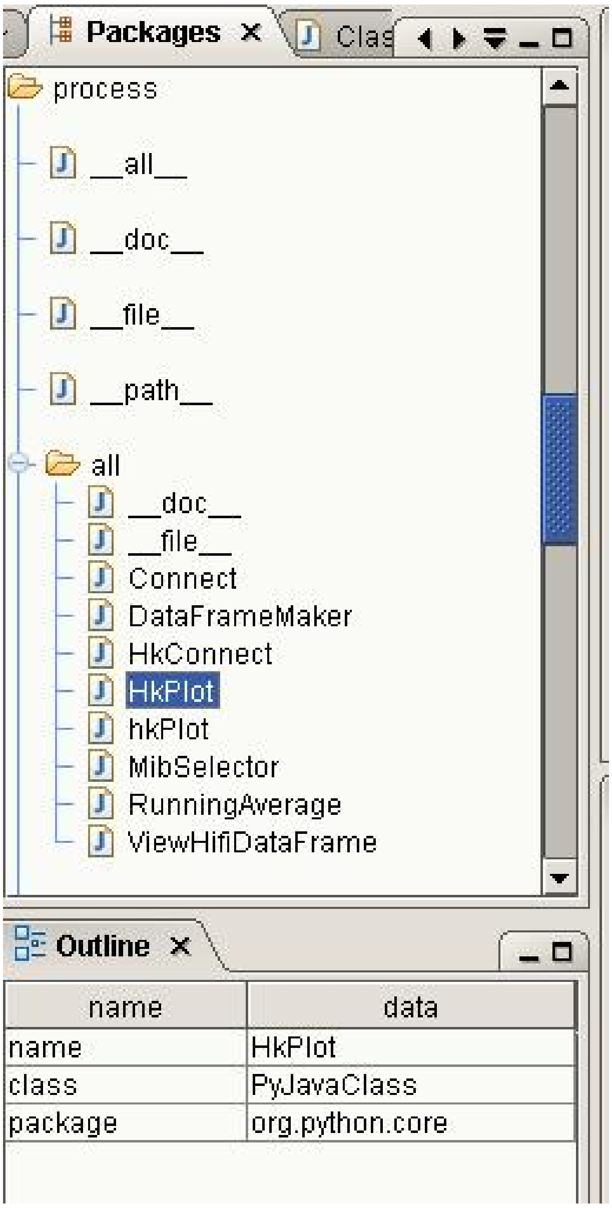


Figure 2.33. The Package view which is one of the tabbed views to the top left of the default Workbench perspective. Information on a package item is shown in the Outline view.

Information on the package item is provided in the Outline view when the item is highlighted.

2.8.13. Tasks

This provides a list of tasks and tools available to the user from HIPE. These can be applied to variables of the appropriate type in a session. A right click on the available task allows a menu with a pulldown that includes "Open With...". This allows a Task dialog where the task can be applied to a given set of data. Some example workflows are given below.

When a variable (see Variable view, below) is highlighted, available tasks appear as available in the "Tasks" listing. The Tasks are available in three folders; All, Applicable, and By Category. The "All" folder shows all available tasks in the system, while "Applicable" tasks are those that are designed and registered to run with data of the type associated with the highlighted variable in the Variables view. The folders can be opened or closed with a double-click. To start a task working on the data variable highlighted, simply double click on the task shown in the list (see Figure 2.34).

Figure 2.34. Tasks available for a given DP session variable are automatically made available in the "Tasks" view. Applicable tasks are shown in the Applicable Tasks folder.

Most of the tasks needed for basic data analysis can be accessed in this fashion. More information on how to use tasks for general data analysis is provided in the set of HowTos that are available in the main Help window (e.g., go to "Help" in the main toolbar at top left of HIPE, which opens up a window with access to the full user documentation).

Double-clicking any task in the Tasks view brings up a GUI dialog in the Editor view. This can be used to run the task in the appropriate way. In all cases an "Accept" button, to bottom right of the dialog, under the progress bar, should be clicked to run the task with the given inputs (see example task dialog at Figure 2.26).

2.8.14. Variables

This view shows the variables established in your session that you can use. You can always see what they are in two ways.

- Click on the variable in the Variables view. It's description and outline are shown in the Outline view (see Section 2.8.11).
- You can print the contents to the screen in the Console view (see Section 2.8.2) by the command

```
print <variable name>
```

Clicking on a variable in the Variables view enable you to see what type of variable it is (this appears in Outline view, Section 2.8.11). In this way it is possible to look at the structure of a complex item in your session containing multiple groups of spectra or images.

A right mouse click on a variable allows a short menu to appear which provides the possibility to do the following:

- Get help information on the variable type. The help information appears in a new browser window tab, and is the User Reference Manual information for the given variable type(see Section 2.4).

- Delete the variable from the session. Note that the equivalent command-line will appear in the Console view (see Section 2.8.2).
- "Open with" allows a list of ways to view the variable other than in outline (e.g., if it is a table you can use a Dataset Viewer, see Figure 2.35 or Spectrum Viewer for spectra). These viewers currently provide output in the Editor view (see Section 2.8.4).

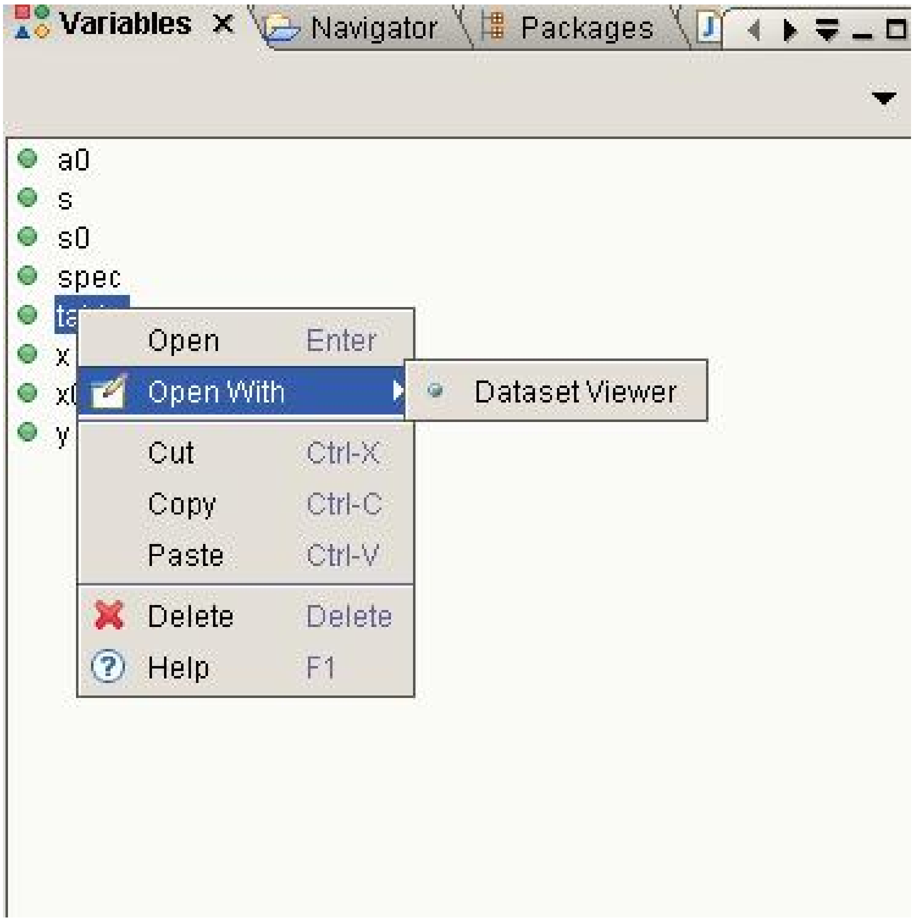


Figure 2.35. Tools available for a given DP session variable are automatically made available in the "Tasks" window.

2.8.15. Welcome

Opens up the initial startup window.

2.9. Viewers in HIPE

A convenient feature of HIPE is that recognizes the type of variables held in a session (is it a dataset, a spectrum, an image, a scalar constant etc). Items appearing in the "Variables" or "Outline" views, *with a green dot beside to their left*, can potentially be opened. A right mouse click on any of these variables appearing in a DP session will provide a small menu of options allowing the user to "Open" the variable or "Open With..." or "Delete" or get help on ("Help Selection") the variable chosen. As previously noted in the "Variables" section.

Choosing "Open" allows opening with the first item in a list of available viewers for the selection. But there can be more than one viewer. These are shown under "Open With...". One of the viewers is chosen as the default for a double-click on the variable -- and this is shown with a dot beside it.

An example is shown for `SimpleImage`. A right-click on a variable of this type in the "Variables" window shows there are two viewers (see Figure 2.36). The Product viewer will show associated metadata and array values while the second viewer displays the image (more is provided in the HowTo on Display and Manipulation of Images).

As examples, viewers are available to show information on headers (metadata), and datasets (numerical arrays), enable table plotting and exploration, show images and/or spectra.

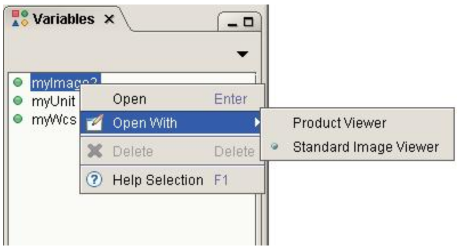


Figure 2.36. Available viewers are shown with a right-click.

Chapter 3. HowTo Access and Retrieve Data from the Herschel Science Archive

3.1. Introduction

The Herschel Science Archive (HSA) is the main repository for the observational data products from Herschel. It is available via a web interface to the Herschel Science Centre. But it is also available directly from within a DP session using HIPE.

In this HowTo we explain how to access data and bring it into a DP session.

3.2. Retrieving Data from the Herschel Science Archive User Interface

Data can be accessed directly from the HSA using the Herschel Science Archive User Interface (HUI; see Figure 3.1)

Figure 3.1. The Herschel Science Archive interface.

In order to retrieve data from the HSA you have to be a registered Herschel user. To register with the Herschel system please go to: Herschel Archive Registration and follow the appropriate instructions. This registration page is also accessible through the "Login/Register" page of the HUI ("Register as New User"; Figure 3.2).

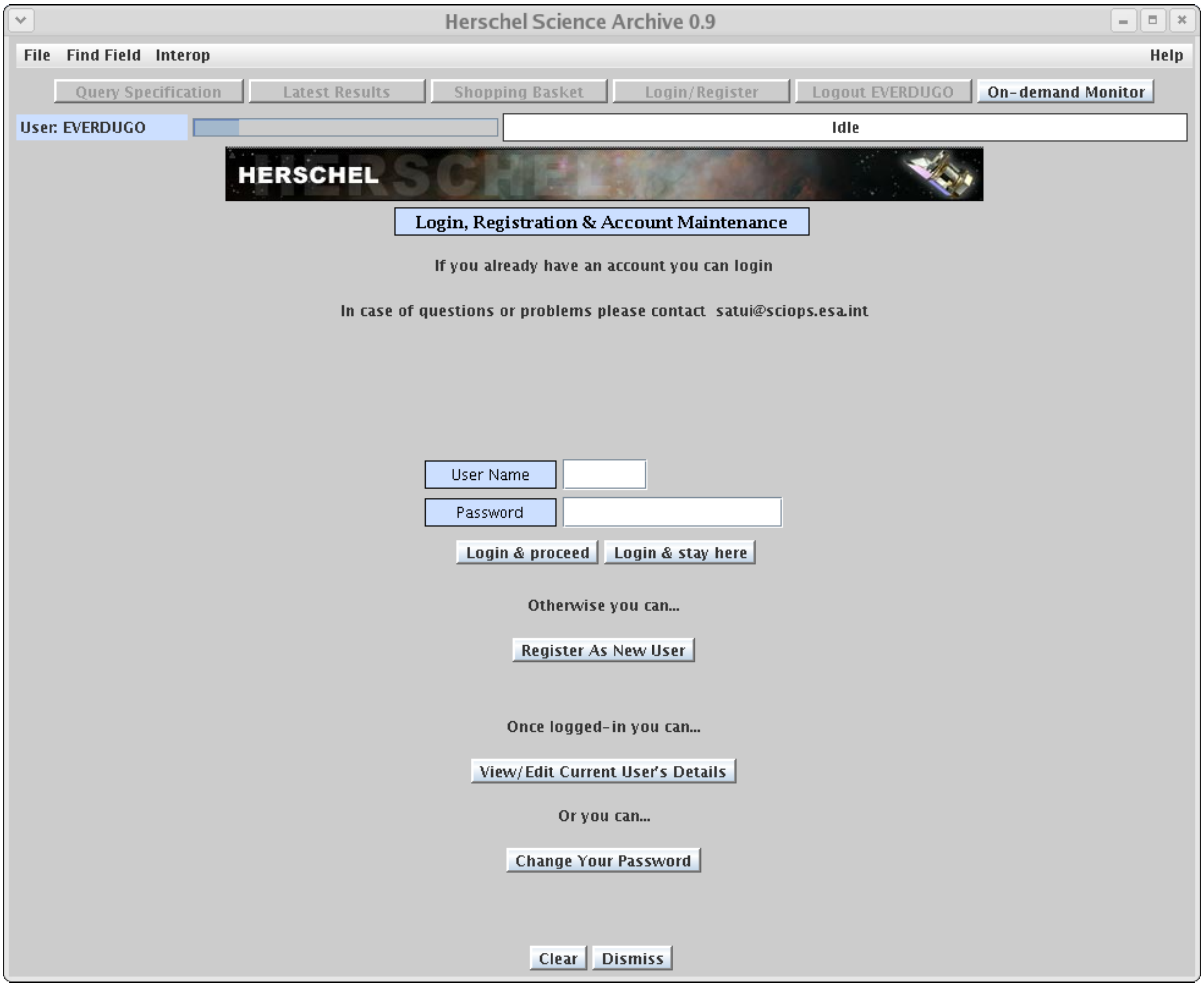


Figure 3.2. Login/registration in the HSA.

You do not need to register if you intend solely to browse the content of the archive. Registration is, however, a precondition to retrieve data from the HSA.

Only the PIs can access data covered by proprietary rights. The same rule applies to the viewable quick-look products of observations, as well as to proposal-related files. They can be viewed by the observation PI only, provided s/he has logged in with her/his registration identifier.

After executing a query in HUI, the user is automatically moved to the "Latest Result" page (see Figure 3.3) which contains the list of observations which match the query. A direct download button ("Retrieve") is available in this page, which starts the ftp retrieval of the selected set of products. The usage of it is only recommended for individual observations.

Figure 3.3. Result of query of the HSA

The "Shopping Basket" (Figure 3.4) page allows to define the set of data to be retrieved for a group of several observations.

Figure 3.4. The shopping basket of data to retrieve from the HSA

Once the shopping basket has been filled in with all the data intended to retrieve, the user is asked to finally confirm the choice, by submitting the data retrieval request. The generation of the dataset to be retrieved is then started, see Figure 3.5.

Figure 3.5. Data retrieval request in the HSA

One or more tar files are automatically transferred to an ftp area. In the HSA all access is via a password protected ftp area, since most data is proprietary. When they are available for retrieval, the user will be informed via e-mail on how to download them.

The tar file with the data retrieved from the HSA contains FITS files ordered in a well "jerarchized" directory structure. is decompressed in a user directory, it can be registered in HIPE as a pool.

3.3. Accessing HSA Data within HIPE

To access the HSA from HIPE the user simply accesses the Herschel Science Archive icon on the "Data Access" page or selecting the "Herschel Science Archive" view via the "Windows" pulldown on the HIPE menu (see Figure 3.6).

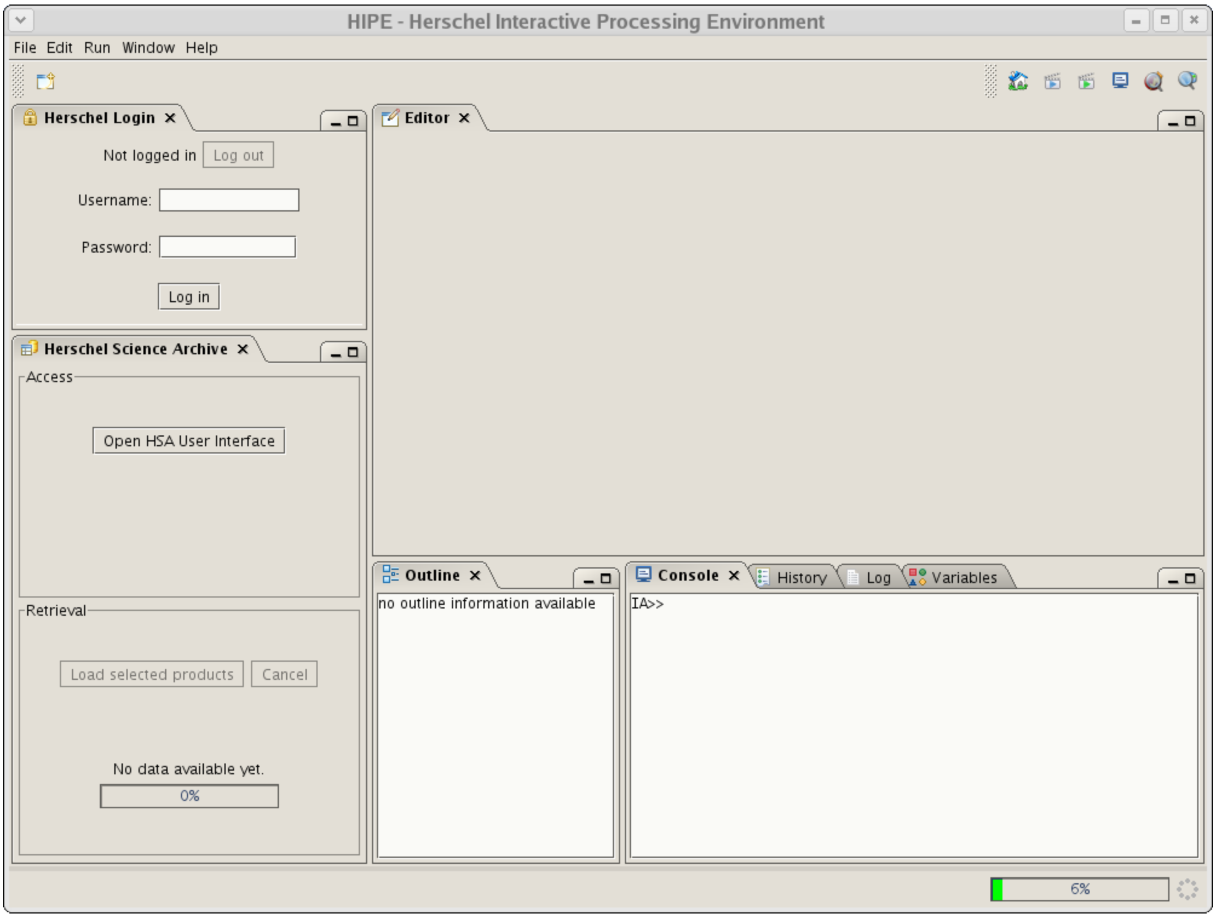


Figure 3.6. HIPE perspective for the HSA

Once logged in, click on 'Open HSA User Interface" to access data in the HSA. Note that if an HUI was opened before starting HIPE, opening a new one is not needed as the Plastic connection will be established automatically between them. Query the HSA for the data to be retrieved and select the products from the "Send to External Application" pulldown menu, either in the "Latest Result" page or in the "Shopping Basket" page. After a few seconds, the number of products selected will be automatically displayed in the "Retrieval" window of HIPE (as in Figure 3.7)

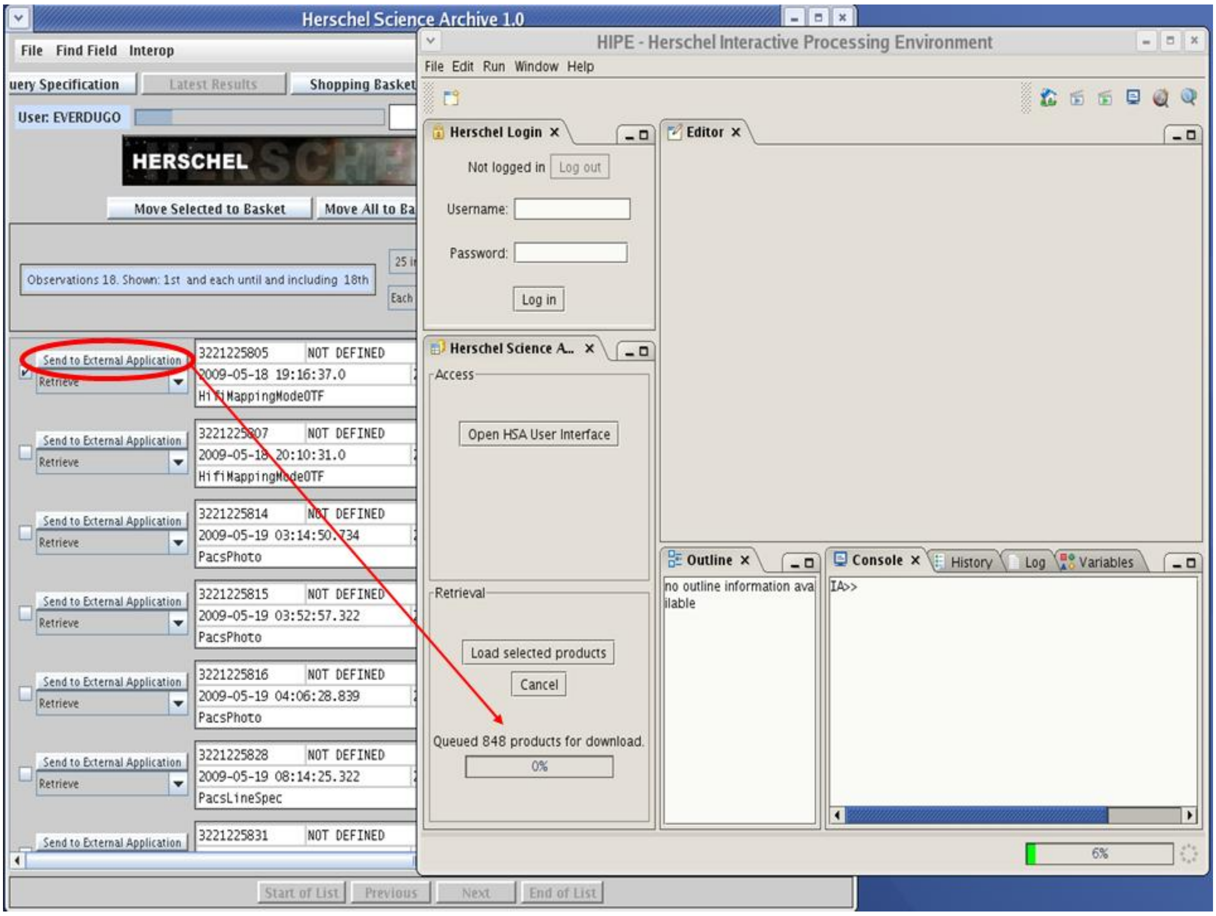


Figure 3.7. Retrieving observations from the HSA into a HIPE session.

Press "Load selected products" and the data will start to be loaded into HIPE (see Figure 3.8). During loading an indicator is shown in the HIPE display indicating that loading is taking place and the system is busy.

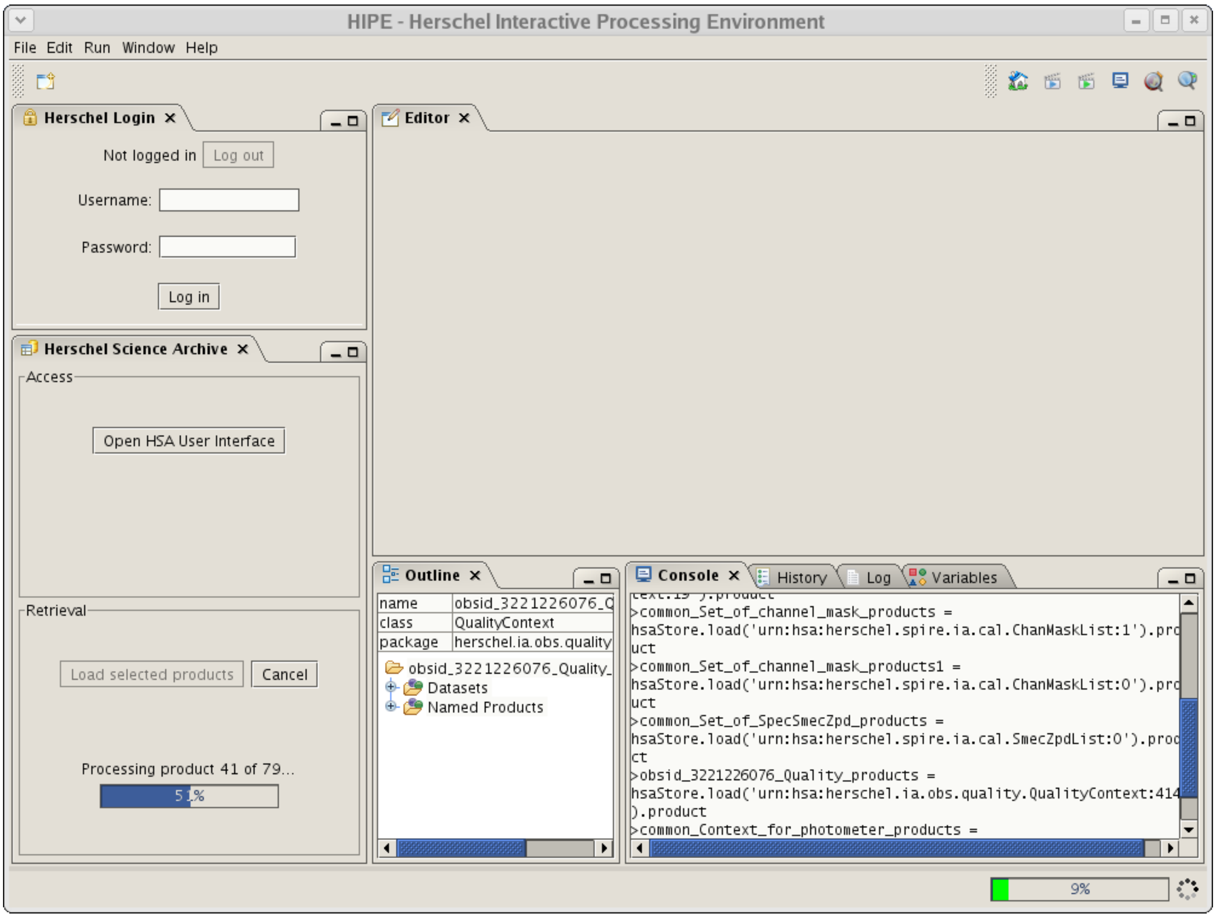


Figure 3.8. Product loading into HIPE from the HSA.

Note that in this way the data is not stored on a user's machine, it is referenced for fetching as needed within the user's working session. So this simply makes the data available in the HIPE session. Products can be inspected, analyzed, plotted, etc... Note also that for this, **the internet connection must be kept open**. Products can be saved/stored into pools later on (see chapter on "HowTo Save and Restore Data").

Chapter 4. HowTo Store and Access Data

Herschel Editorial Board

4.1. Introduction

This HowTo chapter describes the means by which data can be accessed and stored and using the HIPE interface. It should be noted that reading and writing of FITS data is held in a separate HowTo.

In order to access remote areas of data storage users must be on the internet. For access to data in the HSA users must have an appropriate username and password. Full use of the HSA is discussed in the HowTo on Archive Access. Users of the HSA will be allowed access to their own data as well as publicly available data within the HSA.

Users can store their data locally in data product pools which can be accessed for reading through the Data Access area of HIPE. This means that the user's stored data (processed or unprocessed) can also be selectable by queries that can become quite sophisticated.

4.2. Creating and Saving Products in a Pool

Any product (an example being a complete observation in the form of an ObservationContext) can be placed in a pool, or storage area, on the hard disk of the user. The setting up of various types of storage is discussed in Chapter 12 of the "DP Basic User's Manual" available as part of the release documentation. For this HowTo we will simply illustrate how to set up a set of stores (which act a bit like mini databases) in which a user can place any output data that is in the form of a product, such as an observation.

A pool can be set up and populated in the following fashion via the command line.

```
store = ProductStorage() # "store" is the name (handle) we will use
                        # in our session to place things
myPool = LocalStoreFactory.getStore("myTestPool") # "myTestPool" is actually
                           # the directory name where the store will be
store.register(myPool)    # now link it/register it under the name of store
store.save(prod1)     # ....and now we merrily add our products called
store.save(prod2)     # ...."prod1", "prod2" and "prod3" into the store.
store.save(prod3)     #
```

Note that if you start a new HIPE session you will need to register your pool again via something similar to the first three lines.

The directory on your disk where the data physically resides in the directory ".hcss/lstore" which you will find under your login directory. You will see that the information is actually held as a hierarchical set of FITS files that can be treated like a database, allowing us to query and search for data in similar ways to other databases.

4.3. Registering and accessing other data stores

It is possible to register other stores that can then be searched from the data access view, but they first have to be registered in the system (you need to tell the system where they are, in effect). For data stores elsewhere on your machine other than the default area this can be done by using the following lines of code which can be entered at the command line.

```
#import Configuration components into the environment
from herschel.share.util import Configuration
# get a local store (or create a new one if not already existing) with
# an id of "test". The Configuration command changes the directory
# where the store is
Configuration.setProperty('hcss.ia.pal.pool.lstore.dir', 'C:\\.hcss\\myData')
datastore=LocalStoreFactory.getStore("test")
myStore=ProductStorage()  # tell the system it is a store of products
myStore.register(datastore) # register it
# "myStore" is now one of the selectable data stores on the Data Access menu
mystore.save("myProduct")
# will save a Product in the DP session called "myProduct" in the storage area
```

Note

The process of registering/adding pools that the user can use in a session is expected to be made into a simplified tool in the future.

4.4. Data access via the HIPE GUI

Other than interactions with the Herschel Science Archive (which are discussed in another HowTo), access to data is via the Data Access view. The Data Access view is available via two routes within HIPE. The first is via the data access icon on the Welcome page of HIPE (see Section 2.5. The second route is via the "Windows" pulldown on the HIPE menu. Go to "Set View" and pull down to "Data Access". This brings up the page shown in Section 2.8.3 described in the introductory section of the HowTo documentation.

4.4.1. Types of Stored Data

All data is stored in the form of `Products`. These products are kept as FITS files on the local computer system, but are organised into pools of Products/FITS files. This allows querying on the contents of the `Products` (e.g., the metadata or header information). The `Product` wraps information such as images, spectra or tables of data into a storable component. An example is a single Herschel observation (which actually has several products wrapped up into one).

When the user obtains data from the Data Access view it enters the DP session as a `Product` and an overview can be obtained using the "Product Viewer" via a right-click on the name of the product in the session (see information on viewers in the HIPE overview chapter). Datasets such as tables or spectra contained inside the products can be accessed and viewed using Dataset or Spectrum Viewers as described later in this HowTo.

4.4.2. Using the Data Access View

When selecting the Data Access view the user will have certain "pools" of data available. These allow access to data stored in registered data storage areas (basically areas accessible to the user on his/her own computer or via the internet to another computer). Storing data in user-named pools is described in Section 4.3. All pools currently need to be explicitly "registered" to tell the system where to look.

4.4.2.1. Using the Data Access View to Query for Products

There are several ways of searching through your stores of data to get the products you want. You can search for complete observations -- such as those you are PI on which exist in the Herschel Science Archive -- attriibutes or metadata values, or you can go into data mining which involves searches based on the data itself.

For all cases, setup of the data query can be done based on observation data, the attributes of data, meta data or all data (data mining). Once the query of the data store has been set up the search can be done by clicking the Search button to the bottom right of the Data Access view. If the user wishes to access all available data in a data storage then this can be obtained by placing nothing in any of the input boxes of the query.

When the search button is clicked the equivalent command-line version of the request appears in the Console view (see Section 2.8.2). This can be saved and edited and used in batch mode processing. This helps to avoid syntax errors by the user in setting up queries on data stores.

Doing a Search

In order to do a search the user needs to do the following.

- Open the "Data Access" view.
- Select an available pool from the pull-down menu at the top of the view next to the word "Query". If none are available (greyed-out) then you need to first register a pool for access (see earlier sections of this chapter).
- After inserting an appropriate query, click on the "Accept" button to bottom right of the view. Note that if nothing is placed in the query then the total contents of the pool will be obtained. This is a good way to see the total contents of a pool.

Search by Observation

In this case we are dealing with high-level information. The data is part of certain proposal or uses a particular instrument on a particular day. Clicking on the "Observation" tab in the Data Access view allows searches at this level based on instrument, proposal ID, proposal name, observation ID (unique observation numbers or operational day (See Figure 4.1).

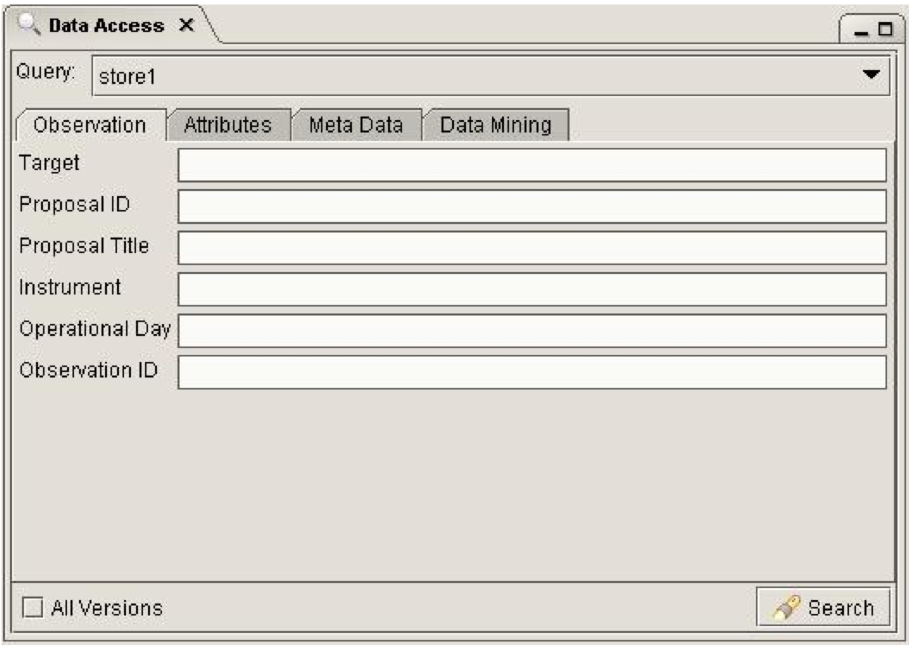


Figure 4.1. HIPE store selection and panel for searching by information on stored observation information in a product.

Search by Attributes

The attributes of a set of data are standard to all (See Figure 4.2) and it is possible to do a search on values in this given set of attributes -- which are listed in the query interface.

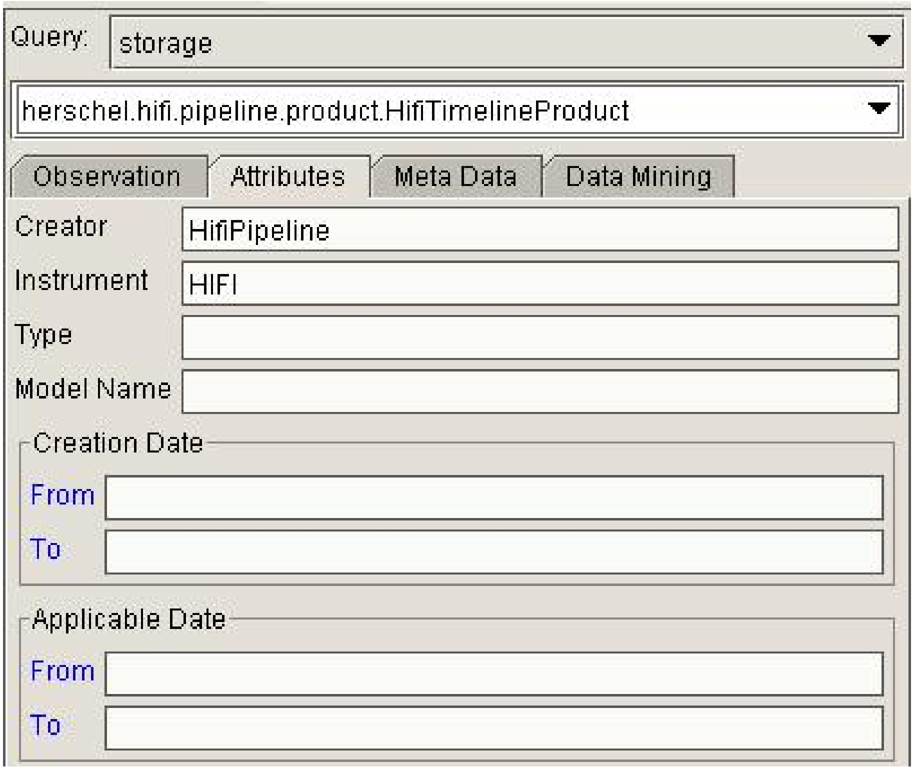


Figure 4.2. Attributes available for search.

Search by Meta Data

Meta data (like FITS header data) is data more specific to a given observation (See Figure 4.3).

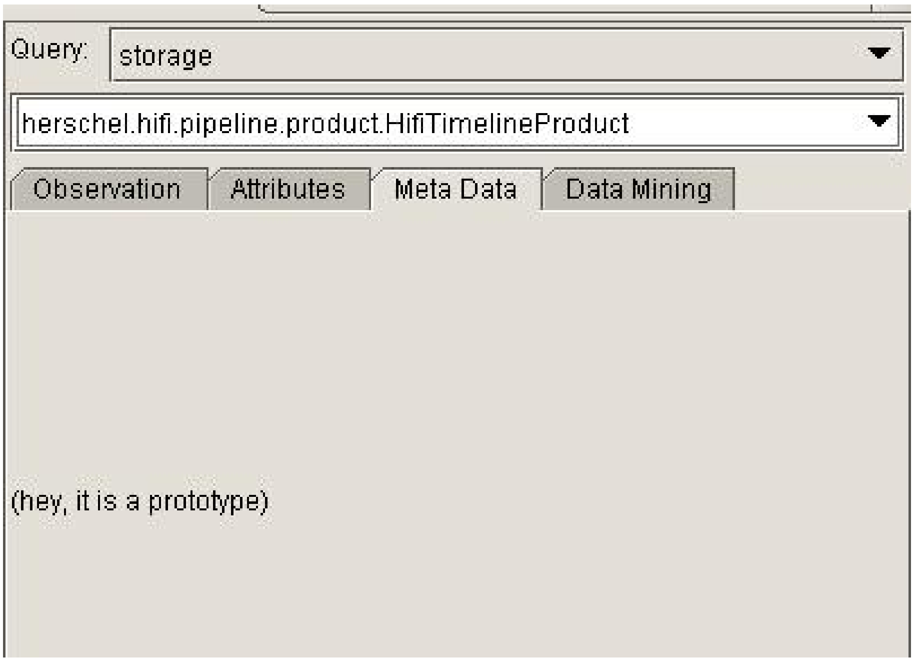


Figure 4.3. Metadata search.

Search by Data Mining

For data mining it possible to search on specific information contained within the science data itself rather than the meta data. YET TO BE IMPLEMENTED (See Figure 4.4).

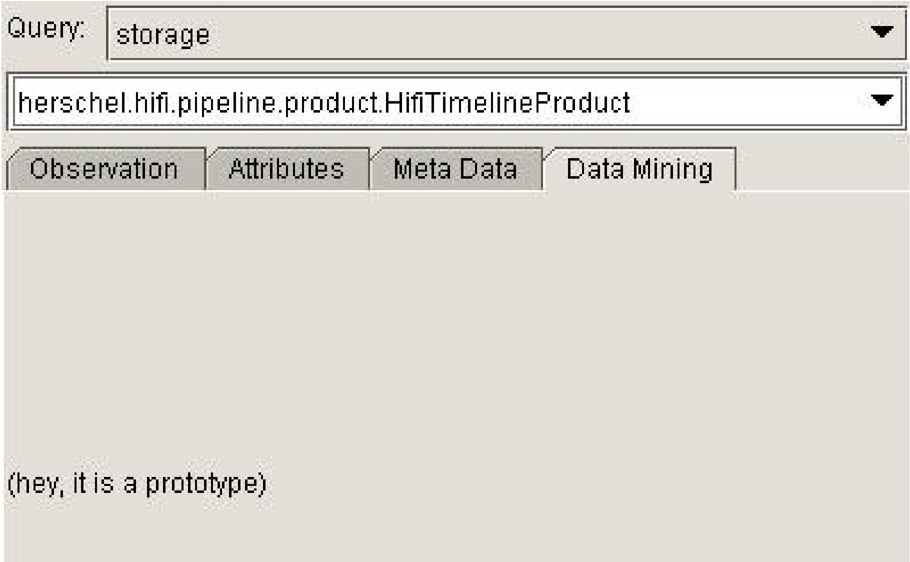


Figure 4.4. Search via data mining.

4.4.2.2. Output from a Query and Searching a Query Result

The output from the first query produces a result "QUERY_RESULT". This will be a group of products (e.g., observations) which can then be looked at by the user. The "QUERY_RESULT" name is highlighted in the Variables view (where the name can also be edited to something more appropriate if desired). This result is also automatically fed back to the Data Access pulldown menu, allowing for a search to be made on the result of the initial search.

The query output can be viewed by double-clicking on the result variable, e.g. "QUERY_RESULT" in the Variables view. This brings up the query results viewer in the Editor view part of HIPE. This lists the selected items. It also makes the outline available in the "Outline" view.

Clicking on one of the results shown in the query viewer extracts the chosen result (for example, the first product in the list is then available as "prod_0" in the session). Clicking on the name of this extracted product when it appears in the "Variables" view allows further assessment of its contents and viewing of any datasets it contains.

4.4.2.3. An Example of Search to Display of Data

In this case, we have partially processed some HIFI data to level 1, which has the format of a HifiTimelineProduct, and stored several versions of this processing in a store given the handle under the HCSS of "store1". This appears under the Data Access view pulldown menu as a selectable store item. The following now leads to displaying some data that has been extracted from our data store.

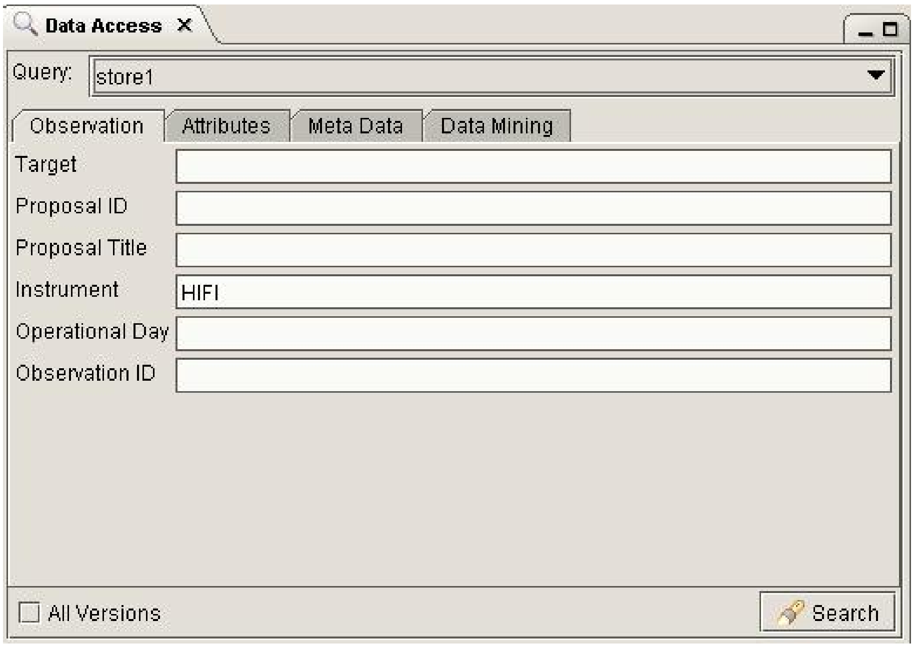


Figure 4.5. Set up of a query for data out of our store.

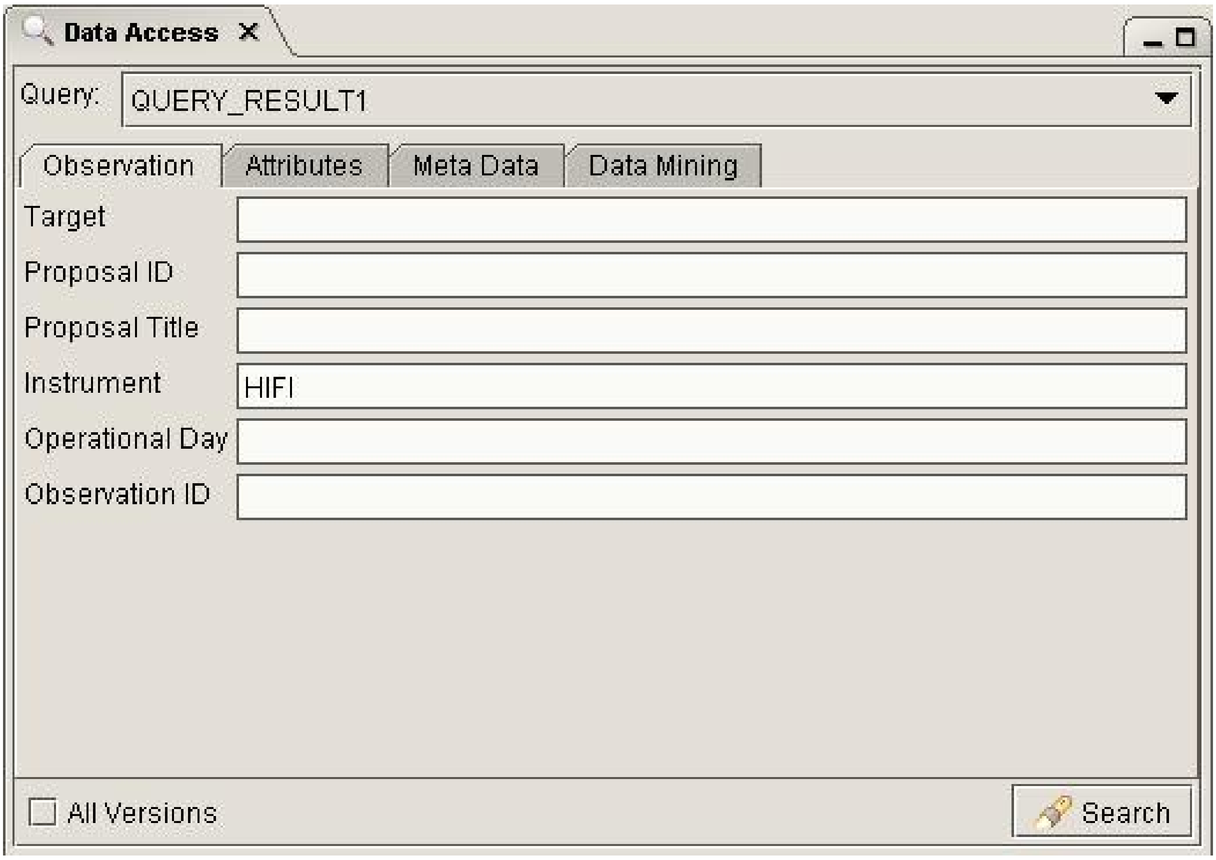


Figure 4.6. Query result obtained.

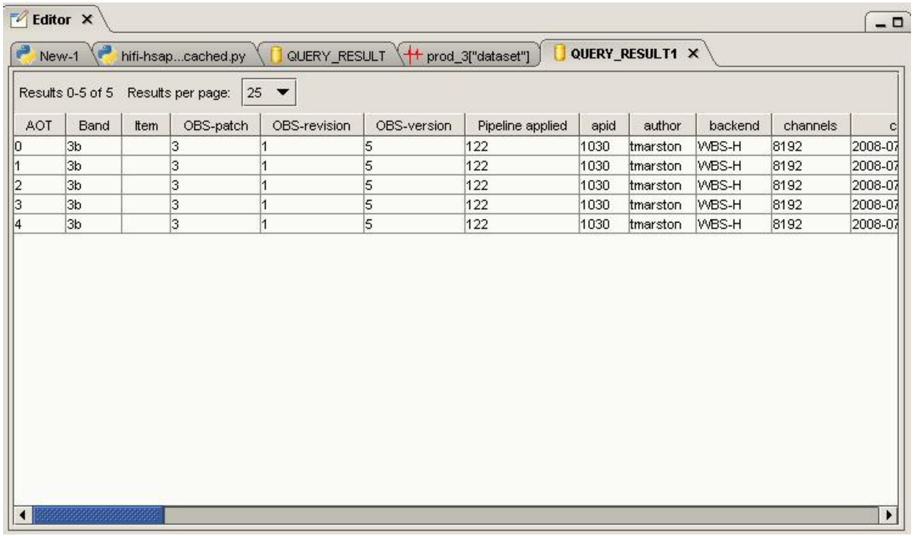


Figure 4.7. List of query results appear in editor window.

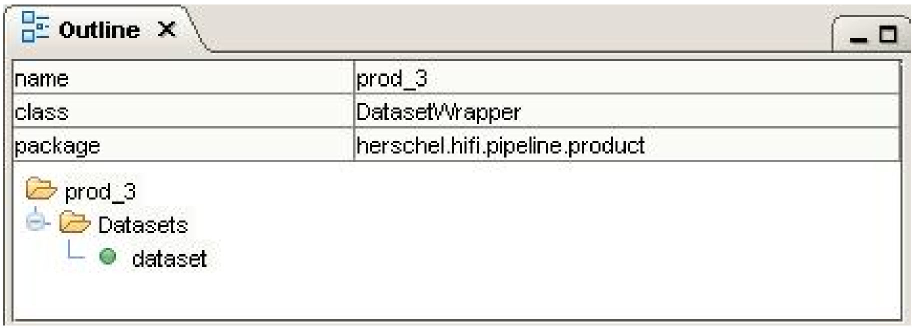


Figure 4.8. One of the items is selected with outline of contents shown bottom left.

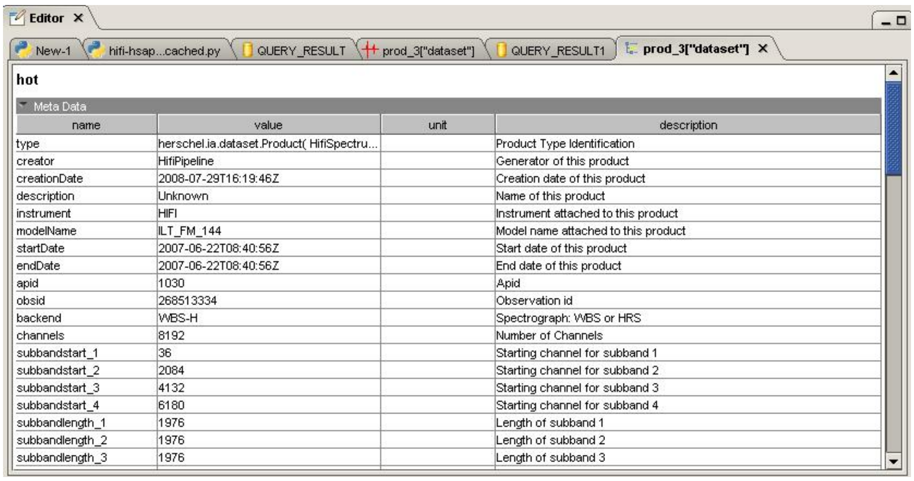


Figure 4.9. Metadata (header) display for the extracted spectrum.

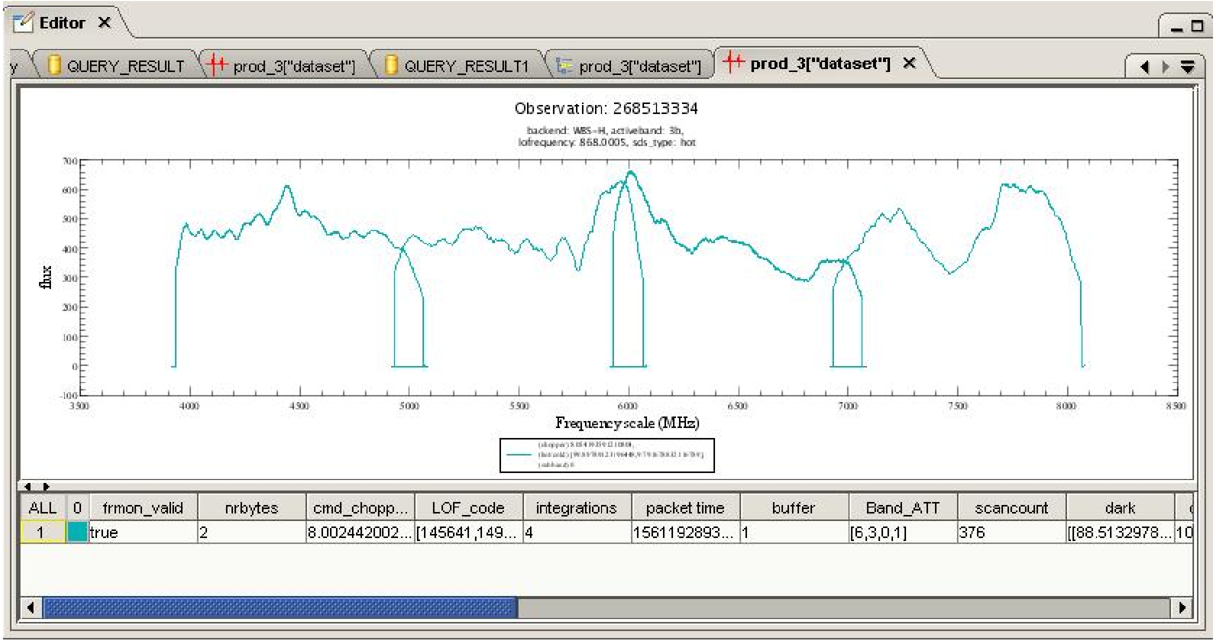


Figure 4.10. Displaying the extracted spectrum. Note that the view has been expanded using the capabilities of the "Spectrum Explorer" viewer.

1. We now intend to search for all HifiTimelineProducts (see second pulldown menu on the screen) with instrument=HIFI within this store by searching on these attributes. The setup should look like the screen shown in Figure 4.5.

2. Once this has been setup we click the "Search" button and the appropriate results are extracted and placed in a query result (see Figure 4.6). A highlighted "QUERY_RESULT1" (the number automatically placed at the end will increase depending on the number of queries you make) appears and the data access store available for querying -- at the top of the Data Access view -- immediately changes to QUERY_RESULT1 ready for further searching on the initial query results.

3. Select the query result in the Variable view (QUERY_RESULT1) via a double mouseclick. This provides a Query result viewer showing a listing in the Editor view of the query results items (see Figure 4.7).

4. Double-clicking on one of the results shown in the editor view creates the item (product) in the session. It allows us to pull out one of the selected products (e.g., "prod_3" for item number 3 in the query viewer) which can be manipulated in standard ways. For example, if we click on this product in the Variables view we get an outline of its contents in the Outline view (as in Figure 4.8).

5. We see that it shows a single folder in the Outline view. Clicking on the first folder, it opens up to show its contents which include a single dataset (as in Figure 4.8).

6. A right-click on the word "dataset" in the Outline view provides a set of viewer options. The Dataset viewer will show the associated metadata (header) information plus a table of various values associated with the spectrum, include flux/count values per channel (as in Figure 4.9).

7. Alternately, we can simply view the extracted spectrum dataset by selecting the "Spectrum Explorer" viewer instead (see Figure 4.10 and HowTo on displaying spectra).

4.5. Data Access via the Console View Command Line

Within the Console View (see Section 2.8.2) it is also possible to access data directly from the command line. The commands for doing this are actually generated in the Console view when using the dialog interactions noted above.

In the following we show how the information can be extracted into "newVariable1" (as per above example) which is then ready for display, fitting etc.

1. First we do a query on attributes in our data store, which was labeled "storage", looking for HifiTimelineProducts and instrument=HIFI. The following should all be on one line and the easiest way to get it is when it is copied to the Console view when using the Data Access view as noted above.

   ```
   QUERY_RESULT1 = StorageResult( store1,
    \
    herschel.ia.pal.query.MetaQuery(herschel.hifi.pipeline.product.DatasetWrapper,
    \"p", "(p.meta.containsKey(\"instrument\")
    \ and p.meta[\"instrument\"].value == \"HIFI\")") )
   ```

   We can always print to the screen the contents using

   ```
   print QUERY_RESULT1
   ```

   Queries on the observation or meta data windows are MetaQuery's rather than AttribQuery's.

2. Now we extract the third in the list of results found.

   ```
    prod_3 = QUERY_RESULT1[3].product
   ```

   Again, we can use the print command to see its contents. Which is actually several products (in this case 5).

3. Get the dataset out of the product.

   ```
    data_3 = prod_3["dataset"]
   ```

   The Outline view of data_3 will show it is of the form WbsSpectrumDataset and that it is a HIFI pipeline product.

   A full explanation of how to handle displays and manipulations (arithmetic and fitting) of spectrum (and image) datasets are covered in other HowTos.

Chapter 5. How to run the HIFI pipeline in HIPE

HIFI Editorial Board

5.1. Running the HIFI pipeline

The HIFI data processing pipeline is used for processing data received from one or more of the four HIFI spectrometers on-board Herschel into a final product that is suitable for interactive analysis.

The HIFI pipeline can be run within the Standard Product Generator (SPG) of the Herschel Science Centre (HSC) or from the Herschel Data Processing interface (HIPE or JIDE) in a user's interactive session. It is designed to: obtain raw data from a database (or local store of data); remove instrument-related properties of the data; calibrate the resulting spectra; and, then combine the separate spectra from a single observation. The final product is dependent upon the observation mode and is either a calibrated spectrum, a set of co-added spectra or spectral 3D cubes. For more information about the pipeline steps and their results, please read the HIFI Standard Product Specification Document, or the HIFI Pipeline Specification document.

Calibration information is required in order to process data from the ICC through the pipeline. You can access this information through the Versant databases at the HIFI ICC or install the `hifi-cal` database locally on your machine.

5.2. How to run the HIFI pipeline in HIPE

We can run the HIFI pipeline from within HIPE in the following fashion.

- Select the "Hifi Pipeline" from the HIPE Window menu (under Show View), if you already have data for processing loaded into the session. Otherwise, double-click on the "hifiPipeline task" under the Hifi Category in the Tasks pane.

Figure 5.1. Starting the HIFI pipeline task. Select "hifiPipeline" from the Windows menus or the task view. Once a HIFI observation context has been selected in Variables view then the "hifiPipeline" task is in the folder of Applicable Tasks.

- The default view allows you to do the following.
- Re-process an already existing observation context, e.g. from the Herschel Science Archive, through the pipeline. You do this by dragging the name of the data to be reprocessed from the Variables view to the observation context bullet in the dialog that appears in the Editor view.
- Select the spectrometers you wish to process data for by checking the desired instruments (Wide Band Spectrometer, WBS, H or V polarization; High Resolution Spectrometer, HRS, H or V polarization).
- Process data from the ICC database by typing in an obsid and database name (NOTE that this is only available for astronomers who have permission to access ICC databases. Only read access is possible).
- Select which level to (re)process the dat to (0, 0.5, 1 or all levels). The default set-up will pipeline data through all levels.

Figure 5.2. HIFI pipeline task: default view. General users would be expected to set up the pipeline this way.

By clicking on the "expert" button, you may control more detailed aspects of the pipeline set-up. The following items are available to experts only and are generally expected to be used only by HIFI calibration scientists.

- ProcessEnviroment is used from SPG (Standard Product Generation), and is another way to determine where the data (including calibration data) is to be retrieved from.
- If you wish to use your own calibration products store (e.g., held as a local store in a location on your hard disk), then the hifi-cal local store can be set as a property (in "myconfig" file) and the "hifi cal" box should be left blank.
- For HIFI ILT data analysis you need to set the observing mode.
- To use a self-defined "palStore" (Product Access Layer data store) in your session, drag its name in from the variable list.
- Define the test environment by typing in the telemetry version, "tmVersion" (e.g., ilt-fm or ist-fm).
- The execMode must be set to "INTERACTIVE" in order that the resulting ObservationContext is stored in your "palStore".
- Select "true" in the drop-down menu to clear the cache store.
- You can select your own pipeline algorithms if you wish to change how the processing is to be done. At present you must drag these in as variables but in the future it will be possible to read in files here.

You can toggle back to the default view by clicking on the "basic" button.

Figure 5.3. HIFI pipeline task: expert view. Expert users can set up the pipeline from this view.

Once you have everything set, choose the name of the observation context that will be produced (or use the HIPE default), and click on accept to run the pipeline. A progress bar charts the status of the `hifipipeline` task. Note that HIPE may freeze for a few moments while data is being accessed from the database.

5.3. Running the Individual Pipelines

The `hrs` and `wbsPipeline` tasks can be run individually to process data to level 0.5. The `genericPipeline` task can then be run to generate up to a level 2 product.

These pipeline tasks are run and set-up in the GUI in much the same way as the `hifiPipeline` task. Note, however, that an obsid and database cannot be passed to the `genericPipeline` task. Instead an HTP or ObservationContext resulting from one of the HRS or WBS pipelines should be passed to it by dragging it from the `Variables` list.

Figure 5.4. Running the `wbsPipeline` task. The ObservationContext, "obs", generated can be passed to the `genericPipeline` task.

Alternatively, you can (re)process a level 0.5 (1) product up to a level 1 (2) product using the `level1Pipeline` (`level2Pipeline`) task. The GUI interface is exactly the same for these pipeline tasks as for the `genericPipeline` task.

Note that you can also (re)process an observation context up to levels 0, 0.5, 1 and 2 using the `uptoLevel...` buttons in the hifiPipeline task (see Figure 5.2).

5.4. Running the Pipeline step by step

The individual steps of the HRS, WBS and Generic Pipelines are also found in the task pane when the HIFI Category is selected. By double-clicking on tasks you may use them to run through the pipeline step by step, in the order of your choosing. As an example, Figure 5.5 shows the GUI for the step in the WBS pipeline that subtracts the darks from the fluxes (DoWbsDark); three methods of dark subtraction can be selected from the drop-down menu. The HIFI Pipeline Specification document should be consulted for descriptions of each step of the pipeline, as well as the methods used and inputs required.

Figure 5.5. Running the pipeline step-by-step. It is possible to modify the steps taken at stages of the spectrometer pipelines.

Additionally, the `wbsPipeline` task has an option to be run step-by-step by selecting "true" from the step drop-down menu, (see Figure 5.2) .

5.5. How to run the HIFI pipeline from the command line

5.6. Using the HIFI Pipeline task

An easy way to run the HIFI pipeline is to use the hifiPipeline task from the command line. The hifiPipeline task calls the hrs/wbs pipeline task, to process data from Level 0 to Level 0.5, followed by the generic pipeline task, which processes data to Level 1 (and beyond). The backend (WBS or HRS) pipeline tasks produce a HifiTimelineProduct, called level 0_5. The generic pipeline results in an observation context, which contains all the products generated by the pipeline task and is stored in a "hifi-pipeline" lstore (~/.hcss/lstore/hifi-pipeline).

Below are several examples showing how to use the HIFI pipeline task from the command line.

The hifiPipeline task is called into JIDE or HIPE by:

`from herschel.hifi.pipeline.PipelineTask import *`

1. Run the pipeline and generate an observation context ('obs') from scratch (Figure 5.6):
   `obs = pipelineTask(obsid=268435583, db="ds1@iccdb.sron.rug.nl 0 READ")`

   If you do not specify the database with `db=...` then the default (var.database.devel) set in the user's properties will be used.

   Figure 5.6. Example 1: Running the HIFI pipeline task

2. You can use a GUI to show a progress bar (test only with no clear information, at the moment).
   `obs = pipelineTask(obs=None, obsid=268435583, db="ds1@iccdb.sron.rug.nl 0 READ", gui=1)`

   NOTE: To re-use the pipeline task, ensure that all the IO parameters are reset by setting `obs=None`

3. For HIFI ILT data it is necessary to provide the obsMode name - the data itself does not have it:
   `obs = pipelineTask(obs=None, obsid=268516902, \ db="ilt_fm_5_prop@iccdb1.sron.rug.nl 0 READ", obsMode="HifiPointModeLoadChop")`

4. Redefine the tmVersion if the selected database requires a different mission phase than the default in your binstruct property `hcss.binstruct.mib.pal.tm_version_map` (hifi default = "`ilt-fm`"). This is only applicable when using pre-launch test data:

```
obs = pipelineTask(obs=None, obsid=268439922, \
db="ilt_par_5_prop@iccdb.sron.rug.nl 0 READ", tmVersion="ilt-par")
```

5. The pipeline task automatically processes data from all four spectrometers. You can select an apid for processing to Level 1 (for now, processing to Level 0 always includes all available apids):
   ```
   obs = pipelineTask(obs=None, obsid=268516902,
   apids=["1030"], \ db="ilt_fm_5_prop@iccdb1.sron.rug.nl 0 READ",
   obsMode="HifiPointModeLoadChop")
   ```

6. The pipeline can be re-run in various ways:

   a. Re-run the pipeline, assuming Level 0 is available:
   ```
   pipelineTask(obs=obs)
   ```

   b. Re-run for Level 0 too. (Note that all calibration and other products are not replaced):
   ```
   pipelineTask(obs=obs, reprocessAllLevels=1)
   ```

   c. Or just re-generate Level 0:
   ```
   obs = pipelineTask(obsid=268516902,
   obsMode="HifiPointModeLoadChop", uptoLevel0=1)
   ```

   d. You can then use this to process from Level 0 up to Level 0.5:
   ```
   obs = pipelineTask(obs=obs, uptoLevel0_5=1)
   ```

7. You can edit the algorithm of the pipeline tasks:
   ```
   obs = pipelineTask(obs=None, obsid=268435583,
   db="ds1@iccdb.sron.rug.nl 0 READ", \ wbsAlgo=myWbsAlgo,
   hrsAlgo=myHrsAlgo,genericAlgo=myGenericAlgo)
   ```

   The algorithms can be found in:

   WBS: {build_root}/lib/herschel/hifi/pipeline/wbs/WbsPipelineAlgo.py
   HRS: {build_root}/lib/herschel/hifi/pipeline/hrs/HrsPipelineAlgo.py
   Generic: {build_root}/lib/herschel/hifi/pipeline/generic/GenericPipelineAlgo.py

8. And provide your own palStore to which the pipeline will write to:
   ```
   obs = pipelineTask(obs=None, obsid=268435583,
   db="ds1@iccdb.sron.rug.nl 0 READ", \ palStore = myStore)
   ```

9. Clear CachedStoreHandler to avoid a block due to none closed stores. Note, this closes ALL stores available in this cache and may affect other applications running in the session.
   ```
   obs = pipelineTask(obs=None, obsid=268435583,
   db="ds1@iccdb.sron.rug.nl 0 READ", \ uptoLevel0=1,
   clearCachedStoreHandler=1)
   ```

10. Finally, if pipeline task is not behaving as you expect you could try a reset:
    ```
    pipelineTask = PipelineTask()
    ```

The calibrated spectrum can be extracted from the resulting Observation Context and then viewed with several tools, such as SpectrumExplorer (see HowTo on Spectrum Display).

5.7. Running the Individual Pipelines

In addition to using the HIFI pipeline task, one can run the underlying pipeline tasks to, for example:

1. Pass an observation context (from running the HIFI pipeline task) to the WBS pipeline task to re-run up to Level 0.5 for only one apid

   ```
   from herschel.hifi.pipeline.wbs.WbsPipelineTask import *
   ```

```
newobs=wbsPipelineTask(obs=obs, apid=1030)
```

2. and then pass that to the Generic pipeline task

```
from herschel.hifi.pipeline.generic.GenericPipelineTask import *

newobs2=genericPipelineTask(obs=newobs, apid=1030)
```

3. Process a HifiTimelineProduct using your own HrsPipelineAlgo

```
from herschel.hifi.pipeline.hrs.HrsPipelineTask import *

newhtp=hrsPipelineTask(htp=htp, algo=myHrsAlgo)
```

4. Load dataframes and housekeeping (create an HifiTimelineProduct)

```
htp = wbsPipelineTask(obsid=268516902,
db="ilt_fm_5_prop@iccdb1.sron.rug.nl 0 READ"))
```

As these examples illustrate, both ObservationContexts ("obs") and HifiTimelineProducts ("htp") can be passed to these tasks. The parameters used in the Section 5.6 above can also be applied to these tasks.

5.8. Running the Pipeline step by step

Another way to run the pipeline is step-by-step using the three pipeline branches separately. This is the simplest way to view or modify the pipeline steps, which are contained in the following scripts:

The WBS pipeline is found in `$HCSS_DIR/lib/herschel/hifi/pipeline/wbs/WbsPipeline.py`
The HRS pipeline is found in `$HCSS_DIR/lib/herschel/hifi/pipeline/hrs/HrsPipeline.py`
The Generic (or AOT) pipeline is found in `$HCSS_DIR/lib/herschel/hifi/pipeline/generic/GenericPipeline.py`

As an example, take (again) the simulated HRS-H (apid=1030) HifiPointModeDBS observation with obsid=268435583 from the simulator data database (ds1).

1. Load the WBS pipelinescript into your JIDE editor. The obsid, apid and database must be entered manually into the script, see Figure 5.7.

Figure 5.7. Running the WBS Pipeline Script step-wise

2. Step through the WbsPipeline.py script until the end, or play the entire script with the run-all button. The output is a HifiTimeline product, which is stored in a simple Pool called `simple.wbspipeline`, it and all the intermediate calibration products can be viewed in the DatasetInspector, see Figure 5.8.

Figure 5.8. WBS Pipeline Products. The WBS pipeline products can be viewed in the Dataset Inspector in JIDE or via the Dataset Viewer in HIPE.

3. To run the Generic pipeline, load GenericPipeline.py into your JIDE editor and step through the script.

- The Generic pipeline requires that data have an AOT-like structure. Older obsids, such as gas cell data, do not have this structure and result in the error `herschel.ia.task.SignatureException: params: Null is not allowed`

- The GenericPipeline.py script does not (yet) store the results in a Pool.
- The HTP and other calibration products generated by the Generic pipeline can be viewed in the DatasetInspector, along with the calibration products from the WBS pipeline, see Figure 5.9.

Figure 5.9. Generic Pipeline Products. Calibration products can be viewed in the Dataset Inspector when the Generic pipeline is run step-wise.

4. The Generic pipeline script cannot be played with the run-all button but can be run from the command line as follows:

```
from herschel.hifi.pipeline.generic.GenericPipelineAlgo import runGenericPipeline
```

```
newhtp=runGenericPipeline(htp, None)
```

However when the Generic pipeline is run this way, no calibration products are generated and only the resulting HTP can be viewed in the DatasetInspector in JIDE or Dataset Viewer within HIPE, see Figure 5.10

Figure 5.10. Running the Generic Pipeline from the command line. The calibrated spectrum can be viewed in the Spectrum Explorer.

The spectrum in Figure 5.10 is viewed in the Spectrum Explorer from within HIPE (also available from the DatasetInspector within JIDE). See the Spectrum Display HowTo for information about using the Spectrum Explorer. Information on the DatasectInspector can be found in Chapter 4 (The DatasetInspector) of A Basic User's Manual Herschel Data Processing, which can also be found in the DP documentation.

Chapter 6. How To run Pacs pipeline within HIPE

Vanessa Doublier-Pritchard, PACS ICC

6.1. Introduction: Purpose of using HIPE to run the Pacs pipeline

The purpose of this How To is to tutor users with regard to running the PACS pipeline within the HIPE application. The PACS pipeline is particularly complex and requires either a careful step-by-step running of the individual recipes or running as a standalone application. Data are expected to be retrieved as FITS files either from the Herschel Science Archive or from a user-defined local pool.

Note

At this time, the full integration of the PACS pipeline has not been finalised into HIPE. We are thus unable to propose a full overview (Howto) of the PACS pipeline within the HIPE framework. However, each tasks and script is already available and can be run at the command line level. At a later date it is expected that the tasks and scripts will run within the HIPE Tools environment.

Constructed on 15 October 2008.

6.2. System requirements

This section describes the system requirements to run the PACS pipeline.

1. What is most important is where the tm (telemetry files) are stored and recovered from. Depending on your operating system you may need to download manually the tm files or be able to access the database directly via the Product Acess Layer (not implemented yet)

2. The Level0 files have to be stored on the local store define as Pool.

3. the amount of available of RAM should be at least 1G

6.3. Populating the Level 0

The PACS pipeline is run from Level 0 frames. Level 0 frames can be obtained by either Populating Local pools, or Browsing already existing Pools.

1. Populating Pools to extract Level 0 frames

   This Task extract the telemetry of a given Observation and split it into various browsable products: HK, Data, CAL etc ...

   - PopulatePacsPoolFromFile

     This Task extract from a given file the various browsable products and split them into HK , Science and CAL.

   - PopulatePacsPoolFromDatabase

     This Task extract from a given database the requested observation and split them into HK, Science and CAL

2. Browsing Product Archive

   If the Level 0 are immediatedly available in a Pool, then the data can be directly accessed by browsing using the Herschel Science Archive data browser or browsing a local store, if already downloaded onto the user's computer. The observations can be browsed using the OBSID or CreationDate

6.4. Loading the Level 0 frames and Starting the pipeline

- Loading Level 0 frames: AVG for PHOT mode and RAW for SPEC mode

To load the "frames" form which the pipeline will work is done either using the GUIs or the command lines. Here is an small programme that will allow you to populate your pool and extract the necessary files/frames and load them into Hipe memory as a starting point.

```
# Access data via GUI
# For instance -- if obtaining the information from a local store
# with the handle "store"
result = browseProduct(store)
print result.size()
# -----------------------
# Observation Context for first selected object -- can also be obtained by
#clicking on "result" in the Variables view of HIPE, then on the first product
#shown in the Outline window.
obsCont = result[0].product
# Now get the level 0
level0 = obsCont.level0
print level0
print level0.refs
#there are frames for the photometer images (AVG) or spectrometer images (RAW)
# in this case the PACS blue photometer
frames = level0.refs["HPPAVGB"].product.refs[0].product
print frames

# and associated Housekeeping data
hk = level0.refs["HPPHK"].product.refs[0].product["HPPHKS"]
print hk
# an example housekeeping value set
print hk["DM_CS1_RES_VAL"]

#
# Access data by command line directly into the "store"
#
query   = MetaQuery(ObservationContext, 'h', 'h.meta["obsid"].value == 0')
results = store.select(query)
print results
print results.size()
#
obsCont = results[0].product
```

-->

With these commands you can adapt to load the necessary frames for either PHOT mode (AVG frames) or SPEC mode (RAW frames).

1. Using the ObservationContext

   The Observation context allows you to browse the different product available in the level 0 and extract the necessary frame types which the pipeline starts from

2. Using the command line

   Using the command line allows you to extract simply and quickly the needed files for the pipeline

- Starting the Pipeline

  PHOT pipeline: using AVG frames

  SPEC pipeline using RAW frames

6.5. PHOT pipeline

Full PACS pipeline documentation explaining the tasks being performed in each module is provided elsewhere. Here we show the steps

6.5.1. Level 0 to Level 0.5

In order to apply appropriate calibrations we first need to provide a calibration tree of ****** information. This can be obtained from doing the following......****.

Level 0 to Level 0.5 Tasks are common to all photometer observing modes.

```
frames = findBlocks(frames, calTree=calTree)
frames = photFlagBadPixels(frames, calTree=calTree)
frames = photFlagSaturation(frames, calTree=calTree)
frames = photConvDigit2Volts(frames, calTree=calTree)
frames = photCorrectCrosstalk(frames, calTree= calTree)
frames = photMMTDeglitching(frames)
frames = addUtc(frames)
frames = convertChopper2Angle(frames, calTree=calTree)
frames = photAddInstantPointing(frames,pp)
frames = cleanPlateauFrames(frames, calTree = calTree)
```

6.5.2. Level 0.5 to Level 2

6.5.2.1. Point Source pipeline

Point Source data reduction steps

- Command: runPhotometerPointSource.py
- Step-by-step Tasks:

```
frames = photMakeDithPos(frames)
frames = photMakeRasPosCount(frames)
frames = photAvgPlateau(frames)
frames=photAssignRaDec(frames, calTree=calTree)
frames = photDiffChop(frames)
frames = photAvgDith(frames)
frames = photDiffNod(frames)
frames = photCombineNod(frames)
print "photRespFlatfieldCorrection"
frames = photRespFlatfieldCorrection(frames, calTree = calTree)
frames = photDriftCorrection(frames)
```

  # Level 1 > Level 2 ##

  f1 = photShiftDith(frames,copy=1)

6.5.2.2. Small Extended Source

Small Extended Source data reduction steps:

- DP script: runPhotometerSmallExtendedSource.py

- Step-by-step Tasks

  # Level 0.5 > Level 1 ##

```
frames = photMakeRasPosCount(frames)
frames = photAvgPlateau(frames)
frames=photAssignRaDec(frames, calTree=calTree)
frames = photDiffChop(frames)
frames = photAvgNod(frames)
frames = photDiffNodSmall(frames)
print "photRespFlatfieldCorrection"
frames = photRespFlatfieldCorrection(frames, calTree = calTree)
```

  # Level 1 > Level 2 ##

  image = photProject(frames,calTree=calTree)

6.5.2.3. Scan Map - simple-

Scan Map, default setup, data reduction steps:

- DP scripts to run: runPhotometerScanMap.py
- Step-bystep Tasks

  # Level 0.5 > Level 1 ##

```
frames = photMakeDithPos(frames)
frames = photMakeRasPosCount(frames)
frames = photAvgPlateau(frames)
frames=photAssignRaDec(frames, calTree=calTree)
frames = photDiffChop(frames)
frames = photAvgDith(frames)
frames = photDiffNod(frames)
frames = photCombineNod(frames)
print "photRespFlatfieldCorrection"
frames = photRespFlatfieldCorrection(frames, calTree = calTree)
frames = photDriftCorrection(frames)
```

  # Level 1 > Level 2 ###

  f1 = photShiftDith(frames,copy=1)

6.5.2.4. Scan Map

Scan Map, any setup, data reduction steps

- DP scripts: runPhotometerScanMap.py
- Step-by-step Tasks

  # Level 0.5 > Level 1 ##

```
frames = photFluxCal(frames)
```

  # Level 1 > Level 2 ##

```
frames = photAssignRaDec(frames, calTree=calTree)
frames = photHighpassFilter(frames, 200)
#Rem: Input paramters (scale =1 means skypix=dectector pixel)
#crota2 =0.0 of output map
scale          = 1
```

```
crota2         = 0.0
tod = makeTodArray(frames, scale, crota2, "test.tod", ".")
filterLength  = 0  maxRelError   = 1e6  maxIterations = 500
if (runNaiveMapper == None):
    runNaiveMapper = Boolean.FALSE
    map = runMadMap(tod, calTree, filterLength, maxRelError,
    maxIterations, runNaiveMapper)
```

6.5.2.5. Chopped Raster

Chopped Raster data reduction steps:

- DP scripts runPhotometerRaster.py
- Step-by-step Tasks:

  Level 0.5 > Level 1 ##

```
frames = photMakeDithPos(frames)
frames = photMakeRasPosCount(frames)
frames = photAvgPlateau(frames)
frames=photAssignRaDec(frames, calTree=calTree)
frames = photDiffChop(frames)
frames = photAvgDith(frames)
frames = photDiffNod(frames)
frames = photCombineNod(frames)
print "photRespFlatfieldCorrection"
frames = photRespFlatfieldCorrection(frames, calTree = calTree)
frames = photDriftCorrection(frames)
```

  Level 1 > Level 2 ###

```
f1 = photShiftDith(frames,copy=1)
```

6.6. SPEC Pipeline

Spec pipeline follows similar steps as the photometer pipeline. However, the operations are made on the ramp frames instead of the averaged frames.

6.6.1. Level 0 to Level 0.5

As for the Photometer Dp, the Level 0 to Level 0.5 Tasks are common to all observing modes. The output of the Level 0 generation may vary depending on the *readout mode of the observing mode: Ramps will be generated if the data are fitted, frames is the data are averaged (see PACS UM for details)*.

6.6.1.1. Ramps

If Level 0 is in the form of ramps. We give the tasks list for the Blue channel, but the same applies to the Red channel:

```
ramp_blue =compareRawWithReducedDataRamps(rampraw_blue,ramp_blue)
ramp_blue = specFlagBadPixelsRamps(ramp_blue, calTree=calTree)
ramp_blue = decodeLabel(ramp_blue)
ramp_blue = cleanPlateauRamps(ramp_blue, calTree=calTree)
ramp_blue = flagGratMoveRamps(ramp_blue, calTree =calTree)
ramp_blue = specFlagSaturationRamps(ramp_blue,pacsCalTree=calTree)
ramp_blue = flagDeviatingOpenDummyRamps(ramp_blue)
Optional: ramp_blue = subtractOpenRamps(ramp_blue)
Optional: ramp_blue = specFlagGlitchRamps(ramp_blue)
ramp_blue = specConvDigit2VoltsRamps(ramp_blue, calTree =calTree)
```

```
frame_blue = fitRamps(ramp_blue)
```

6.6.1.2. Frames

If Level 0 generates frames:

```
dmcHead_blue = extractDmc(pdfs, "blue")
frame_blue = compareRawWithReducedDataFrames(rampraw_blue,frame_blue)
frame_blue = specFlagBadPixelsFrames(frame_blue,calTree=calTree)
frame_blue = decodeLabel(frame_blue)
frame_blue = cleanPlateauFrames(frame_blue,dmcHead=dmcHead_blue, calTree=calTree)
frame_blue = flagGratMoveFrames(frame_blue,dmcHead_blue,calTree=calTree)
frame_blue = specFlagSaturationFrames(frame_blue,calTree=calTree)
frame_blue = flagDeviatingOpenDummySignals(frame_blue)
Optional: frame_blue = subtractOpenFrames(frame_blue)
frame_blue = specConvDigit2VoltsPerSecFrames(frame_blue,calTree=calTree)
```

6.6.1.3. Common to both Ramps and Frames treatment

```
Optional: frame_blue = addUTC(frame_blue)
frame_blue = convertChopper2Angle(frame_blue, zeroOffset=664,calTree=calTree)
#### Only for Test data not containing Ra and Dec ####
frame_blue = convXyStage2Pointing(frame_blue, seq,noInter=True)
frame_blue = specAssignRaDec(frame_blue, calTree=calTree)
########### End of Test data part
frame_blue = specFlagGlitchFramesQTest(frame_blue)
frame_blue = addObcp2Frames(frame_blue, seq=seq)
frame_blue = specExtendStatus(frame_blue, calTree =calTree)
frame_blue = findBlocks(frame_blue, calTree = calTree)
```

6.6.2. Level 0.5 to Level 2

As opposed to the photometer observing modes, the spectroscopic modes do not have systematic differences in the observing techinques. For matters of clarity and document consistency, each observing mode is treated separately, although only a few lines in each case will differ.

6.6.2.1. Chop Nod on a Star

- DP script: pacsspecchopnodstar.py
- Step-by-step Tasks

  #level 0.5 to 1 processing ####################################

  ```
  frame = waveCalc(frame, calTree=calTree)
  frame = specCorrectSignalNonLinearities(frame,calTree=calTree)
  frame = specCorrectCrosstalk(frame, calTree =calTree)
  frame = convertSignal2StandardCap(frame,calTree=calTree)
  frame = rsrfCal(frame, calTree=calTree)
  frame = specAvgPlateau(frame)
  frame = specDiffChop(frame)
  frame = specAddNod(frame)
  ```

  # Level 1 to Level 2 ####################

  ```
  cube = specFrames2PacsCube(frame)
  Default oversample: waveGrid = wavelengthGrid(cube)
  User set oversampling: waveGrid = wavelengthGrid(cube,oversample=3)
  rebinnedCube = specWaveRebin(cube, waveGrid)
  ## to be implemented ###
  projectedCube = specProject(rebinnedCube)
  ```

6.6.2.2. Chop Nod Mapping

- DP script: pacsspecchopnodmap.py
- Step-by-step Tasks

  #level 0.5 to 1 processing ####################################

```
frame = waveCalc(frame, calTree=calTree)
frame = specCorrectSignalNonLinearities(frame,calTree=calTree)
frame = specCorrectCrosstalk(frame, calTree = calTree)
frame = convertSignal2StandardCap(frame, calTree=calTree)
frame = rsrfCal(frame, calTree=calTree)
frame = specAvgPlateau(frame)
frame = specDiffChop(frame)
frame = specAddNod(frame)
```

  # Level 1 to Level 2 ################################

```
cube = specFrames2PacsCube(frame)
Optional: cube = spec3dDrizzling(cube)
Default oversampling: waveGrid = wavelengthGrid(cube)
User defined oversampling: waveGrid = wavelengthGrid(cube,oversample=3)
rebinnedCube = specWaveRebin(cube, waveGrid)
### to be implemented ########
projectedCube = specProject(rebinnedCube)
```

6.6.2.3. Chop Nod mapping with Offsets

- DP script: pacsspecoffmap.py
- Step-by-step Tasks

  #level 0.5 to 1 processing ####################################

```
frame = waveCalc(frame, calTree=calTree)
frame = specCorrectSignalNonLinearities(frame,calTree=calTree)
frame = specCorrectCrosstalk(frame, calTree = calTree)
frame = convertSignal2StandardCap(frame, calTree=calTree)
frame = rsrfCal(frame, calTree=calTree)
frame = specSubtractOffPosition(frame)
```

  # Level 1 to Level 2 #############################

```
cube = specFrames2PacsCube(frame)
Optional: cube = spec3dDrizzling(cube)
Default Oversampling: waveGrid = wavelengthGrid(cube)
User defined oversampling: waveGrid = wavelengthGrid(cube,oversample=3)
rebinnedCube = specWaveRebin(cube, waveGrid)
### To be implemented #####
projectedCube = specProject(rebinnedCube)
```

6.6.2.4. Wavelength Switching

- DP script: pacsspecwaveswitch.py
- Step-by-step Tasks

  #level 0.5 to 1 processing ####################################

```
frame = waveCalc(frame, calTree=calTree)
frame = specCorrectSignalNonLinearities(frame,calTree=calTree)
frame = specCorrectCrosstalk(frame, calTree = calTree)
frame = convertSignal2StandardCap(frame, calTree=calTree)
frame = rsrfCal(frame, calTree=calTree)
Optional: frame = subtractBackgroundFreqSwitch"
```

Level 1 to Level 2

```
cube = specFrames2PacsCube(frame)
Optional: cube = spec3dDrizzling(cube)
Default oversampling: waveGrid = wavelengthGrid(cube)
User defined oversampling: waveGrid = wavelengthGrid(cube,oversample=3)
rebinnedCube = specWaveRebin(cube, waveGrid)
```

to be implemented

```
projectedCube = specProject(rebinnedCube)
```

6.7. Conclusions

PACS pipeline is mostly implemented in the HIPE framework. Because the objective of the developers has been to implement a functional and stable pipeline to allow Calibration Scientists to test and quality control the instrument, registration of the different modules is not complete. This HowTo chapter will be updated as the modules become not just available but fully integrated within the HIPE framework.

Chapter 7. How to perform SPIRE pipeline processing in HIPE

Herschel SPIRE Editorial Board

version 1.0.0, 07-November-2008

7.1. SPIRE pipeline processing

This HowTo gives a step by step cookbook of how to run the SPIRE pipeline using HIPE.

7.2. Steps to running the SPIRE pipeline with the HIPE GUI

7.2.1. HIPE Welcome Window

Open HIPE by typing `hipe` in a command line (on Unix/Linux) or via the Start Menu on Windows. This will bring up HIPE's welcome window:

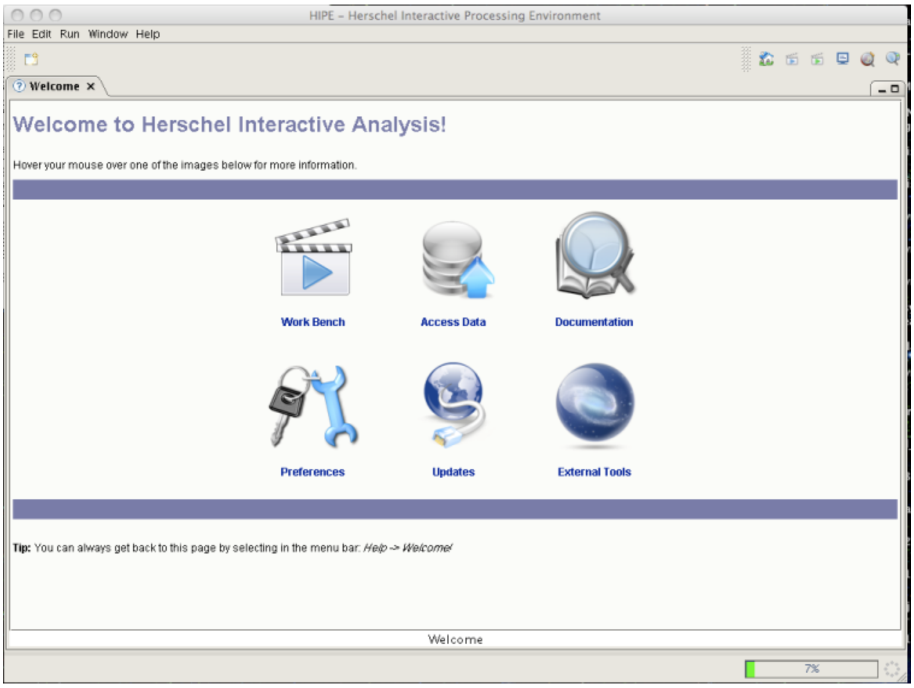


Figure 7.1. HIPE Welcome Window

Click on the work bench icon:

Figure 7.2. HIPE Work Bench Icon

7.2.2. HSA Log-in and retrieval

The user must ensure that they have retrieved their data and have stored it a location accessible by HIPE. Assuming that the user does not already have data to hand, the data can be obtained via the Herschel Science Archive. To access the HSA through HIPE, select Window-> Show Perspective -> Herschel Science Archive, and click on 'Open HSA User Interface' to launch the Archive browser. To access the HSA, a valid HSA login is required (see HowTo on accessing the Herschel Science Archive). Once you have successfully logged in, you should see the following:

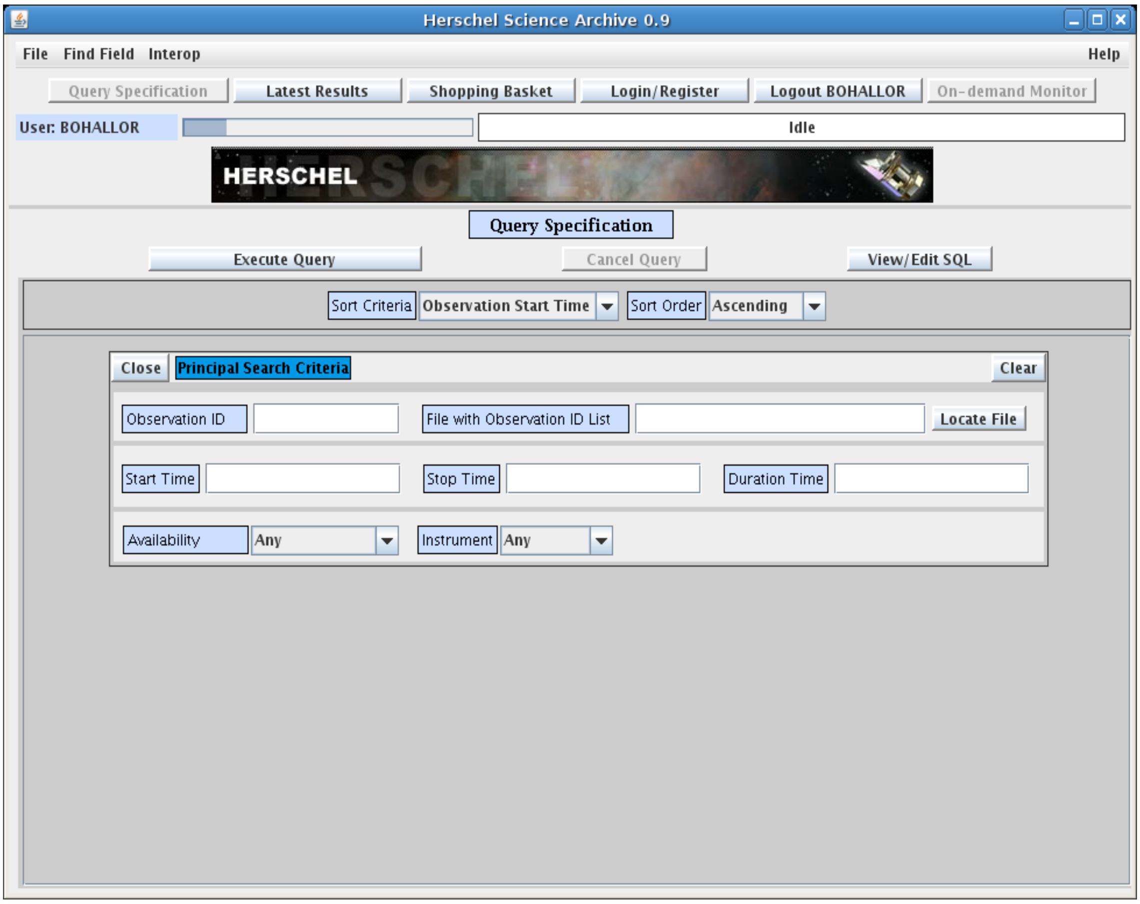


Figure 7.3. HSA Log-in

The HSA can be used to search via ObsID, Availability and Instrument. For the purposes of this example, we will obtain and process a POF5 observation (Scan Map Without Chopping).

Select the required observation (for our example case, ObsID: 3221226086),and move it to the Shopping Basket where you can submit a request to the HSA to download the relevant files for this observation. This can be retrieved via FTP, or more conveniently using the HSA Retrieval mechanism, which will bring the observation into your HIPE session (see HowTo on accessing the Herschel Science Archive).

In the latter case the default name of the downloaded observation product is

```
obs_<observation number>
# in our case, obs_3221226086
```

Using our access to the HSA -- we select the single observation

7.2.3. Setting up HIPE for processing data.

Once you have retrieved your data, you need to store the data in a pool and set up HIPE to access the data for processing. The simplest way of doing this is by setting up a pool,such as a LocalStore (in ".hcss/lstore" directory under user's login directory -- by default):

```
pool=LocalStoreFactory.getStore("spg_128") # Access to Products in a local
                                    # store called "spg_128"
storage=ProductStorage()  # tell the system it is a store of products
storage.register(pool) # register it
# "storage" is now one of the selectable data stores on the Data Access menu
storage.save("obs_3221226086")  # where "myobservation" should be replaced
                               # by the name of the downloaded observation
   # from the HSA
```

The id is the name given to the pool on creation. For the purposes of this HowTo, we will set up a pool with an ID of 'spg_128' that will be placed in our *${HOME}/.hcss/lstore* directory. Further information on how to manipulate pools and the other methods to store and access data via HIPE can be found in Section 12.2 of the Herschel User's Manual.

We can inspect the downloaded data and import it for processing in HIPE from our pool via the Data Access view of HIPE or through the Product Browser (as illustrated below). If you have not done so, you first need to register your pool for access -- otherwise, skip the first three lines and just open the Product Browser. Selections out of the browser will go to the variable "result" in the HIPE session.

pool=LocalStoreFactory.getStore("spg_128")

storage=ProductStorage()

storage.register(pool)

result = browseProduct(storage)

In this example, we are searching in terms of the Observation Context. Select the observation to be processed (in this case, 3221226090). The product window on the right provides us with a list of the files associated with the observation (attributes, meta data, auxillary and calibration files, level 0 products etc.)

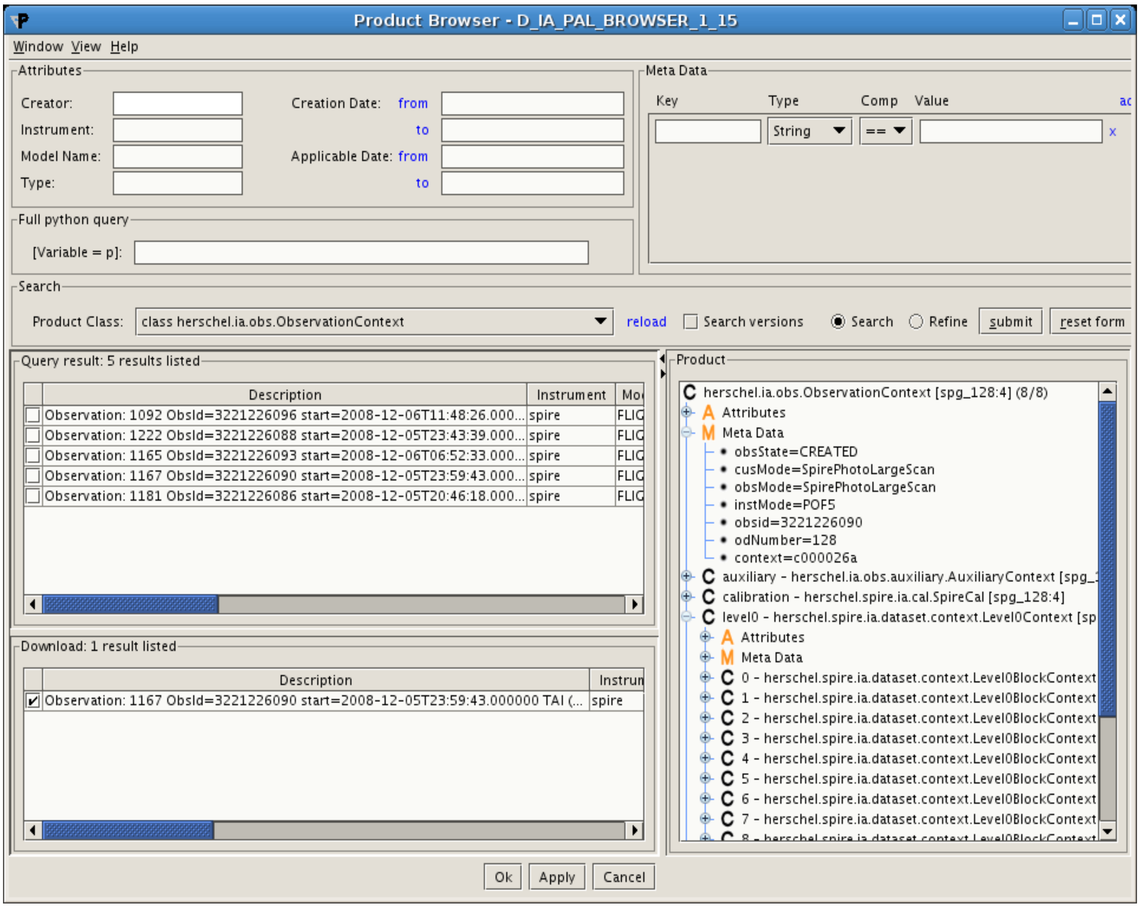


Figure 7.4. Selecting the Navigator window.

Once you have sucessfully downloaded your data and configured HIPE to access it via a pool, we can begin to pipeline process the data in HIPE. To start this, click on the 'Window' pull-down menu within HIPE, and select 'Show View' in order to access the 'Navigator' file/directory window:

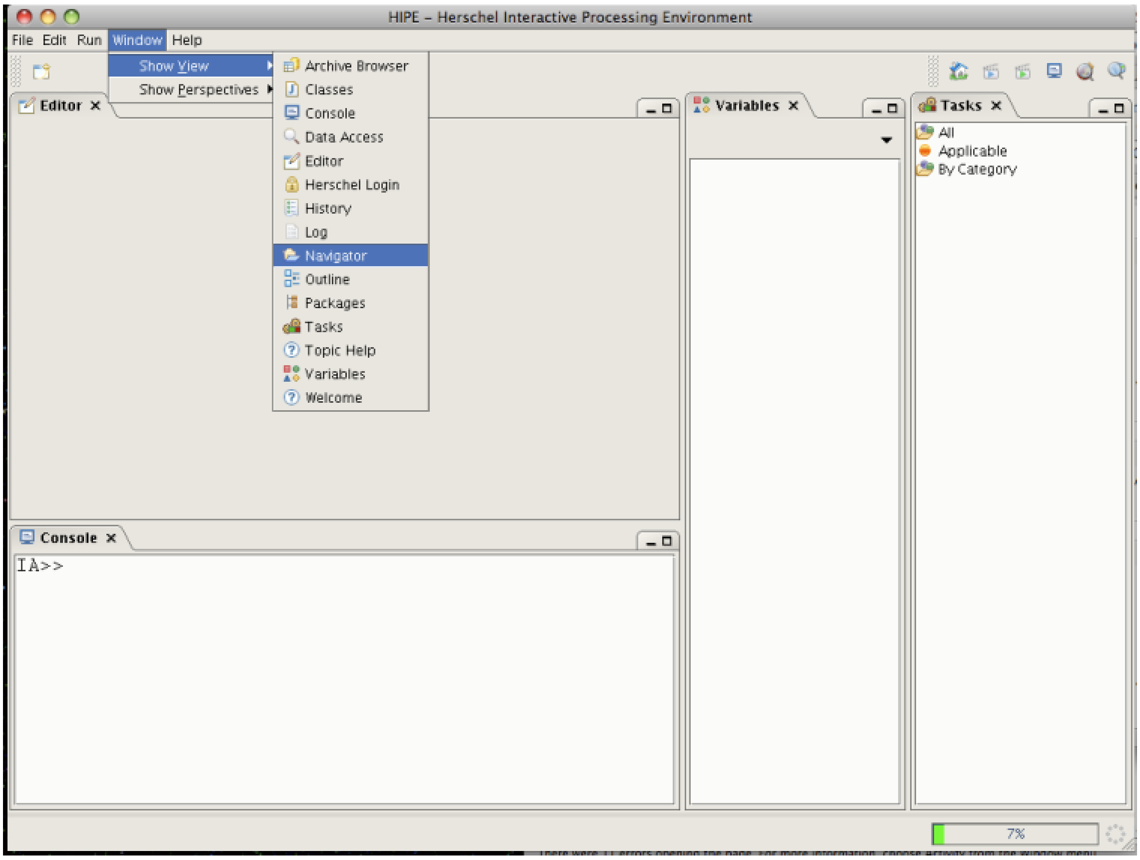


Figure 7.5. Selecting the Navigator window.

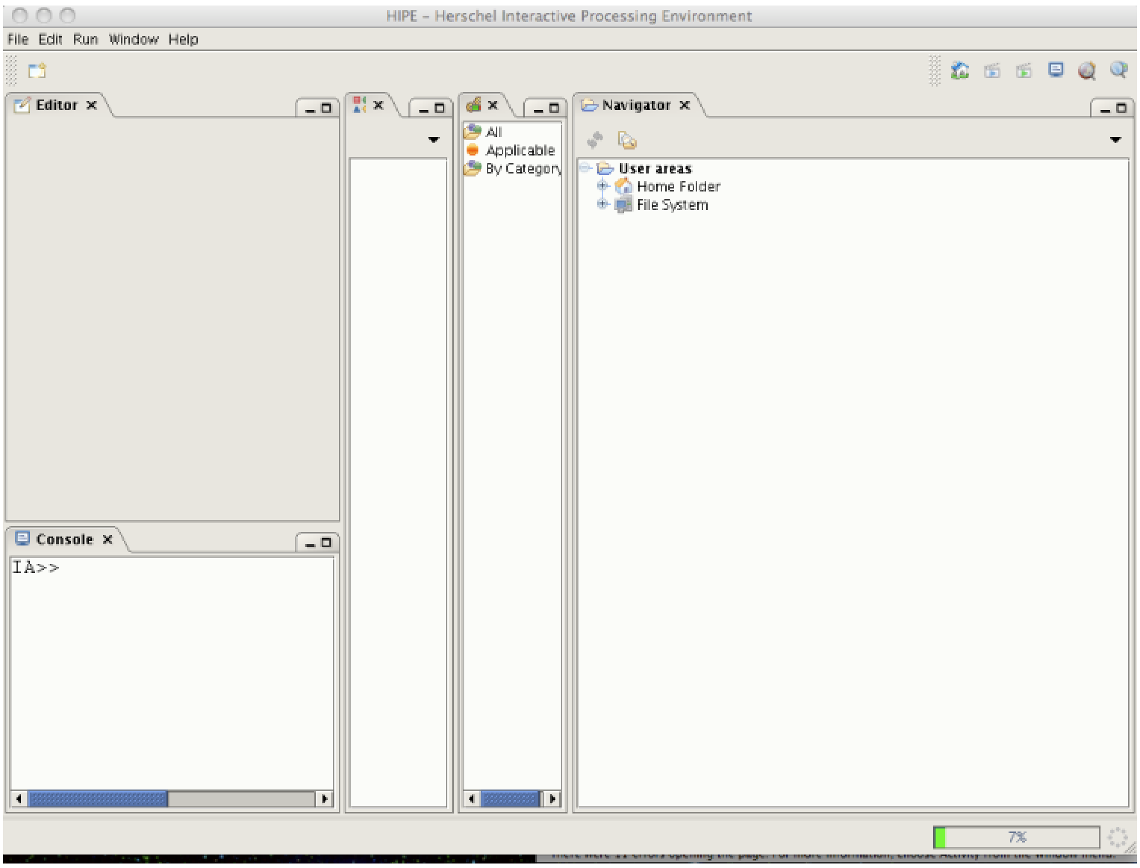


Figure 7.6. HIPE with Navigator window.

7.2.4. Running the pipeline.

Using the 'Navigator' window, navigate to the directory: ***/herschel/spire/ia/pipeline/scripts/*** within the build tree and choose the appropriate pipeline file (those with the `pipeline.py` extension) as shown in the figure below for the case of the POF5 pipeline script:

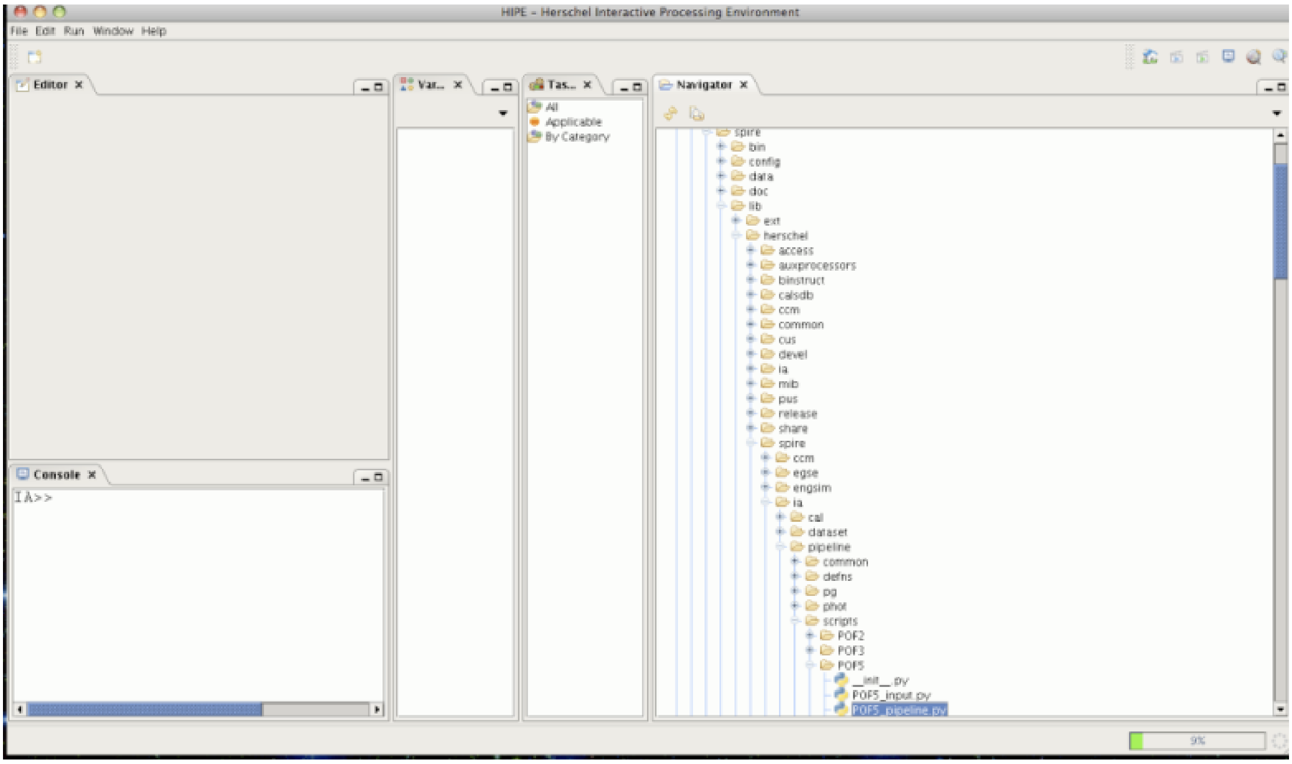


Figure 7.7. Selecting the pipeline script.

Double click on the name of the relevant pipeline script in the Navigator window, and the contents of the pipeline file will appear within the Editor window:

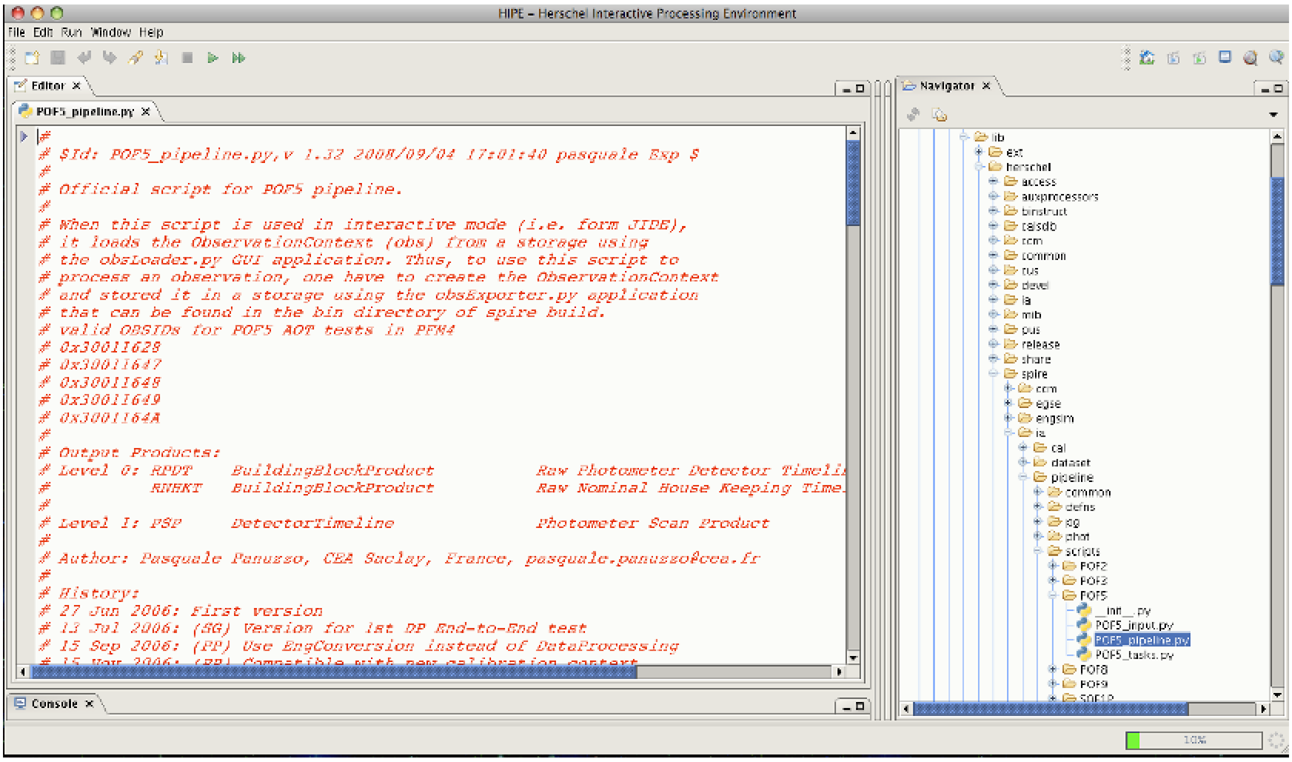


Figure 7.8. Loaded pipeline script, and contents loaded in Editor window.

There are two ways to run the pipeline though HIPE: 1) Either running through the pipeline script step-by-step by repeatedly clicking the Play button:

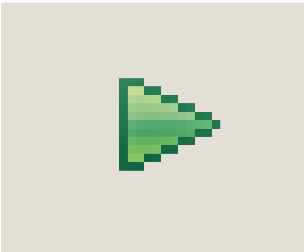


Figure 7.9. Play button.

2) or you can run the entire pipeline straight through by first highlighting the contents of the entire script within the Editor window and then clicking the Play button. Once you have started running the pipeline, you obtain the following window to track the progess of the processing:

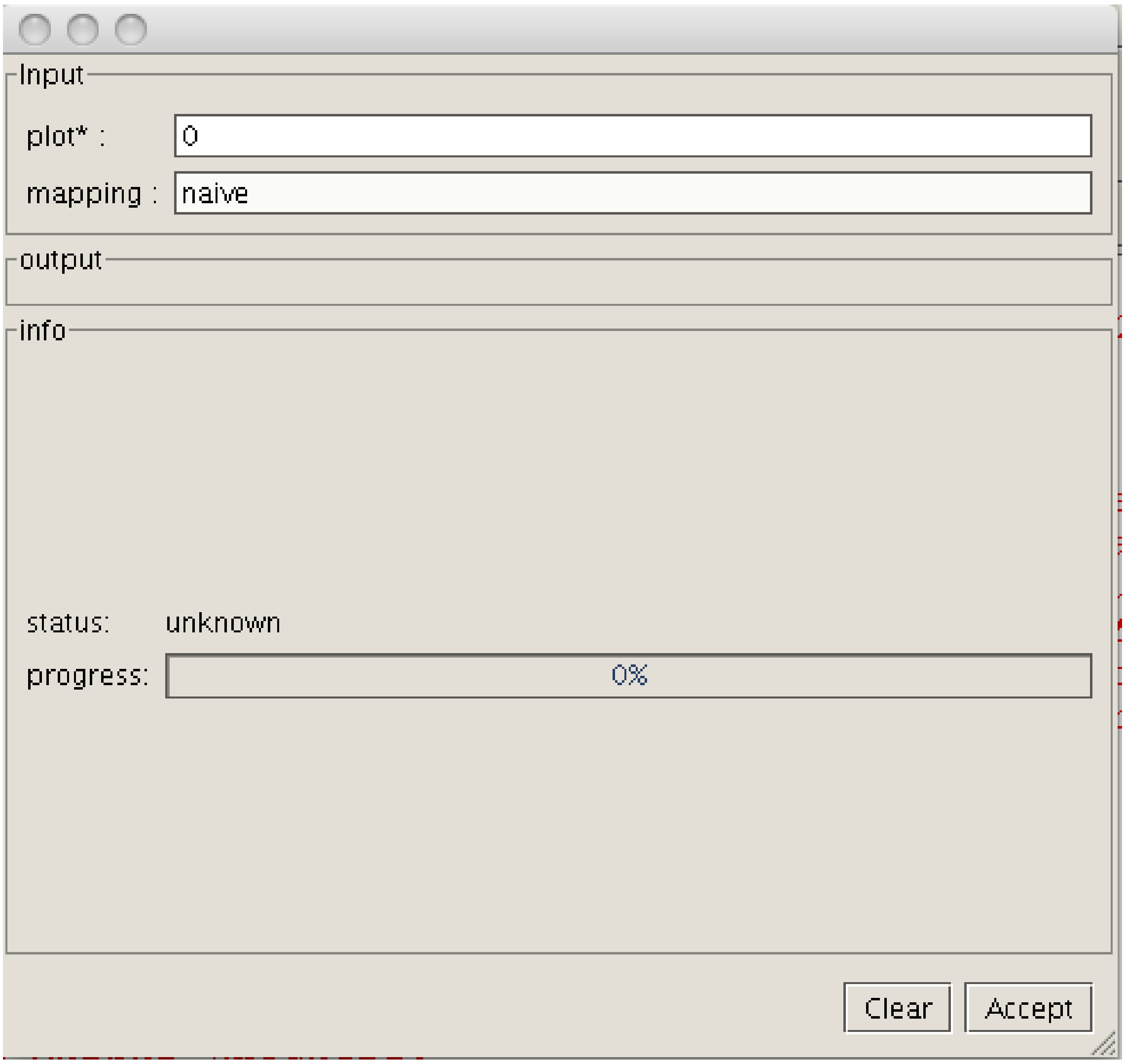


Figure 7.10. Plot dialog.

If you wish to obtain plots, change the 0 to 1 in the 'plot' dialog, otherwise just hit the 'Accept' button. This window needs to manually closed. Press the Stop button:

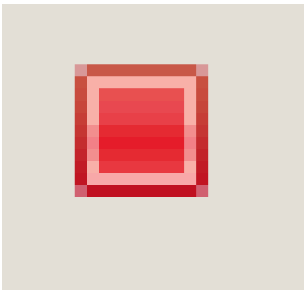


Figure 7.11. Stop button.

Next, you will see the `Observation_Loader` window - for HIPE to access and process the data via the pipeline, this requires a Storage ID of the data pool (in this case, spg_128) and an Observation ID (again in this case, 3221226090) for the data to be processed:

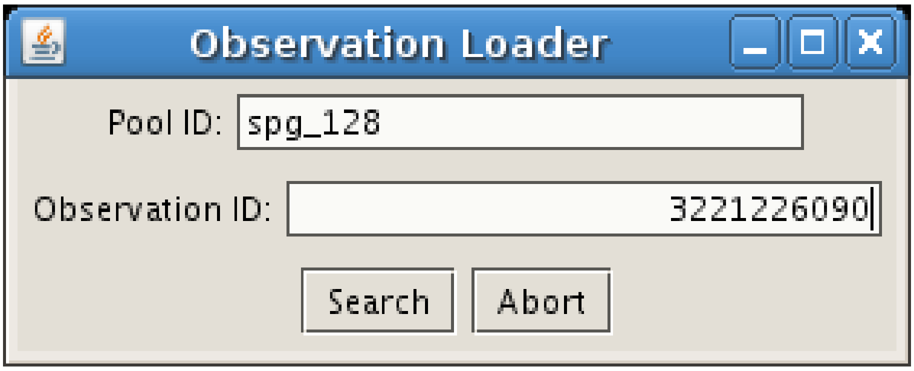


Figure 7.12. Observation Loader.

Then hit Search and the pipeline processing of data will start automatically. The pipeline will process data automatically from the initial product Level 0 to the final Level 2 products without further user interaction.

When the pipeline is finished running, a new dialog will appear on screen, asking you whether you wish to save the processed ObservationConext. Click yes to proceed.

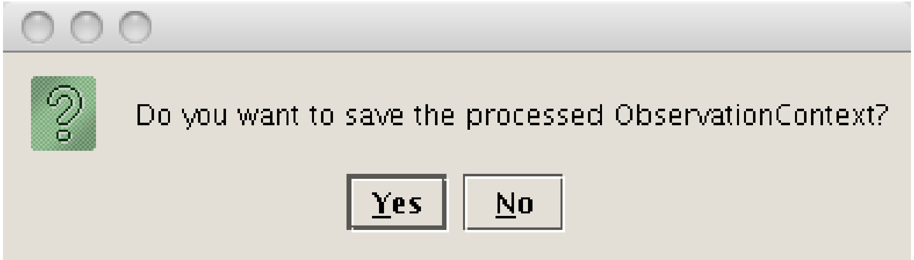


Figure 7.13. Observation context dialog.

Now enter the name of the pool where the user wants to save all the processed data in the dialog that pops up.

7.2.5. Accessing the processed data.

In order to browse the processed data, within the 'Variables' window, select 'obs' from the list of the available variables. Doing this will bring up the data summary information in the Editor window:

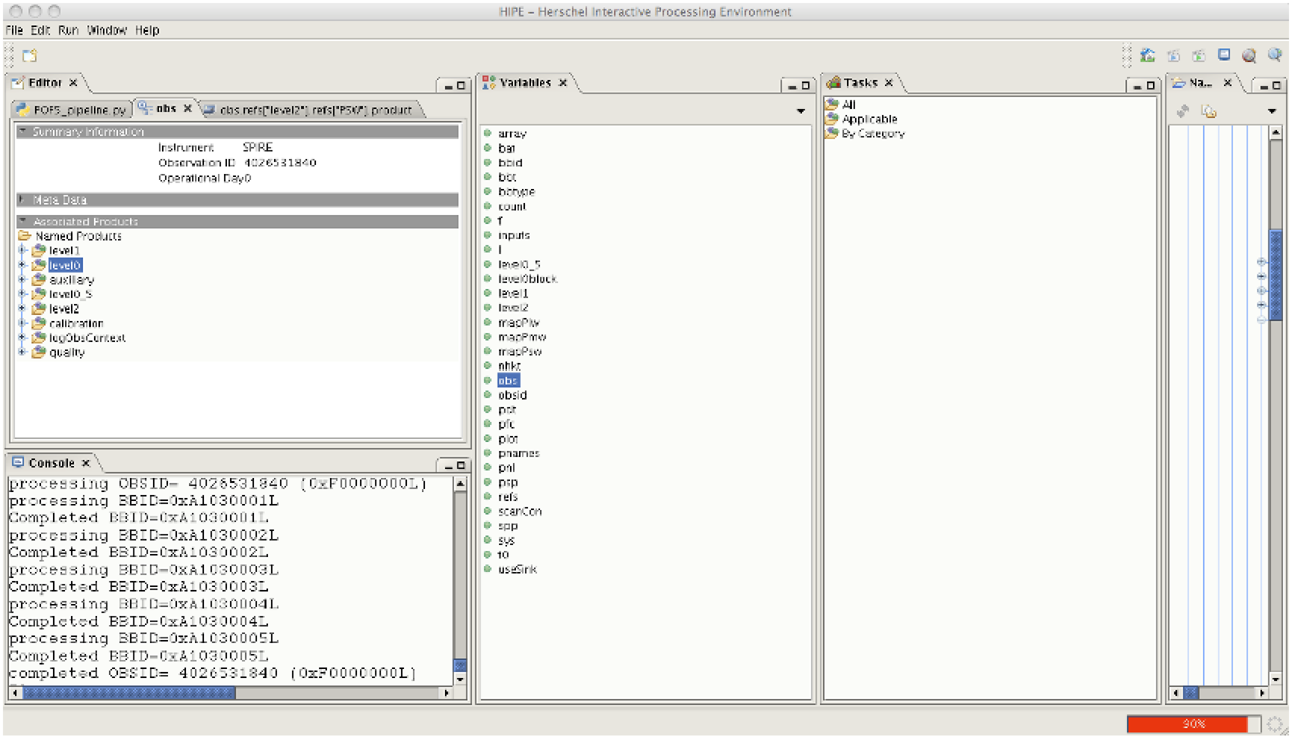


Figure 7.14. Data summary information.

Furthermore, different levels of data processing can be accessed and inspected from the associated products window. For an example of accessing a level of pipeline processing, we have accessed the level 2 product from the Map Making module :

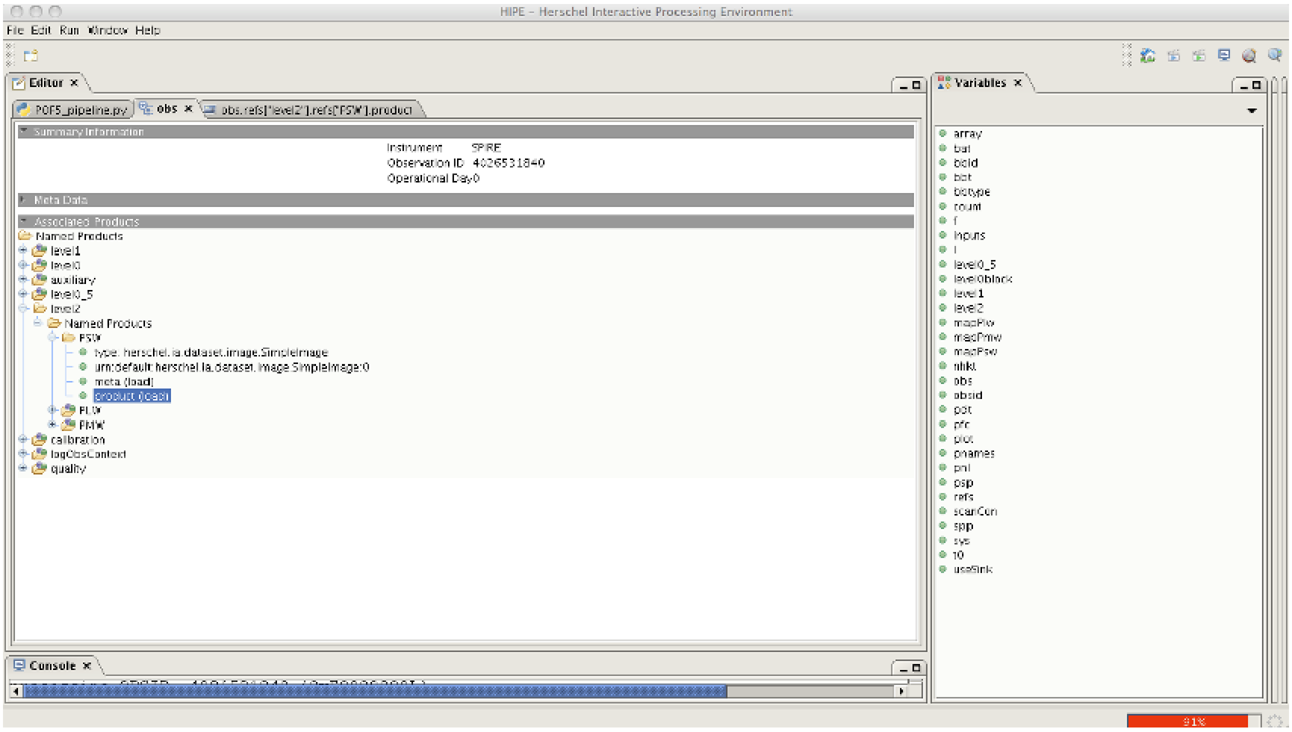


Figure 7.15. Accessing Level 2 product from Map Making module.

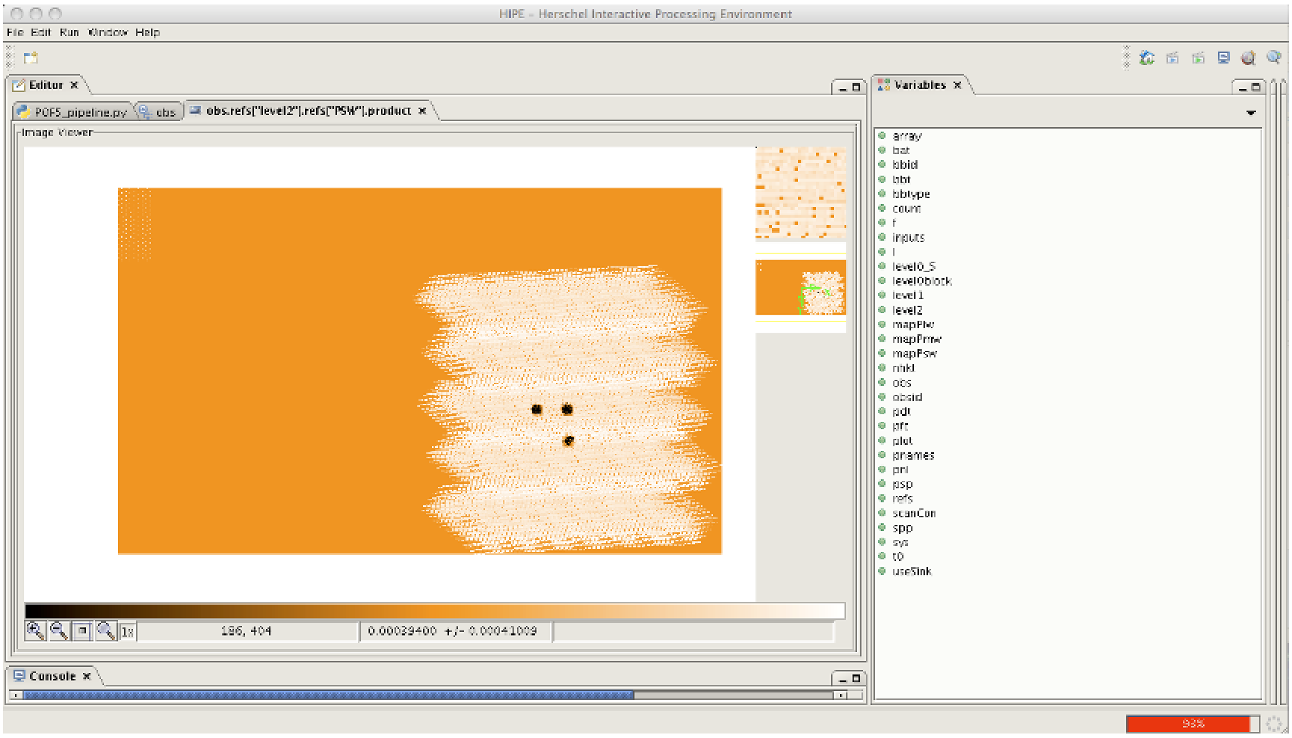


Figure 7.16. Level 2 product from Map Making module.

7.3. Additional reading

We have include processing of only the map making pipeline as an illustrative example - however, the processing of other pipelines will follow exactly the same prescription. Further, additional information regarding the structure of data at the various levels of processing, post-running of the respective pipeline scripts for each of the different pipelines can be found in the SPIRE Pipeline Description document.

Chapter 8. How to Save and Restore Data (ASCII and FITS)

Herschel Editorial Board

8.1. Introduction

Saving HCSS data and exporting it to a format which can be read by other tools outside HCSS is a very important task. It is necessary to have a correct understanding of the data structures in an object-oriented environment like the HCSS, because the data may come in different types of classes and hierarchies. The following drawing shows the general overview of the available high level data in HCSS:

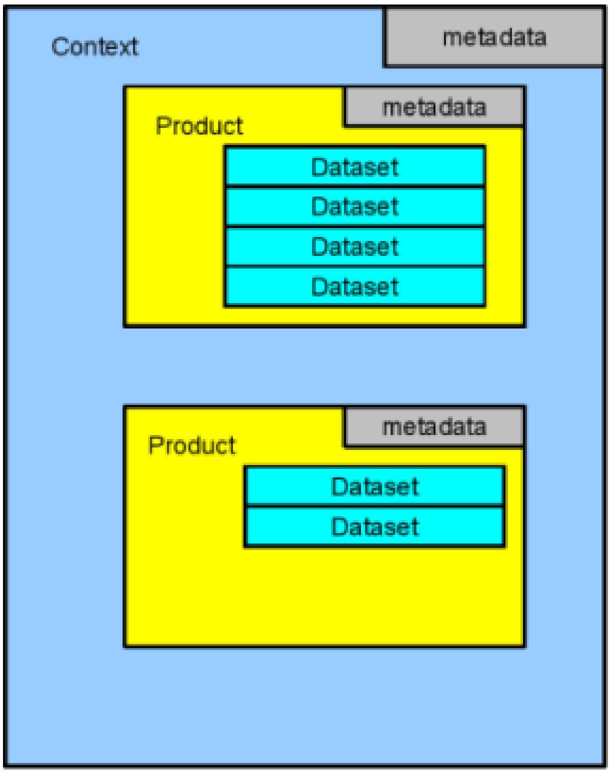


Figure 8.1. HCSS high level data hierarchy

In this scheme, the highest level data is the Context, which is a product that stores references to other products. So, the Context shown above in reality does not physically keep the two Products in it but instead it only keeps their references, also known as URNs.

The next level in the data hierarchy are Products, which may contain datasets of different types: ArrayDatasets, TableDatasets etc. Some examples of Products are the SimpleImage, which holds the images...

> **Note**
>
> How to save and restore variables of different kinds in a DP session is explained in Section 10.2 of the DP User's Manual.
>
> In short:

```
save("myfile.sav","x,y,z") # x, y, z are variables that are saved
 in a disk
                           # file called "myfile.sav"
save("myfile.sav") # saves all variables in a DP session on disk
 file
restore("myfile.sav")  # restores all variables to the session.
# WARNING: This file can get very large for many sets of data
 being accessed
# and saved from a DP session.
```

8.2. How to save and restore data from the command line

In order to illustrate the steps to save and restore data we need to create some HCSS high level data, like products and datasets.

First, let's create a product. An example of an HCSS product is the `SimpleImage` which is the product used to hold images, together with the optional mask, flag, exposure and errors images. It also includes necessary metadata in form of keywords (similar to FITS header keywords).

```
myImage = SimpleImage(description="An image",image = Double2d(100,100),\
        flag=Flag(100,100),error=Double2d(100,100),exposure=Double2d(100,100))
```

8.3. How to save and restore products using the local store

Note

All the information in this section is covered in much more detail in the PAL chapter of the "DP Basic User's Manual" available from within the Help environment of HIPE.

The easiest way to save and restore products in a HIPE session is to place them into a "local store". The "local store" is a simple folder structure which contains pools of products in the form of FITS files with corresponding metadata. Pool folders usually reside in the ".hcss/lstore" directory under the user's own home directory.

Saving data

In order to save a product into the local store we need to open a product storage, register a pool in it and then save the product into the pool. This is shown in the following example:

```
store = ProductStorage()
myPool = LocalStoreFactory.getStore("myTestPool")
store.register(myPool)
store.save(myImage)
```

Now the product "myImage" is saved in the local store in the directory on local disk called ${HOME}/.hcss/lstore/myTestPool.

Note that you can only save products, which means that if you want to save a Dataset of any kind - `TableDataset`, `Spectrum1d` or `2d` Dataset etc. you need to wrap them in a product as is shown in the following example

```
# create a TableDataset with two columns index and xvalue
table = TableDataset(description = "A table")
table["index"] = Column(data=Int1d.range(100))
```

```
table["xvalue"] = Column(data=Double1d(100).apply(RandomUniform()))
```

Next we need to put it in a product:

```
      tProduct = Product(description="A table")
tProduct["myTable"] = table
store.save(tProduct)
```

Placing things into products allows for the proper header information to be included. Products can be wrapped within products (e.g., several images in a single product such as an observation) and each level has its own metadata/header information.

Restoring data using the productBrowser

Restoring the data back in HCSS is more complicated as it is necessary to know the product URN in the local store in order to retrieve it. One way to do this is using the productBrowser(). We can also use the Data Access view (*see also HowTo on accessing data*):

```
store = ProductStorage()
myPool = LocalStoreFactory.getStore("myTestPool")
store.register(myPool)
result = productBrowser(store)
```

You can then query the available products in "myPool", select the one you need, add it to the basket and then exit the `productBrowser`. The steps are shown in the following screenshot:

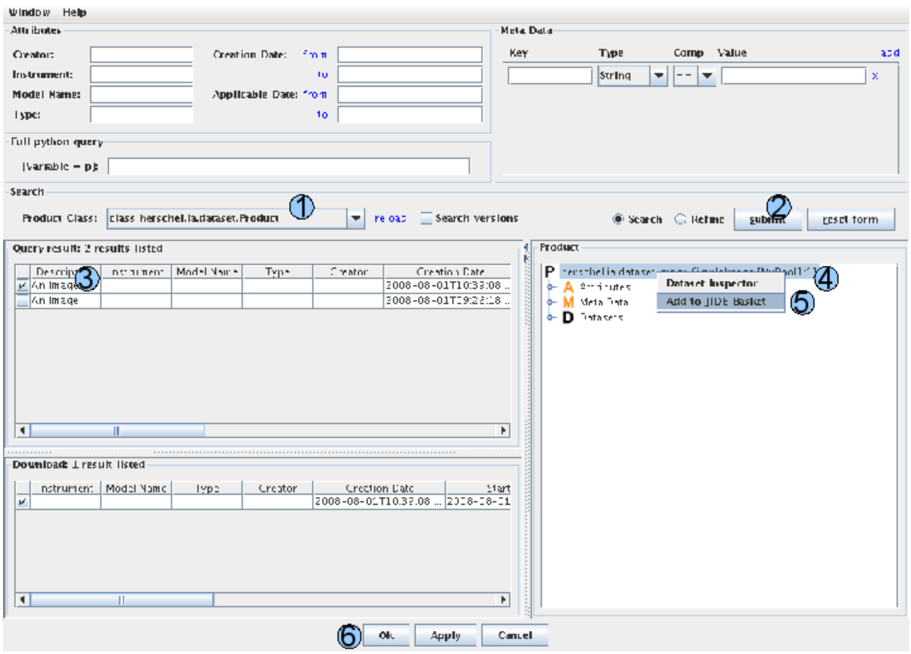


Figure 8.2. Restoring product from the local store using the `productBrowser`

These are the steps explained:

1. Select the product class to be of type Product
2. Click on "Submit" button to execute the search
3. Results for the products in myPool of type Product are shown in "Query result" view. Select the one you want.

4. The selected product structure appears in the "Product" view: Attributes, Metadata and Datasets are shown for this particular product.

5. Click with the right-hand mouse button on the product line (with the large "P" in front) opens a menu with "Dataset inspector" and "Add to JIDE Basket". Select the second item. The selected product will appear in the "Downloads" view.

6. Click "OK" to close the productBrowser().

At the end the reference to the product will be stored in the result variable and you can restore the SimpleImage following this example:

```
print result
# [urn:MyPool1:herschel.ia.dataset.image.SimpleImage:0]

image = result[0].product
```

If there is more than one result then we can refer to it with an index ([0] in the previous example).

The same way we can retrieve products which contain datasets (`TableDataset` or `ArrayDataset`) instead of `SimpleImage`.

Restoring data using command line queries

We can search the local store for products with a given attributes. For example, querying the local store pool "myPool" for products with description matching "An image":

```

query=MetaQuery(Product,"p","p.description=='An image'")
results2=store.select(query)
print results2
# [urn:MyPool1:herschel.ia.dataset.image.SimpleImage:0]

image = results2[0].product
```

The same as above, if there are more than one result then we can refer to it with the index.

8.4. How to Save Images and Tables as FITS files

It is possible to save and read using command-line input or task dialogs in HIPE. For all task dialogs it should be noted that the dialog appears in an Editor view window. To run the task via the dialog always hit the "Accept" button to bottom right in the dialog box.

8.4.1. Saving with a Task Dialog

The simplest way to save data is using the `simpleFitsWriter` task. It is required that the data (image/table/set of spectra) are wrapped up as a product. An example product is an observation itself. But we can wrap any dataset into a product so that the appropriate metadata (header information) and history is available. Clicking on the name of any product will show this task available in the Tasks under the Applicable Tasks folder. A double-click on the task (shown in green) brings up the simple dialog shown in Figure 8.3.

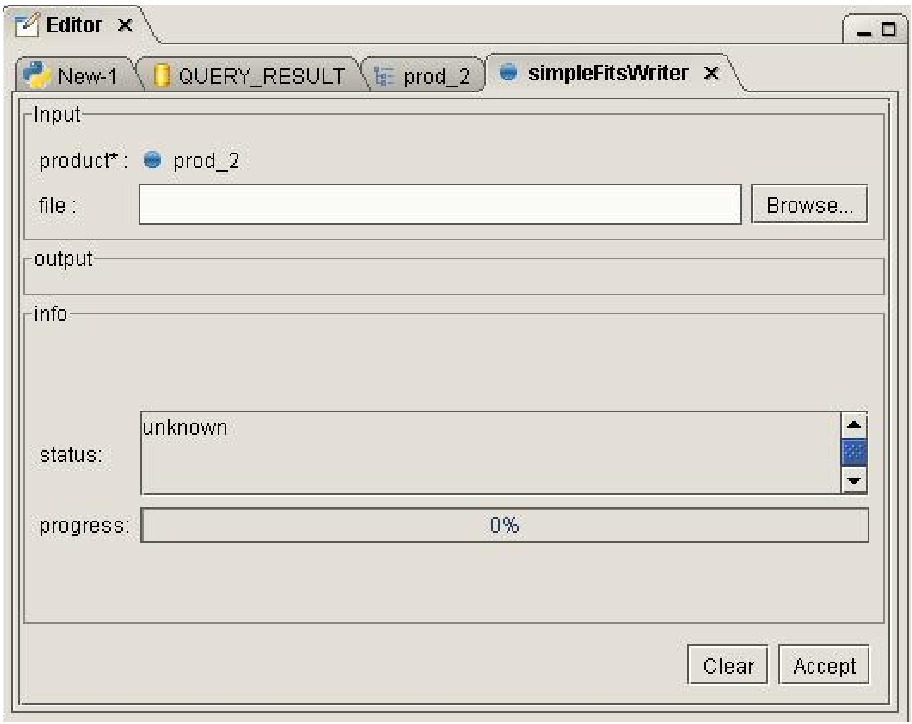


Figure 8.3. FITS save task dialog.

The only option the user needs to fill in is the name of the output FITS file. The default directory is the one that hipe was started from, so the full path name is usually required. Hitting the "Accept" button runs the task.

8.4.2. Saving Using Command-line Inputs

The FITS reader and writer in HCSS is in the FitsArchive package. First, let's store the product "myImage" from our previous example in a FITS file.

```
fits=FitsArchive()
fits.save("testFits-file1.fits",myImage)
```

The file "testFits-file1.fits" will be saved in the folder from where you started up HIPE. Otherwise, the full directory path should be supplied. It is a multi-extension FITS file with all the content of the SimpleImage product. Here is the structure of the saved FITS file:

```
 No. Type     EXTNAME      BITPIX Dimensions(columns)

   0  PRIMARY                 32     0
   1  IMAGE    image         -64     100 100
   2  IMAGE    flag           16     100 100
   3  IMAGE    error         -64     100 100
   4  IMAGE    exposure      -64     100 100
```

8.4.3. How to Save TableDatasets as FITS Files

Once we have the TableDataset wrapped in a Product we can save it like all other products. We can use the same FITS writing task from HIPE as noted above, or we can use a command-line method. For example:

```
      fits=FitsArchive()
      myTable = TableDataset() # create an empty table
      myTable["X values"] = Column(Double1d([2,3.4,4])) # create fake column
```

```
myTable["Y values"] = Column(Double1d([2,4.5,4.8])) # create 2nd column
tProduct = Product(description="This is a table") # create the product
tProduct["firstTable"] = myTable  # add in the table and give it a label
fits.save("testFits-file2.fits",tProduct)
```

The resulting structure of the saved FITS file is:

```
   No. Type    EXTNAME      BITPIX   Dimensions(columns)

   0  PRIMARY                 32     0
   1  BINTABLE table           8     2(3)

 Column Name                       Format     Dims       Units     TLMIN  TLMAX
      1 X values                   1D
      2 Y values                   1D
```

We can see that the column names, which we named as "X values" and "Y values" are in the file.

8.4.4. How to Read FITS Files

The `simpleFitsReader` task allows FITS files to be read in. Two types of FITS readers are available -- for HCSS FITS and Standard FITS. You can let the software choose the appropriate one or choose a specific reader (see Figure 8.4).

To run the command from the HIPE dialog, go to the "Tasks" view -- select the "All" tasks folder and scroll down to `simpleFitsReader`. A double-click on the name brings up the dialog. Once a name is input and the FITS reader chosen, click the "Accept" button to run the task and read in the FITS file.

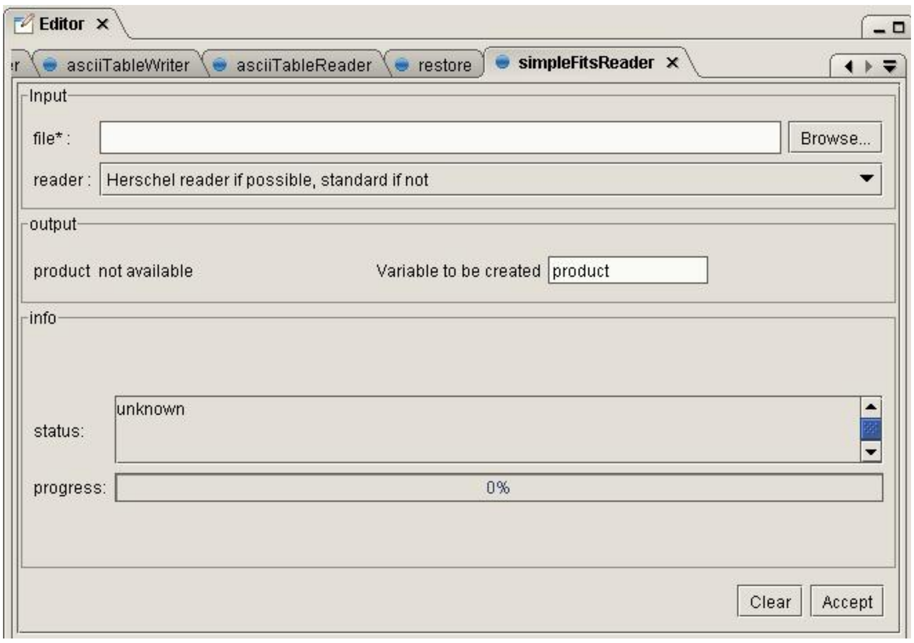


Figure 8.4. FITS read task dialog.

8.5. How to Create and Read ASCII Table Files

In this case it is not necessary to put the TableDataset in a Product and we can directly save the Dataset to an ASCII file. As for FITS writing, we can do this from within a HIPE task dialog or from the command-line.

8.5.1. Using HIPE Task Dialogs to Create and Read ASCII Tables

If we click on a variable that is a TableDataset -- such as "myTable" in the example above -- in the Variables view of HIPE, then we see that an Applicable Task in the Tasks view window is `asciiTableWriter`. Double-clicking on this task brings up a dialog for creating an ASCII table. The simplest way of formulating an ASCII table is to take the defaults and simply fill in a name for the output table. But more sophisticated options are available (see Figure 8.5).

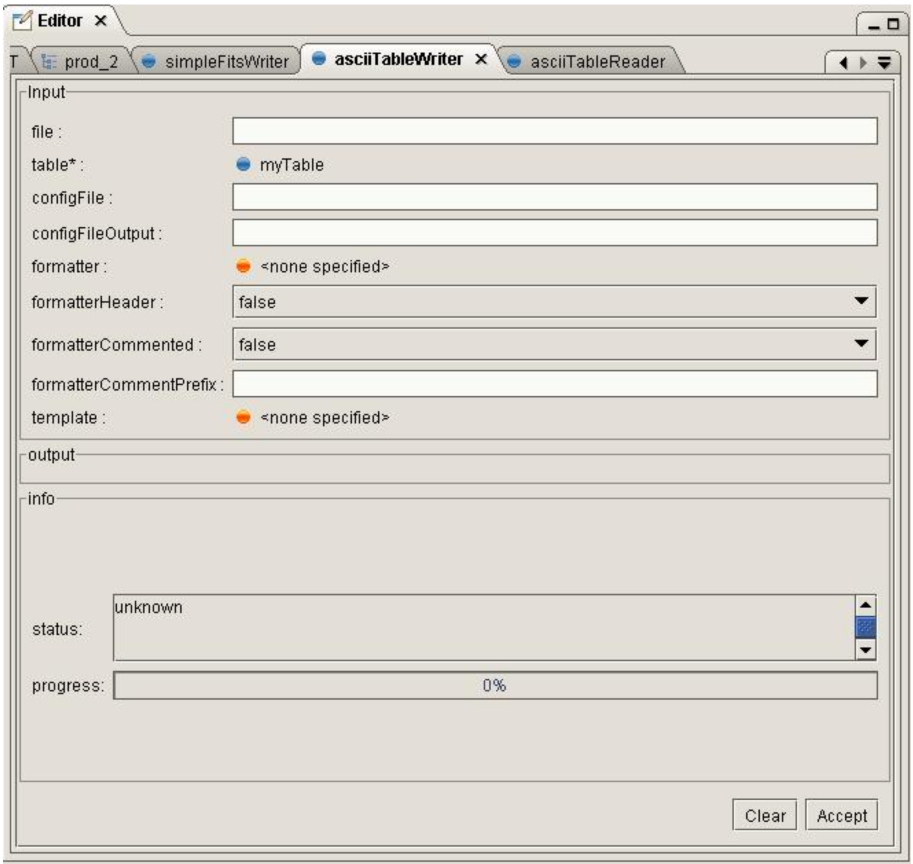


Figure 8.5. FITS save task dialog.

The other possible inputs for the task are the following (this information is also available by hovering the mouse over the parameters shown in the dialog).

```
* file = output file name.
* table = TableDataset to write.
* configFile  = configuration file where the formatter
(AsciiFormatter), parser (AsciiParser) and table template
(TableTemplate) must be specified. When configFile parameter is specified,
any parameter related to parser or to table template are not allowed.
* configFileOutput = if a config file is specified, an output configuration
file will be created.
* formatter (default AsciiTableTool formatter) = AsciiFormatter object.
* formatterHeader (default AsciiFormatter header allowed) = Specifies
if header information to be provided (true/false).
* formatterCommented (default AsciiFormatter comments allowed) = Specifies
if there are comments when writing a file (true/false).
* formatterCommentPrefix (default AsciiFormatter comments prefix value) =
Specifies what the prefix is for identifying all comments.
* template (INPUT, default value: extracted from the first file rows) =
TableTemplate object for specifying the data structure (see DP Basic User's
```

```
    Manual for more details).
```

Clicking on "Accept" at the bottom of the task dialog window runs the task and creates an ASCII table.

Reading an ASCII table into HIPE can be done using the `asciiTableReader`. Go to the "Tasks" view and open the folder "All". Double-click on the word `asciiTableReader`. This provides a dialog. For standard CVS tables the only thing that needs to be filled in is the file name of the ASCII table to be read in. More

```
    * file = input file containing ASCII table.
    * table = TableDataset object name for loaded table.
    * configFile = configuration file where the formatter (AsciiFormatter),
    parser (AsciiParser) and table template (TableTemplate) must be specified.
    When configFile parameter is specified, any parameter related to parser or
    to table template are not allowed.
    * configFileOutput = if a file is specified, an output configuration
    file will be created.
    * parser (default AsciiTableTool parser) = AsciiParser object.
    * parserIgnore (default AsciiParser ignore value)
    = String expression to ignore when parsing a file.
    * parserSkip (default AsciiParser skipping rows value) = Number of rows to
    skip when reading a file.
    * parserTrim (default AsciiParser trim rows value) = Specifies if the parser
    must trim each row when reading a file (true/false).
    * parserGuess (default value AsciiParser.GUESS_NONE) = specifies if
    the parser should guess column types. Files should not contain HCSS header
    (use skip=AsciiReader.HCSS_HEADER for skipping HCSS header or comment these
    lines)

      Valid options:
          o AsciiParser.GUESS_NONE: (default) file must contain template
   or template must be provided (no guess)
          o AsciiParser.GUESS_TRY: guess types based on the first 100 records
          o AsciiParser.GUESS_ALL: guess types based on all records
          o AsciiParser.ALL_STRING: each record is a string (no guess required)
          o AsciiParser.ALL_BOOLEAN: each record is a boolean (no guess required)
          o AsciiParser.ALL_BYTE: each record is a byte (no guess required)
          o AsciiParser.ALL_INTEGER: each record is an integer (no guess required)
          o AsciiParser.ALL_LONG: each record is a long (no guess required)
          o AsciiParser.ALL_FLOAT: each record is a float (no guess required)
          o AsciiParser.ALL_DOUBLE: each record is a double (no guess required)
          o AsciiParser.ALL_COMPLEX: each record is a complex (no guess required)
    * parserDelim (INPUT, default value: comma) = Specifies the field delimiter.
    If it is one character, a csvParser is selected. If it is an expression,
    a RegExpParser (regular expression) is selected.
    * template (INPUT, default value: extracted from the first file rows) =
    TableTemplate object for specifying the data structure. See TableTemplate.
```

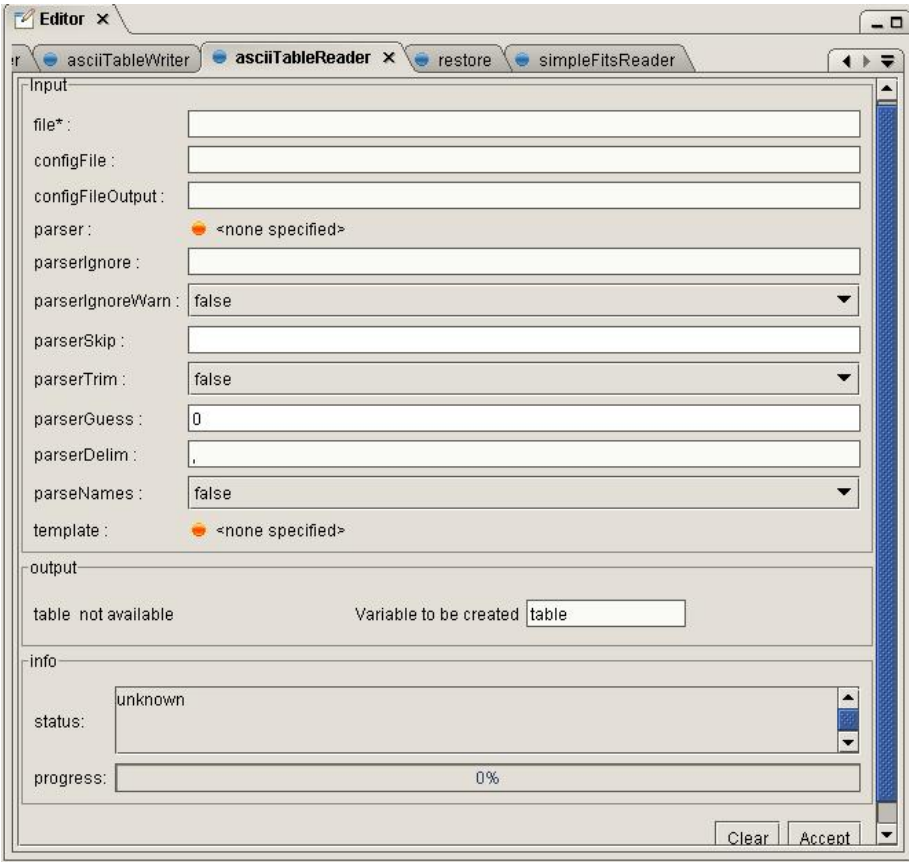


Figure 8.6. FITS read task dialog.

8.5.2. Using Command-line Input to Create and Read ASCII Tables

We can also do simple ASCII table creation from command-line inputs.

```
ascii = AsciiTableTool()
ascii.save("testAscii-file1.txt",table)
```

To read the table in again we need to "load" it using the same tool.

```
 ascii = AsciiTableTool()
table=ascii.load("testAscii-file1.txt")
```

Loading uses the same parser/formatter (see below for how this may be changed) as is applied for saving.

By default the table is saved as a coma-separated-value file with 4 header lines, for example

```
X values,Y values       # column names
Double,Double           # column data types
,                       # column data units
,                       # description of the column
2.0,2.0  # the data start from this line
3.4,4.5
4.0,4.8
```

The default output delimiter can be changed to another symbol, like is shown in the following example:

```
ascii.formatter = CsvFormatter(delimiter = '*')
ascii.save("testAscii-file2.txt",table)
```

Or the columns at output can have a fixed width using the FixedWidthFormatter with an indication of the column widths given.

```
ascii.formatter = FixedWidthFormatter(sizes=[8,12])
ascii.save("testAscii-file3.txt",table)
```

More information on creating and reading tables is available in the DP Basic User's Manual.

Chapter 9. How to plot in HIPE

Herschel Editorial Board

Revision History		
Revision 0.1	23 June 2008	IV
Created using RS template.		
Revision 0.2	16 Oct 2008	IV
Some corrections and modifications.		

9.1. Introduction

Plotting in HCSS is object oriented - each element of the plot window is an object and the user can interact with it using its methods. For example, the main plot objects (or class) is called PlotXY() and we have the axes and the different plotting layers as distinct objects and we can change their properties, like the number of tick marks, the colour of the plotting symbols, adding new layers etc, without the necessity to redraw the whole chart. This is powerful but seemingly complicated and this How To is targeted to make it a bit more accessible and provide you with a simple receipts on how to do simple plots.

An extensive introduction to plotting in HCSS-DP is given in the DP User's Manual and here we very briefly introduce the basic plotting from the command line and using the plot properties GUI.

9.2. Simple plots from the command line

In order to illustrate the steps to produce simple plots we need an input x and y variables:

```
x = Double1d.range(11)
y = x*x
```

1. Simple plot:

   ```
   from herschel.ia.gui.plot import *

   plot = PlotXY()
   plot.autoBoxAxes=1
   layer = LayerXY(x,y)
   plot.addLayer(layer)
   ```

2. Overplot a second x and y dataset

   ```
   x1 = 10.0*Double1d.range(11)/10.0 - 5.0
   y1 = x1**3.0
   ```

   Note that we do not need to repeat all plotting commands from the above example, we simply add a new layer

   ```
   layer2 = Layer(x1,y1)
   plot.addLayer(layer2)
   ```

   And we note that the axis ranges are expanded correspondingly and that the new layer is with a different colour.

3. Change the plot title and subtitle

```
plot.title.text="Example plot"
plot.subtitle.text="two layers"
```

or if you don't want to have plot title and subtitle you can switch them off

```
plot.title.setVisible(0)
plot.subtitle.setVisible(0)
```

4. Change the axis labels:

```
plot.xaxis.title.text="X-values"
plot.yaxis.title.text="Y-values"
```

5. Change the axis ranges

```
plot.xaxis.setRange([-2.0,2.0])
plot.yaxis.setRange([-10.0,10.0])
```

or go back to the auto range

```
plot.xaxis.setAutorage(1)
plot.yaxis.setAutorage(1)
```

6. Change the tick marks spacing and then the number of minor tick marks

```
plot.xaxis.getTick().setInterval(3.0)
plot.yaxis.getTick().setInterval(30.0)
```

and to have 5 minor tick intervals between the major tick marks (which means 4 minor ticks)

```
plot.xaxis.getTick().setMinorNumber(4)
plot.yaxis.getTick().setMinorNumber(4)
```

7. Draw grid lines

```
plot.xaxis.getTick().setGridLines(1)
plot.yaxis.getTick().setGridLines(1)
```

Note that the grid lines are drawn at the major tick marks.

8. Change the axis from linear to log

```
plot.xaxis.setType(Axis.LOG)
plot.xaxis.setType(Axis.LINEAR)
```

Warning

The axis ranges need to be positive otherwise you will get "IllegalArgumentException" error.

9. Change the line style for a given layer

```
layer.setLine(Style.NONE)
```

The line styles for setLine() can be

- Style.NONE - symbols only
- Style.MARKED - symbols connected with lines
- Style.SOLID - solid line, no symbols
- Style.DASHED - dashed lines

- Style.MARK_DASHED - symbols connected with dashed lines

  Note that in the MARKED styles the default plotting symbol is used

10. Change the plotting symbol and its size. In order to have an effect you need to change the line style first to be one of NONE or MARKED styles

```
layer.setLine(Style.NONE)
layer.setSymbol(Style.FSQUARE)
layer.setSymbolSize(10)
```

The symbols can be:

Table 9.1. Symbols codes

DOT = 1	a dot	VCROSS = 2	a "+" sign
DCROSS = 3	an "x" sign	VDCROSS = 4	a "+" + "x" sign
CIRCLE = 5	an empty circle	TRIANGLE = 6	an empty triangle
UTRIANGLE = 7	an empty upside-down triangle	SQUARE = 8	an empty square
SQUARE_CROSS=9	an empty square + "x"	DIAMOND = 10	an empty diamond
DIAMOND_CROSS=11	a diamond + "+"	OCTAGON=12	an empty octagon
STAR = 13	an empty star	FCIRCLE=14	a filled circle
FTRIANGLE=15	a filled triangle	FSQUARE = 16	a filled square
FDIAMOND=17	a filled diamond	FOCTAGON=18	a filled octagon
UARROW = 19	an up arrow	DARROW = 20	a down arrow
RARROW=21	a right arrow	LARROW = 22	a left arrow
DARROW_LARGE=23	a large down arrow	UARROW_TRIANGLE = 24	a large up triangular arrow
DARROW_TRIANGLE = 25	a large down triangular arrow		

Note

You can use either the code or the numeric value for the symbol, that is, setSymbol(Style.FSQUARE) is equivalent to setSymbol(16).

11. Change the colour of the symbols and lines for a given layer

```
layer.setColor(java.awt.Color.RED)
```

12. Show or remove the legend for the layers

```
plot.setLegendVisible(1)
```

and we can also remove itt

```
plot.setLegendVisible(0)
```

13. We can also change the legend name for a given layer

```
layer.setName("Test 1")
```

and we can also remove the legend for a particular layer if we don't want it to appear on the plot

```
layer.setInLegend(0)
```

14. Histogram mode. You need to be in MARKED or SOLID line style for this mode to work:

    ```
    layer.setLine(Style.MARKED)
    layer.style.setChartType(Style.HISTOGRAM)
    ```

    The chart type can be HISTOGRAM - the data point is in the middle of the histogram horizontal bar, HISTOGRAM_EDGE - the data point is on the edge of the histogram horizontal, LINECHART - the data points are connected with lines.

15. Add error bars to x and/or y values. First we need to create arrays with errors

    ```
    xerr = SQRT(x)
    yerr = SQRT(y)

    layer.setErrorX(xerr,xerr)
    layer.setErrorY(yerr,yerr)
    ```

    Note that the upper (the first argument to setError() method) and the lower (the second argument to setError() method) error limits can be different.

16. Add an annotation

    ```
    layer.setAnnotation(0,Annotation(6.5,-10,"Test",color=java.awt.Color.GREEN))
    ```

17. Use math and special symbols for axis titles, annotations, title plot etc. It is possible to use TeX like formating of strings, for example the following can be used to set the title of the x axis

    ```
    plot.xaxis.title.text="$A_{1.3}^{b-3/2}$"
    plot.xaxis.title.text="$\\alpha_{1.3}^{\\beta-3/2}$"
    ```

    Note that it is necessary to use "\\" to escape the "\" symbol.

    Warning

    Not all special symbols (mainly Greek characters) are available. If the symbol is not available (for example $\\Alpha$ is not available) then this produces an IllegalArgumentException error.

18. Change the plot window size. You can resize the window with the mouse or you can specify the desired window size once you have added layers to the plot

    ```
    plot.setWidth(400)
    plot.setHeight(300)
    ```

19. Save plot in a file

    ```
    plot.saveAsJPG("myfile.jpg") # JPEG format
    plot.saveAsEPS("myfile.eps") # Encapsulated PS
    plot.saveAsPNG("myfile.png") # PNG format
    ```

9.3. Interacting with plots using plot properties GUI

Once you have done steps 1., 2. and/or 3. from the above then most of the following interactions with the plot properties, i.e. steps from 4. on from the previous section, can be done via the plot properties GUI. To open up the plot properties GUI you need to click with the right-hand mouse button and choose "Properties..." entry in the menu. This will bring up the following window:

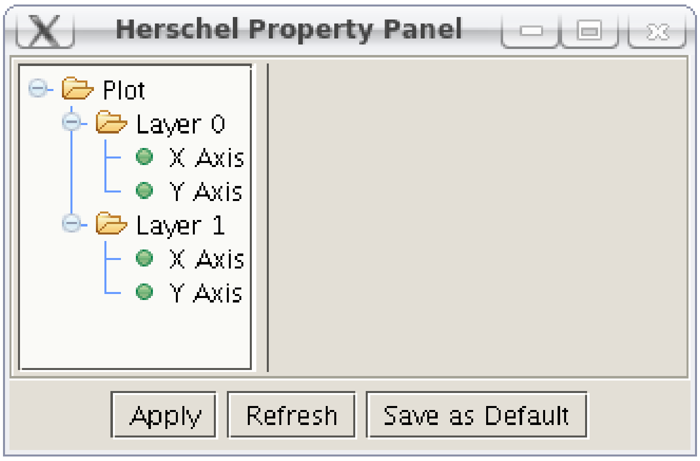


Figure 9.1. Plot properties initial windows.

On the left frame of the window we can see the hierarchical structure of the plot we created following steps 1., 2. and 3. from the above example: we have the root element our plot and we have two layers added on top of it. Each layer has also sub-objects for the x and y axes. Navigating to each of these elements we can change the properties for each of the items.

1. First, let's enter the Plot properties:

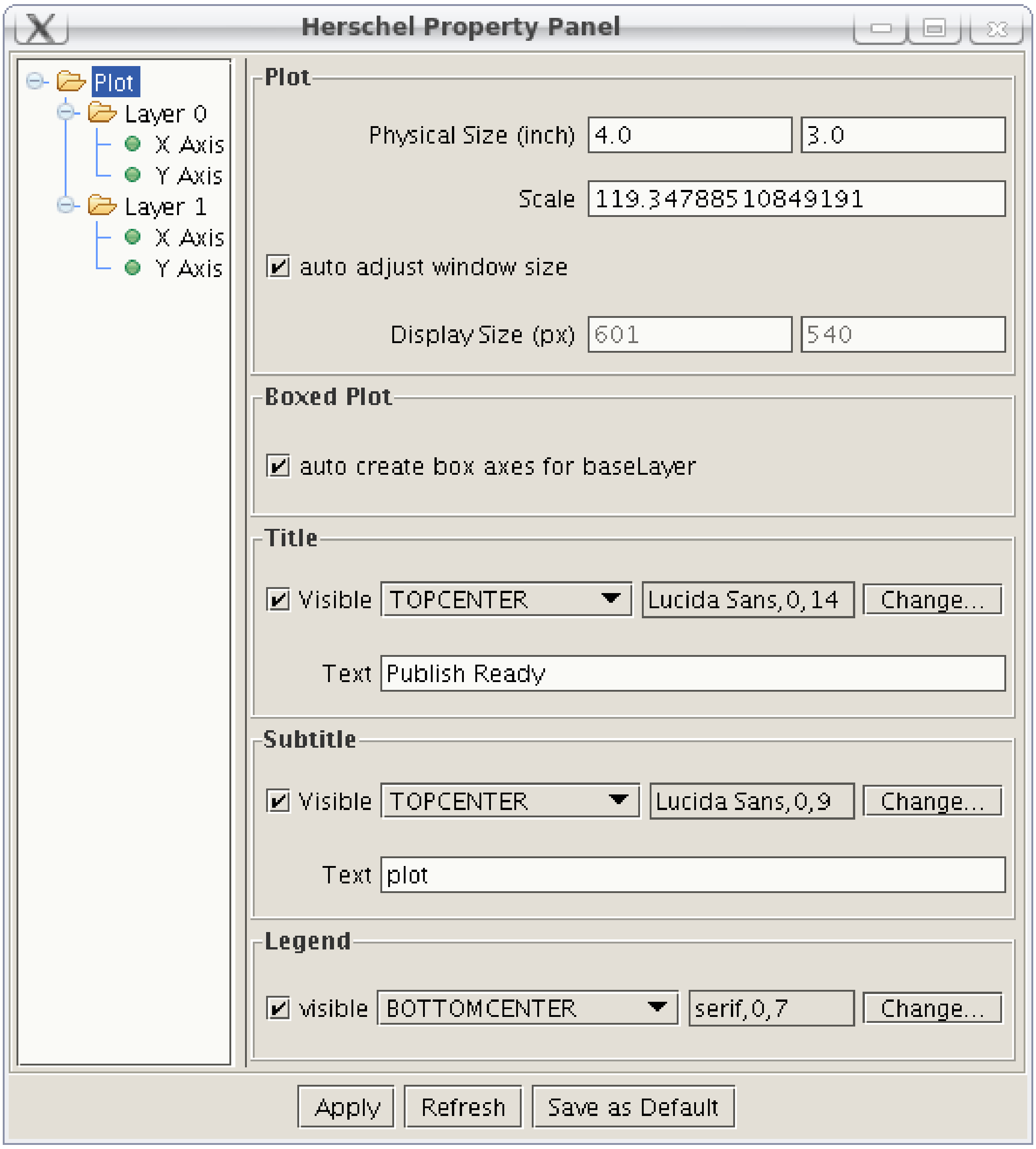


Figure 9.2. Plot properties items

The different base plot entries are self-explanatory and they can be changed interactively and applied. Also, if you plan to reuse some of them for all subsequent plots you may save them as default.

2. Layers properties

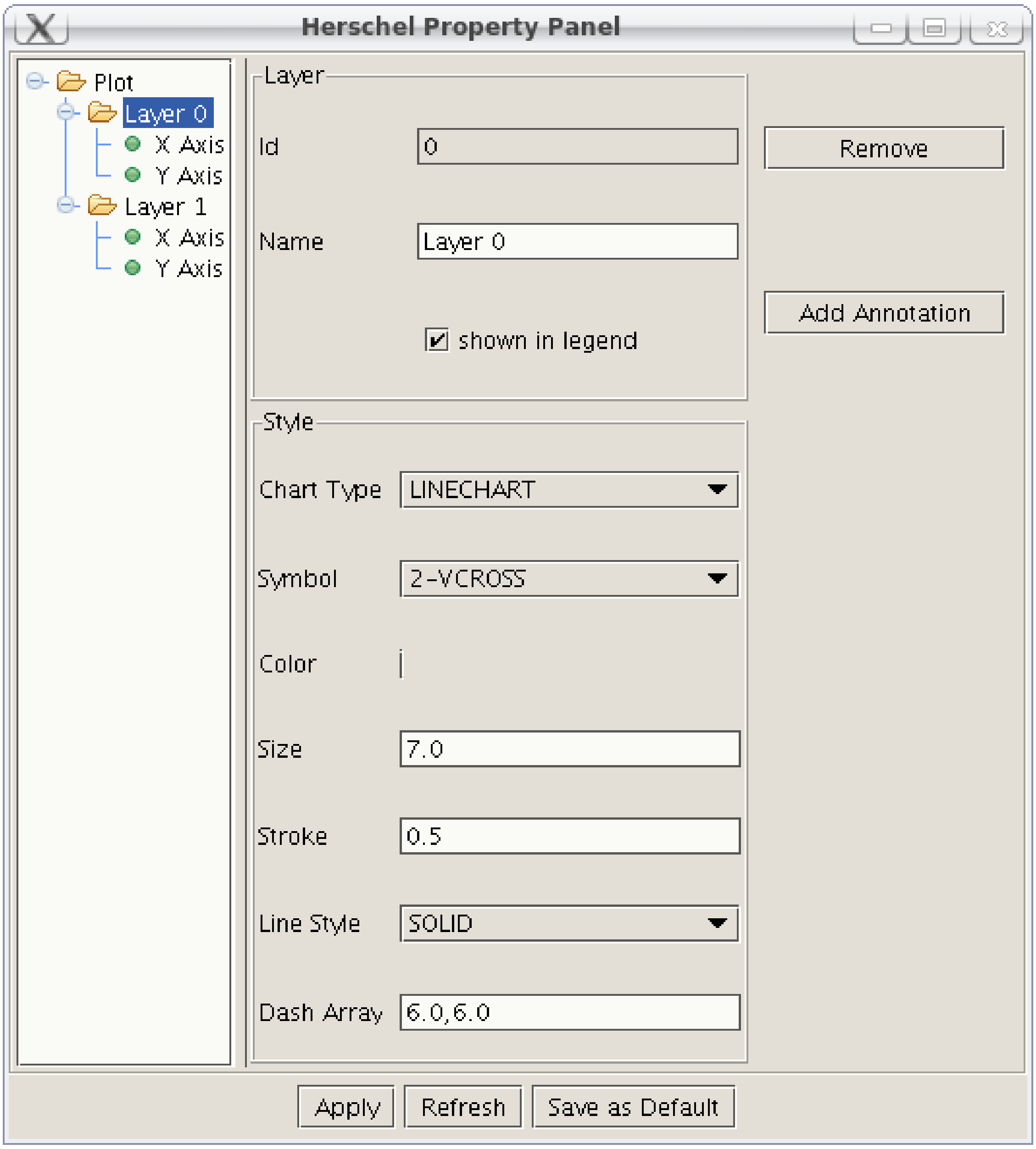


Figure 9.3. Layer's properties

These are the property entries at a layer level, they are the same for all layers. From this panel you can also add annotations to the plot. Note that this annotation is attached to the corresponding layer so any change in layer colour will affect the annotation as well.

3. Axes properties

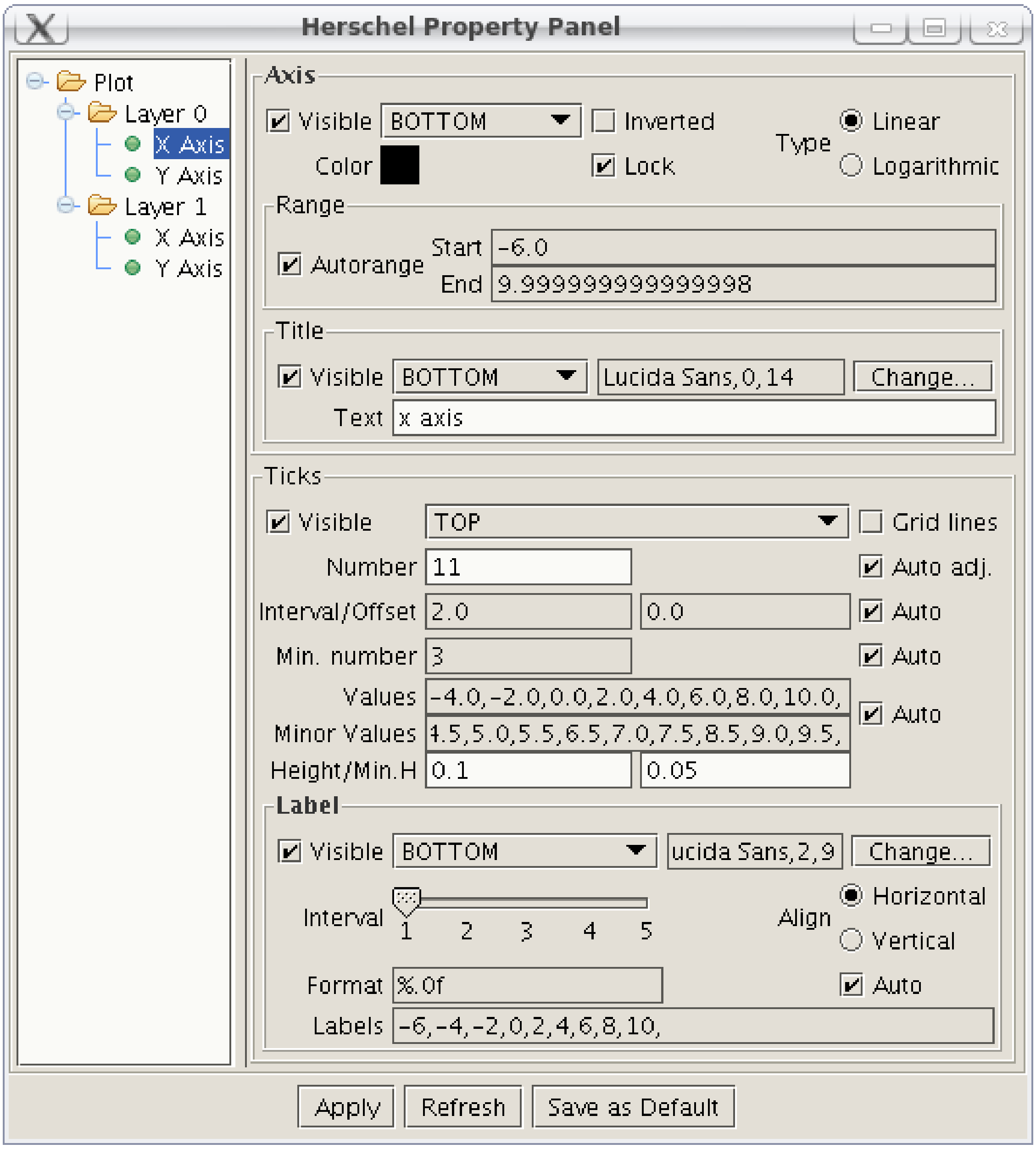


Figure 9.4. X-axis properties

Here you have almost complete control over the axis properties: range, title, tick marks and minor tick marks, axis label. And different radio buttons allow you to turn on/off auto features.

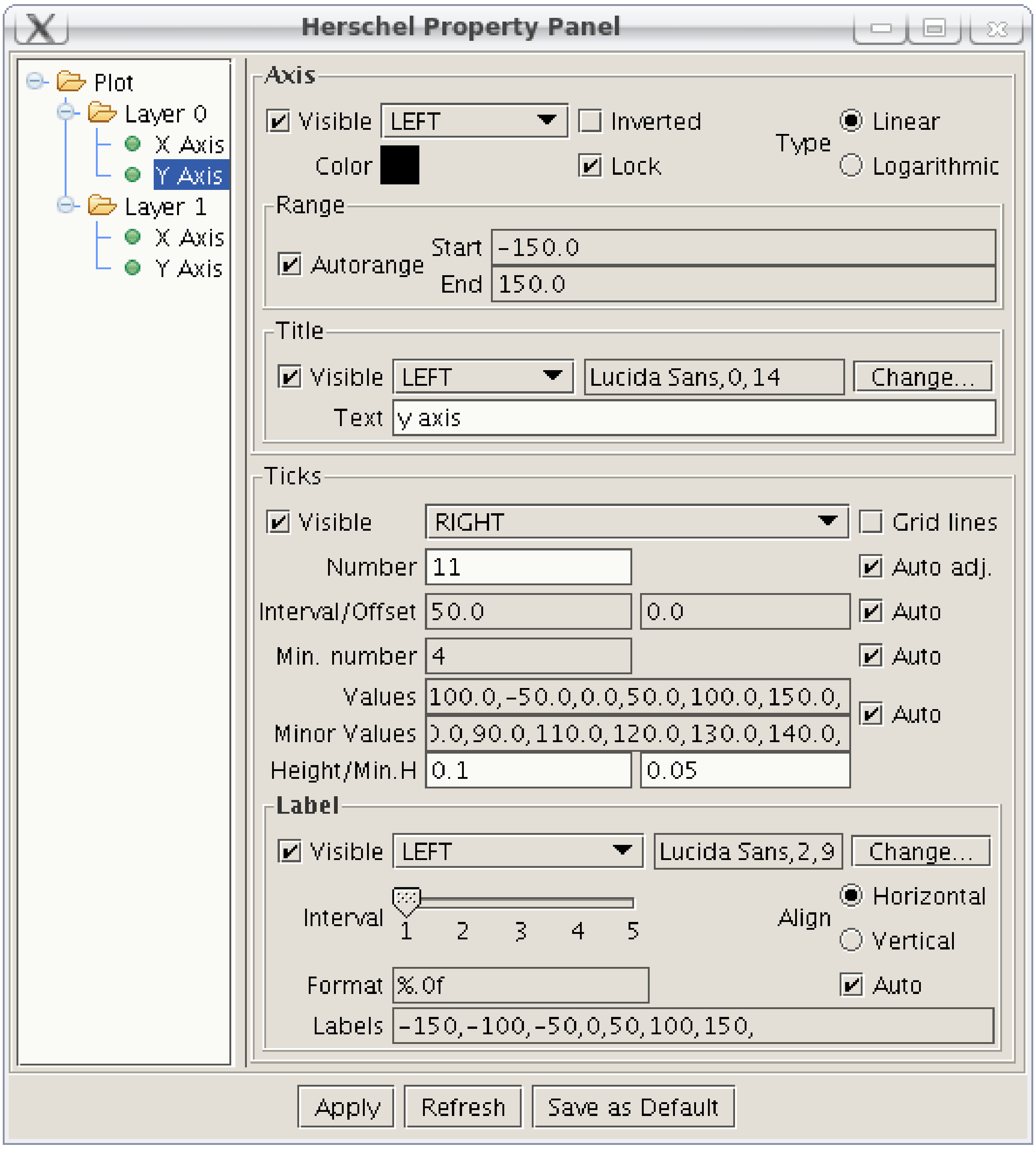


Figure 9.5. Y-axis properties

4. Printing of a plot. In the menu which pops up when you click with the right-hand side mouse button you have "Print..." menu which allows you to send the plot directly to a printer (if you have configured one for your system).

5. Saving the plot. In the menu which pops up when you click with the right-hand side mouse button you have "Save as..." menu which allows you to save the plot in different image formats: Encapsulated PostScript file (EPS), JPG or PNG files.

Note

For plots, layers, annotations or axis titles when using the TeX notation you should not escape the "\" symbol, that is you can directly use α in the text field of the GUI.

9.4. Advanced plotting

Here we introduce some more advanced plotting. Most of these are explained in greater detail and with examples in the DP User's Manual.

1. Multiple plots per window.

   When we add layers to the plot we can specify their position on a grid as in the example below which places 4 layers onto a 2x2 grid (running indeces from 0,0 to 1,1).

```
plot = PlotXY()
layer = LayerXY(x,y)
layer1 = LayerXY(x1,y1)
layer1x = LayerXY(x1,y1/5.0)
layer1y = LayerXY(x1/5.0,y1)
plot.addLayer(layer,0,0) # top left
plot.addLayer(layer,0,1) # top right
plot.addLayer(layer,1,0) # bottom left
plot.addLayer(layer,1,1) # bottom right
```

   Now, if we open the plot properties GUI we have all four layers and we can change each one of them if necessary. We can interact with each layer and change its properties following the command line methods too.

2. Create a plot in batch mode.

   This is useful when you have many layers to add to the plot and you want to avoid to have the plot window redrawn and reajusted each time a new layer is added. From the above example:

```
plot = PlotXY()
plot.setBatch(1)
layer = LayerXY(x,y)
layer1 = LayerXY(x1,y1)
layer1x = LayerXY(x1,y1/5.0)
layer1y = LayerXY(x1/5.0,y1)
plot.addLayer(layer,0,0)
plot.addLayer(layer,0,1)
plot.addLayer(layer,1,0)
plot.addLayer(layer,1,1)
plot.setBatch(0)
```

3. Different units or style for the upper/right axes.

   TBW.

9.5. Plotting table datasets - using the TablePlotter

A powerful tool exists which allows you to plot HCSS TableDataset products. This tool is called TablePlotter and there is an extensive documentation on it - the DP User's Manual and How to inspect and plot datset tables in HIPE document.

Chapter 10. How to inspect and plot datset tables in HIPE

Herschel Editorial Board

10.1. Introduction

This HowTo is a description of how to create and inspect a simple TableDataset in HIPE. It will walk you through the necessary steps to create a dummy TableDataset, if you don't already have one--using the command line window. We will show you how to manually inspect the values in the table and, then use TablePlotter to plot data within the table, NOTE: Currently it is not yet possible to run TablePlotter within HIPE, but this option will soon become available.

A `TableDataset` is made up of a number of columns. Each column contains an `ArrayDataset` (data), a description and a quantity value associated with the `ArrayDataset`. Each `ArrayDataset` can have up to 5 dimensions and can be of varying types.

Constructed on 2008/06/23 19:14...

10.2. Steps to creating and viewing a simple TableDataset with the HIPE GUI

These are the steps to follow to create, view, and plot graphs of a TableDataset within HIPE.

1. Step 1: Open HIPE's "Welcome" window and click on Workbench Icon

2. Next we assume here that you do not have a TableDataset loaded into your session. If you already have one loaded into HIPE, then skip to the next item. Otherwise read on. Type the following commands into command-line window containing the "IA>>" prompt (bottom center in the default view). In the example given here, we will create a `TableDataset` with 3 columns each containing a 1D dataset, one being a sequence of numbers from 1 to 100, the second being the sine value of each of the numbers in the first column, and the final column containing the values in the first column multiplied by 100. The column names are `x`, `sin` and `y` respectively.

```
from herschel.share.unit import *
x = Double1d.range(100)
t = TableDataset(description="This is a table")  # ❶
t["x"] = Column(data=x, unit=Duration.SECONDS)  # ❷
t["sin"] = Column(data=SIN(x),description="sin(x)")  # ❸
t["y"] = Column(data=x*100,description="x*100")
```

❶ This sets up the table dataset with an associated description

❷ This creates our first column which has the data, `x` and its associated units, which in this case is a time duration of SECONDS.

❸ Here we have applied the `SIN` function from the numeric package, and we have also added a description for the second column.

Notice that when you create the variable x and the TableDataset t, they appear in the "Variable" window in the top right.

3. Next we wish to view the table we have created. Move your cursor over the item "t" in the Variables window and right mouse click on it. Choose the OPEN WITH option in the drop-down menu and select Dataset Viewer. At his point a view of the table will appear in the Editor window and you can

scroll down and view the table, and expand it if necessary using the cursor and left-mouse clicking at the boundries of the window to re-size it.

4. NOT YET IMPLIMENTED Now we wish to view the table in the TablePlotter task. Again right-mouse click on the item "t" in the variable list and select OPEN WITH item "TablePlotter". This will bring-up the TablePlotter GUI in its own window??. A complete guide to the TablePlotter is found in the Herschel UserManual. We list below a brief guide to TablePlotter.

10.3. Guide to TablePlotter Controls and their functions

The TablePlotter provides the following control buttons to view and analyze data.

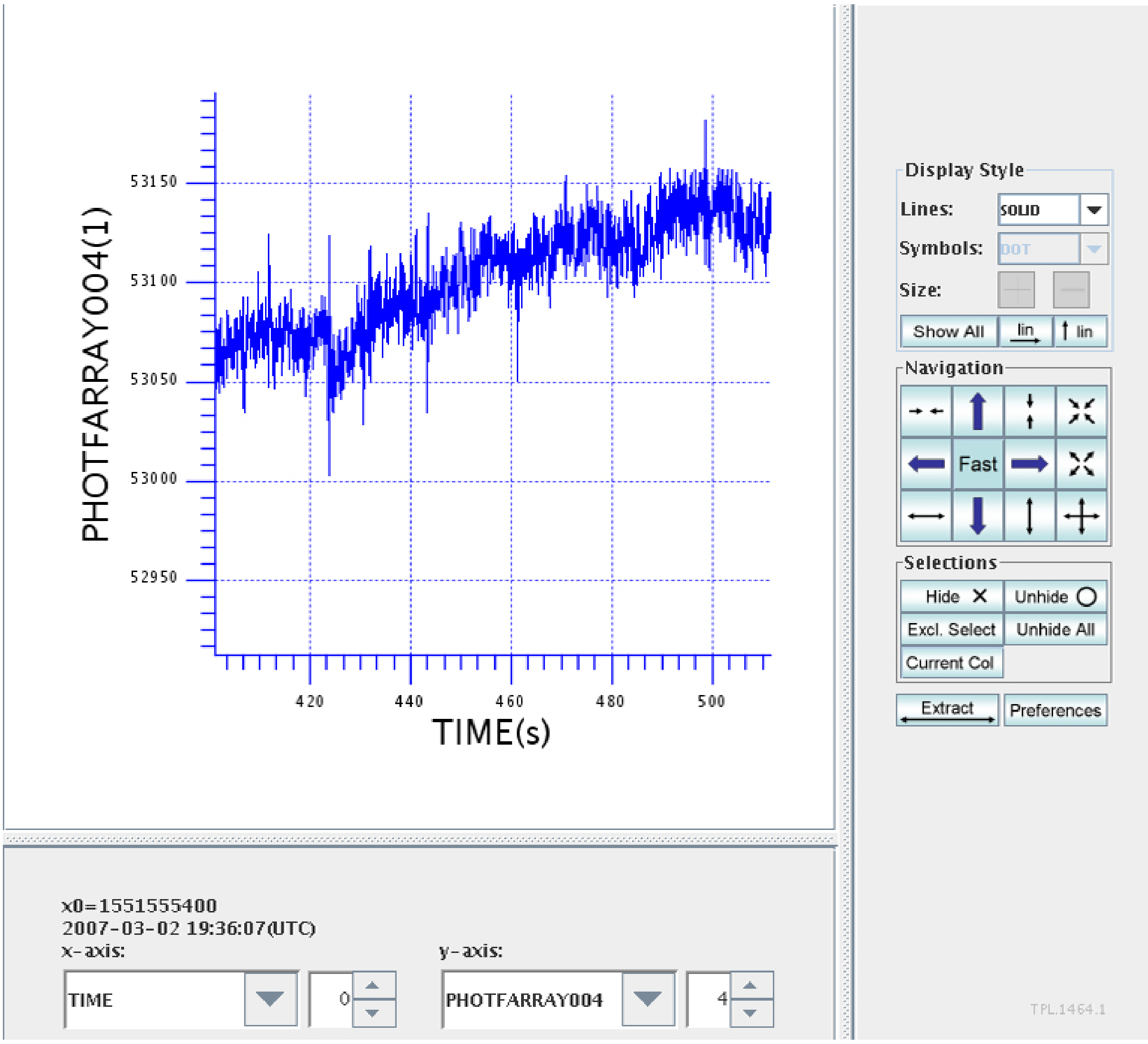


Figure 10.1. Example of the TablePlotterGui.

- **X and Y- Axis Selection:**

  Under the graphics display area, two sets of Combo Box buttons and spinner buttons allow users to select X and Y-axis data. The first column of the TableDataset is associated with X-axis by default. The second coumn is initially associated with the Y-axis.

  Users can choose a column by name in the Combo Box and by number in the spinner.

Fast forward/backward selection of columns in the spinner can be achieved by holding the left mouse button down and moving the mouse up or down to select.

- **Display Style:**

  The control buttons in this secion allow to change the axis style (linear or log), line style (solid or dashed and more), and symbol style.

  The default axis scaling is linear. The toggle buttons 

 / 

 and 

 / 

 allow to switch between linear and logarithmic scales in the X / Y axes respectively.

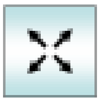
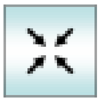
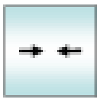
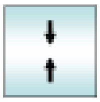
  The pull-down menus of Lines and Symbols allow to select line style and symbol style. The selection of symbol styles is only available when the line styles are either *MARKED*, *MARK_DASHED* or *NONE*. To increase or decrease the symbol size, click either  or .

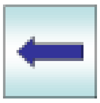
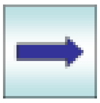
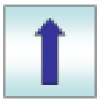
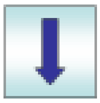
  Another toggle button 

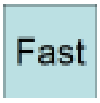
 / 

 determines whether all data points or only the selected ones are shown (see detail in Selections below).

- **Navigation:**

  Two buttons are provided to simultaneously zoom in  and zoom out  in both X and Y directions. The buttons  and  zoom out individually in either X or Y direction .

  Four buttons with arrows     pan the view onto the graph in X or Y direction.

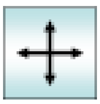
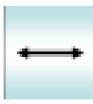
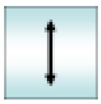
  The size of each zooming or panning step is controlled by a toggle button 

 / 

 and the exact factors of both fast and slow modes can be adjusted in the 

 menu (for details see the *Preferences* section below).

  To Zoom-in on specific areas of the graph, press the left mouse button, and hold-and-drag a rectangle around the area of interest with the mouse pointer.

  There are three *Free Scaling* buttons. The  button will adjust the scales of both axes such that all visible datapoints are optimally distributed within the display, while  and  will do the same but for either X or Y axis alone.

- **Selections:**

The selection feature of TablePlotter allows to hide or select a particular portion of the data points. In combination with Show All / Sel Only in the Display Style section, and Multi Column Mode, this feature can be used to display only selected data to get fast automatic scaling when scanning through many columns of data. The main purpose, however, is the extraction of specific data points into new datasets. A typical purpose could be for instance to remove electronic glitches from detector data, or to extract a specific piece of signal from a sequence of instrument configurations.

The following buttons Hide ✕ Unhide ◯ Excl. Select, hide, un-hide or exclusively select all data points within a rectangular area in the plot. This area is selected after pushing one of those three buttons by holding and dragging the mouse pointer in the same way as for zooming in.

Clicking the button Unhide All will re-select all hidden data points.

If the Current Col toggle button is visible, the TablePlotter is in single column mode. In this mode hiding or selecting operations will only apply to the current column. Clicking on this button will toggle into all columns mode and the button will change to All Cols Now all the columns are affected and selections are done based on the selected intervals on the X-axis only. The Y-coordinate will be ignored in this mode.

In Show All mode the hidden data points will be marked with red symbols. See Figure 10.2 below. Clicking on Show All toggles to Sel Only mode, where all hidden data points disappear from the graph.

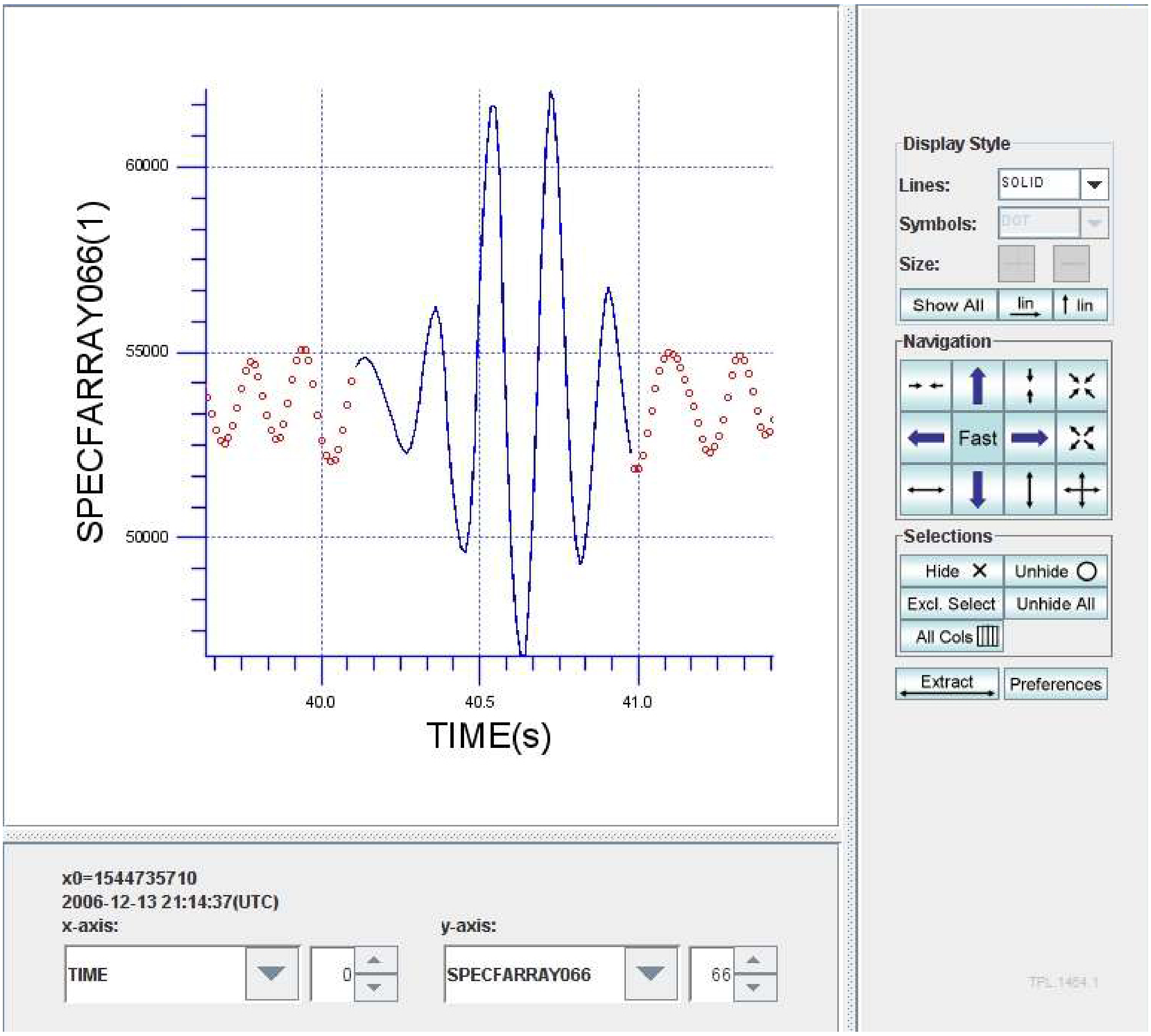


Figure 10.2. The plot with selected and hidden data points.

- **Dataset Extraction:**

  To extract a subset of the data after performing the necessary selection operations, press the  button. The selected data will be extracted into a new dataset that will be fed back to DataInspector, where it will appear in the leftmost panel under "Datasets".

  If  is selected, only the selected data points in the currently displayed column will be extracted.

  If  is selected, the selected data points in all the columns become available for extraction. After clicking , a column selection window will pop up to allow users to **Add** individual columns or **Add All** columns to a list. Users can also **Remove** individual columns or **Remove All**. **Up** and **Down** buttons allow to change the order of columns in the new dataset.

  Hitting the **Close** button will complete the extraction and an option is provided to change the default name of the new dataset.

- *Preferences:*

  Finally the Table Plotter proivides a Preferences menu with two options. The first one is Set properties... where preferred zooming and panning factors for Fast and Slow modes can be set.

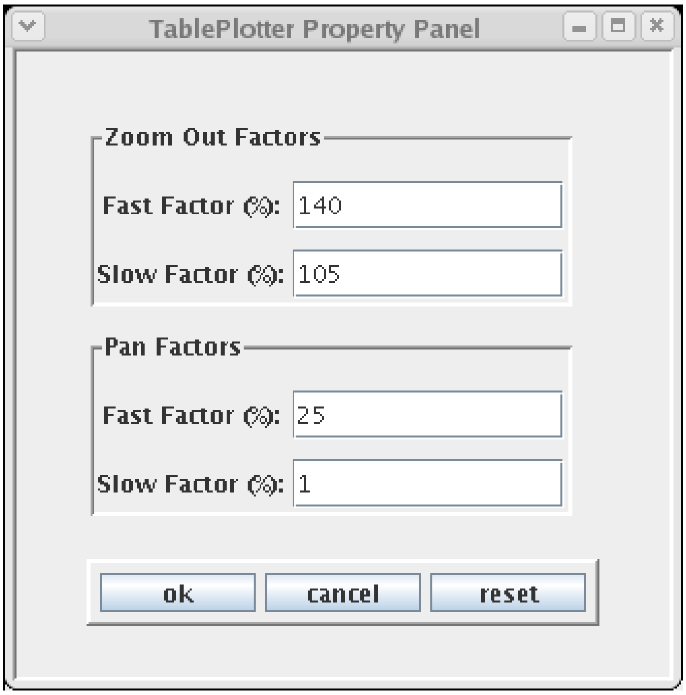
  

  Figure 10.3. Preferences: Set Properties

  The second one controls the display of Complex Data. TablePlotter allows only one graph to be displayed at a time. Here the user has three choices: plot modulus only, plot real part only, or plot imaginary part only.

  The selected preferences are stored in a properties file and will be "remembered" in the next call to Table Plotter.

Chapter 11. HowTo Display Spectra

11.1. Introduction

HIFI spectra can be visualised in several ways, at various levels of sophistication and user-friendliness. At the lowest level, individual X and Y axis values can be extracted and units applied and the basic plotting facilities of the PlotXY package can be used (see "HowTo Plot" and the "HIFI DP User's Manual" for more details). However, a simpler way for most users of HIPE is to use the "Spectrum Explorer" package.

11.2. Obtaining a Spectrum from an `ObservationContext`

Most users will obtain spectra from downloading observations from the Herschel Science Archive (HSA). The main product form of an observation is referred to as an `ObservationContext`. An `ObservationContext` contains all the components of an observation, including all calibrations needed for repeating pipeline processing data. The full observation download from the HSA includes all levels of processed data from level 0 (raw data) to level 2 (final pipelined product).

An observation context can contain many spectral products (e.g., from all 4 spectrometers of HIFI) at each of the different levels of processing.

We can consider a HIFI `ObservationContext` called "prod" which has been downloaded from the HSA. A double-click on the variable name "prod" in the "Variables" view of HIPE provides an outline view of its contents in the "Outline" view (see Figure 11.1). This shows the containers of spectra at the different processed levels (also quality and calibration information associated with the observation). If we now open the level 2 folder and click on the product (highlighted in Figure 11.1), we get a view in a new "Editor" window like the one shown in Figure 11.2) -- after expanding out the folder labeled 1030.

At present, the HIFI spectrometers are identified by the values 1028 = HRS H polarization, 1029 = HRS V polarization, 1030 = WBS H polarization and 1031 = WBS V polarization. In this case the 1030 folder under level 2 is the final, processed data product from Standard Processing at the Herschel Science Centre of WBS H polarization data. The folder labeled "1" contains the first (and in this case only) final product. A double-click on product(load) -- highlighted in Figure 11.2) -- provides access to a listing of the metadata and the final dataset (scroll to bottom of "Editor" window) which is marked with a green dot beside it.

We can display this final spectrum via a viewer called `Spectrum Explorer`. As with all viewers in HIPE -- click with right mouse button on the dataset word, choose "Open With..." from the menu that appears and then click on the words "Spectrum Explorer". This will display a new "Editor" window with a blank spectrum display (initially). To fill in the spectrum the user needs to click on the four boxes (label above shows 1, 2, 3 and 4) to the left under the plot or simply clicking on the box under "ALL". This fills in the 4 sub-bands of 1GHz wide CCDs that make up the full backend spectrum from a WBS spectrometer (see Figure 11.3). An appropriate legend is automatically created.

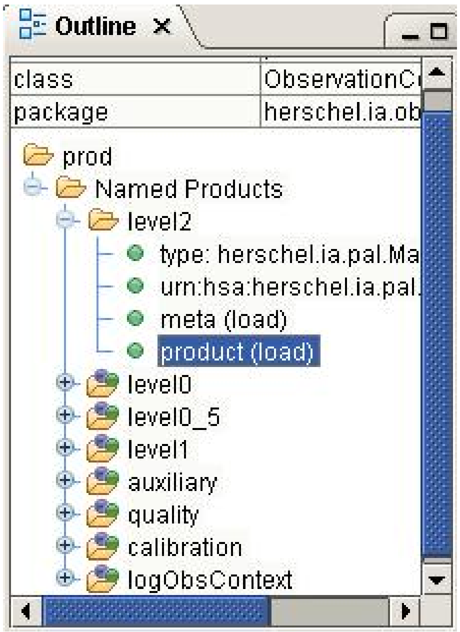


Figure 11.1. Accessing the level 2 (final) processed product from an observation.

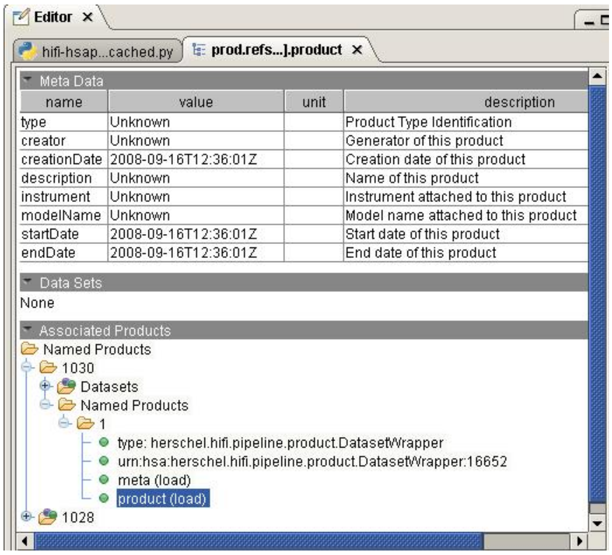


Figure 11.2. Editor view showing access to the final WBS H polarization spectrum from within the full observation tree.

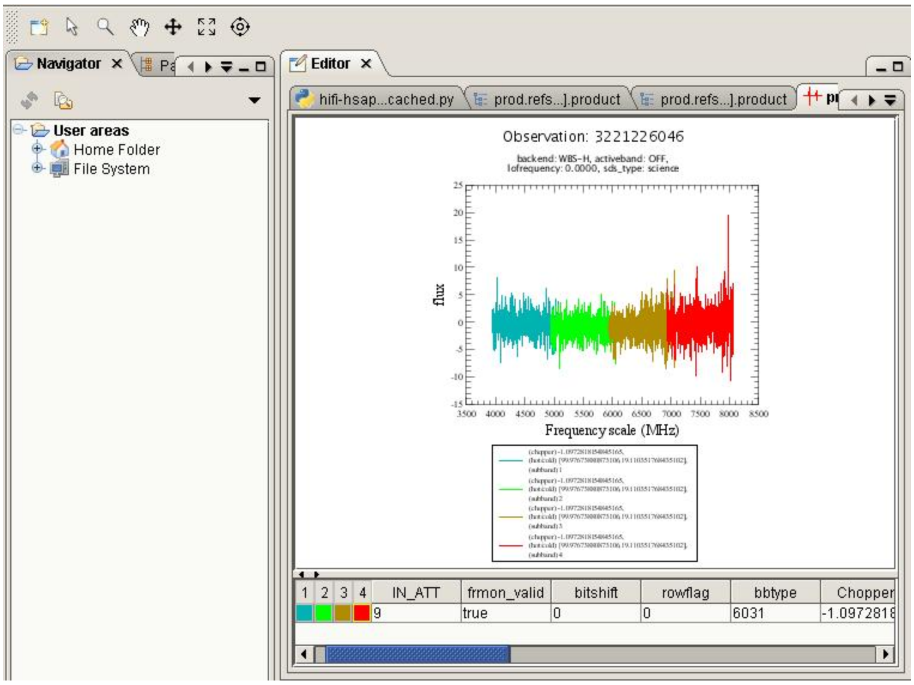


Figure 11.3. Display of a test data (no source) produced by the HIFI pipeline using the `SpectrumExplorer`.

11.3. The `SpectrumExplorer` Package

The `SpectrumExplorer` package is based on the PlotXY package, but allows the user to visualize `SpectrumDatasets` in a friendlier, interactive way.

In the example from the previous section, the different colors indicate different WBS sub-bands. Individual sub-bands or individual scans can be plotted by clicking on the appropriate boxes in the bottom panel and removed by double-clicking. Any plot parameter (plot range, titles, colors etc.) can be modified using the right mouse bottom in the same way as for the PlotXY package (see "HowTo Plot Data"). For example, a right-click on the plotted spectrum allows changes in the axes and plot properties (e.g., labels fonts etc.). It is also possible to save and print the plot from the right-click menu.

In addition the plot can be modified interactively after clicking the appropriate action button which `Spectrum Explorer` places in the top left of the HIPE display (see top left of Figure 11.3). Hovering the mouse over the icons allows provides the user with a tooltip for what the icon allows you to do. From left to right:

- button 1: highlight/select a spectrum (or WBS sub-band) by moving the mouse over it and click the right mouse button to change its color, description, or remove it.
- button 2: change the horizontal and vertical plot ranges by drawing a rectangular box using the left mouse button. Also, one can scroll the spectrum along the horizontal and vertical axes by clicking on an axis with the left mouse button and then moving the mouse or using the mouse wheel. The mouse wheel can also be used to (un)zoom the spectrum.
- button 3: pan through the spectrum by clicking the left mouse button and moving the mouse.
- button 4: click on a spectrum (or WBS sub-band) and drag it to right or down to another or a new panel which is automatically generated on release (however, dragging to the left or top of the first panel is not possible).
- button 5: click on this button to auto-range the displayed spectra (after zoom).
- button 6: only show the active plot panel, and change the axis ratio in order to fit the screen.

This allows images such as the Figure 11.4 to be constructed from the displayed data.

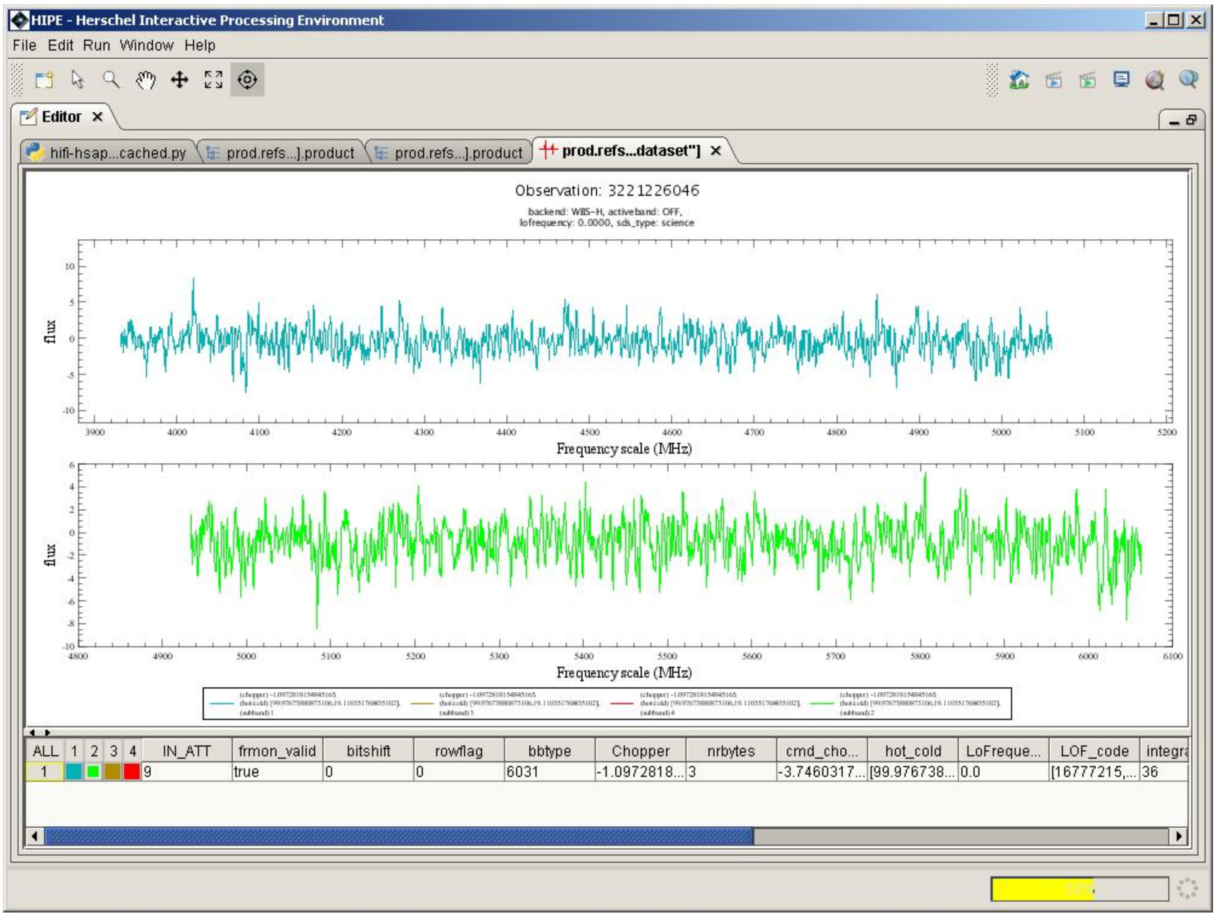


Figure 11.4. Two sub-bands extracted interactively (button 4) using `SpectrumExplorer` shown with the full resolution of the screen (button 6).

11.4. Future developments

Finally, the `SpectrumExplorer` package is still under development. Future developments include:

1. clicking on product will plot all SpectrumDatasets included (likely a HIPE functionality, not JIDE)
2. apply a filter to the meta data and only plot the applicable spectra (e.g. a certain chopper position)
3. applying functions to greyed-out/flagged spectral regions
4. saving modified `SpectrumDatasets` back into the session, e.g. with interactively masked points removed.
5. overplotting multiple `SpectrumDatasets`.

Chapter 12. Spectral Arithmetic and Mathematical Operations

12.1. Introduction

The spectrum arithmetic toolbox allows us to combine Herschel spectrum data. Operations are performed either on subclasses of spectrum datasets (`Spectrum1d, Spectrum2d`), on cubes (`SimpleCube, SlicedCube`), or on products containing such data structures (e.g., HifiTimelineProduct).

Operations on Spectra include Selection and Arithmetic Operations.

This chapter explains how to work with spectra so that basic spectral arithmetic can be done on a 1D spectrum dataset. It also indicates how to handle datasets composed of multiple 1D spectra. When working with these larger sets of 1D spectra it is also possible to select spectra based on information held in the data or metadata of the individual spectra before applying the arithmetic transformations.

12.2. Starting point -- using a dataset of a number of HIFI spectra.

It is assumed that an observation product containing spectral data is available and active within your HIPE session. For this HowTo, we will have an active variable called "prod" which is a HIFI observation downloaded from the HSA (see HowTo Access Data). This contains several levels of data processing. We will be dealing with level1 data -- double-click on the highlighted "product(load)" in Figure 12.1. The results appear in a new Editor window and include some metadata on the product plus (scrolling down) a set of associated products (see Figure 12.2). Clicking on the highlighted "summary" will provide a list of what datasets are contained for apid=1030 (the WBS spectrometer H polarization). In the particular case (a Double Beam Switch observation) we are using we see that there a comb (frequency calibration measurement), a hot-cold internal calibrator measurement (hc), a tuning measurement (other) and two science measurements datasets for ON and OFF target (datasets 4 and 5). We will pick out dataset 4 for our purposes (double-click highlighted "product(load)" gives Figure 12.3). This produces a list of metadata for the selected product and a dataset (with green dot beside it) at the bottom of another Editor window. Drag-and-drop the dataset to the "Variables" view and this dataset is automatically given a name in the session -- typically "newVariable."

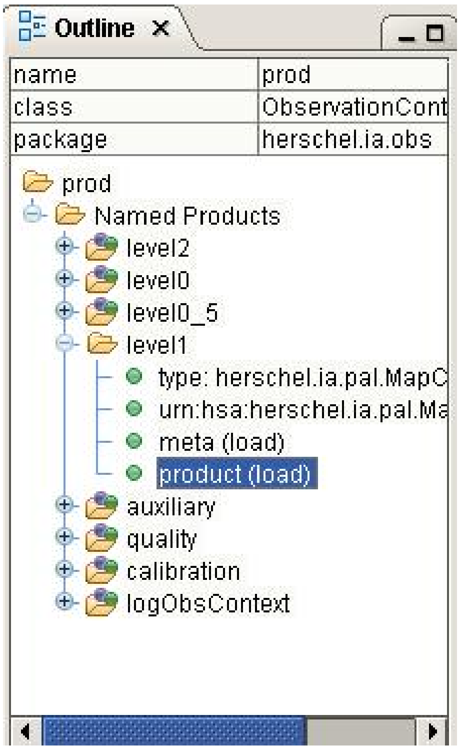


Figure 12.1. Selecting Level 1 data from a downloaded archive observation done by HIFI.

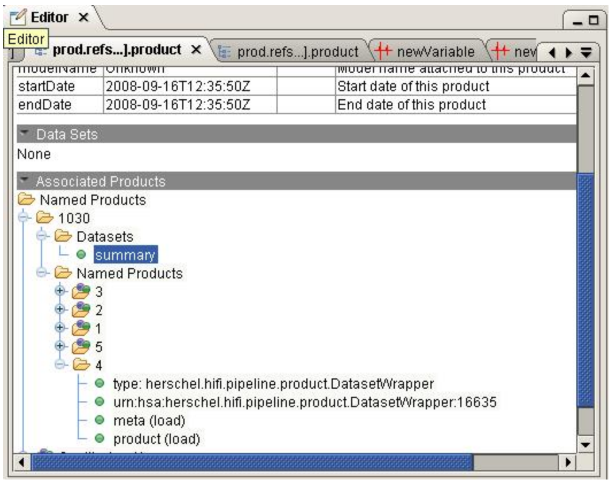


Figure 12.2. Display of product set.

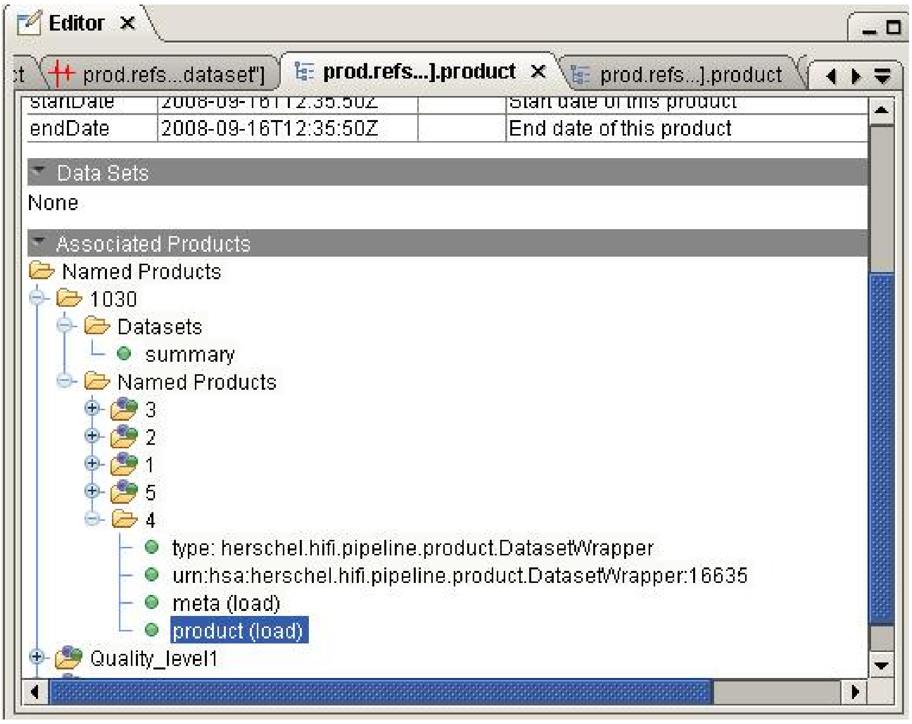


Figure 12.3. Choosing the product with the dataset we want.

A double-click on newVariable in the "Variables" view will open the dataset using the `SpectrumExplorer` (see HowTo on Spectral Display for information on how to manipulate the visualization). In the example dataset used here there are 18 spectra.

12.3. Using HIPE to Access the Spectrum Arithmetic Tasks

In HIPE the "Tasks" view gets filled with the currently available tools. Tasks that are available for use on datasets of spectra will appear under the "Applicable Tasks" folder that appears in the Tasks view. Available tasks include add/subtract/multiply/divide/average which can be seen in the Applicable Tasks folder after highlighting "newVariable" (see Figure 12.4).

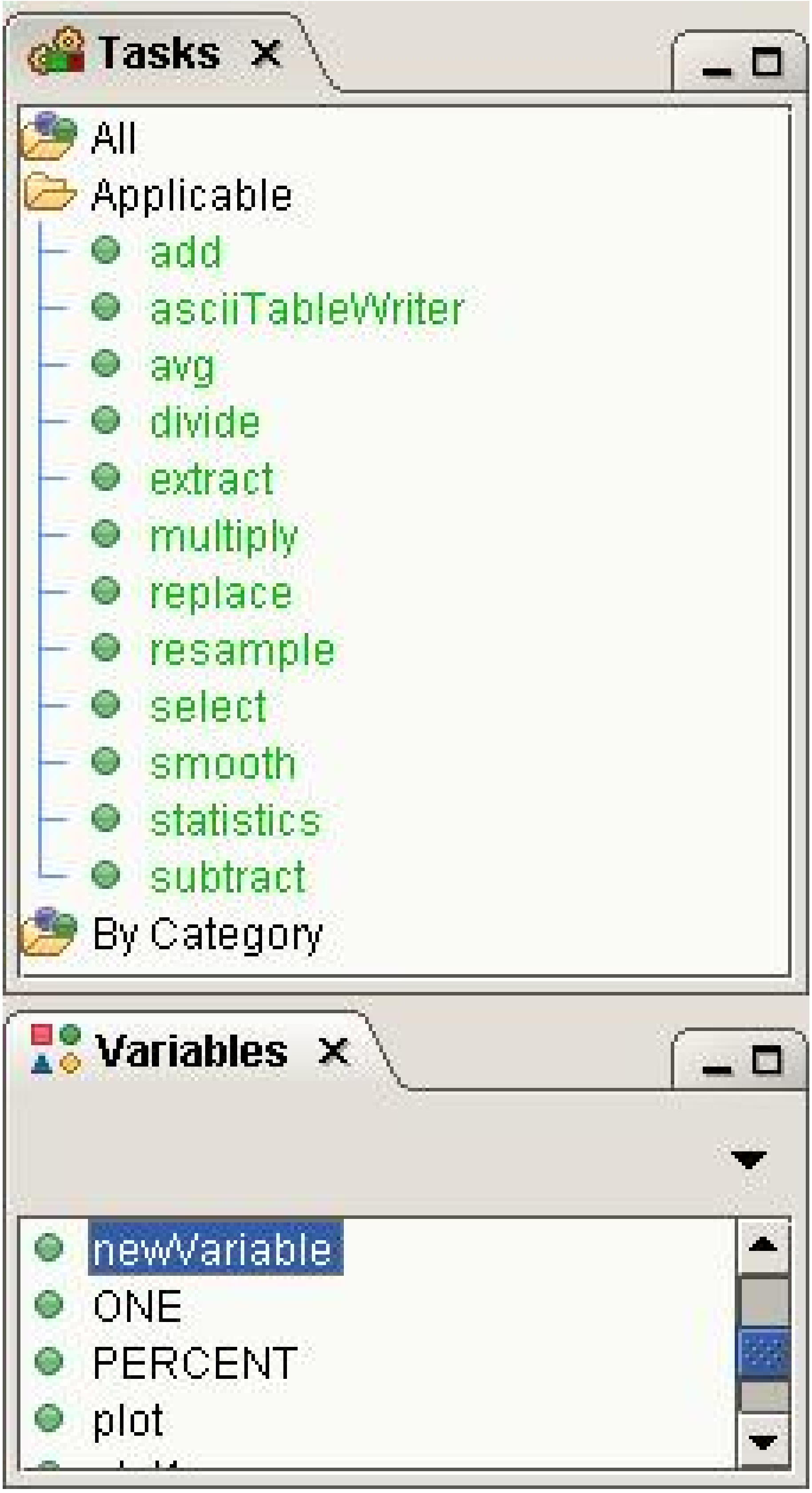


Figure 12.4. Display of tasks available for our dataset of spectra.

In this section we discuss each of the available arithmetic tasks in turn.

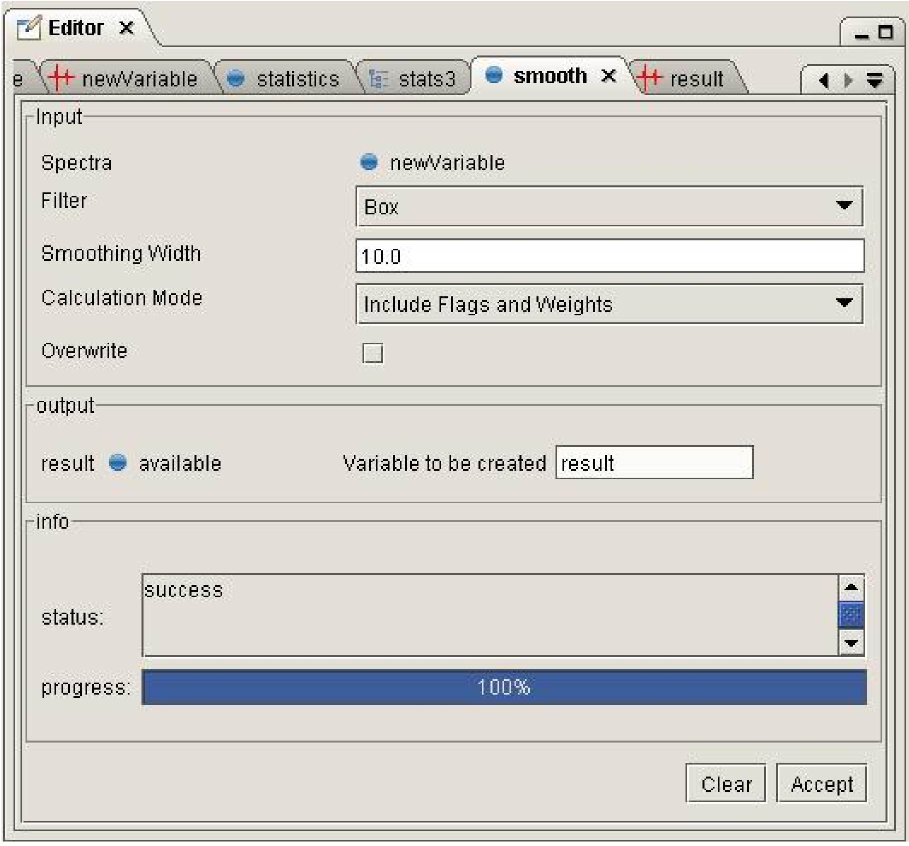


Figure 12.7. Using the `smooth` task

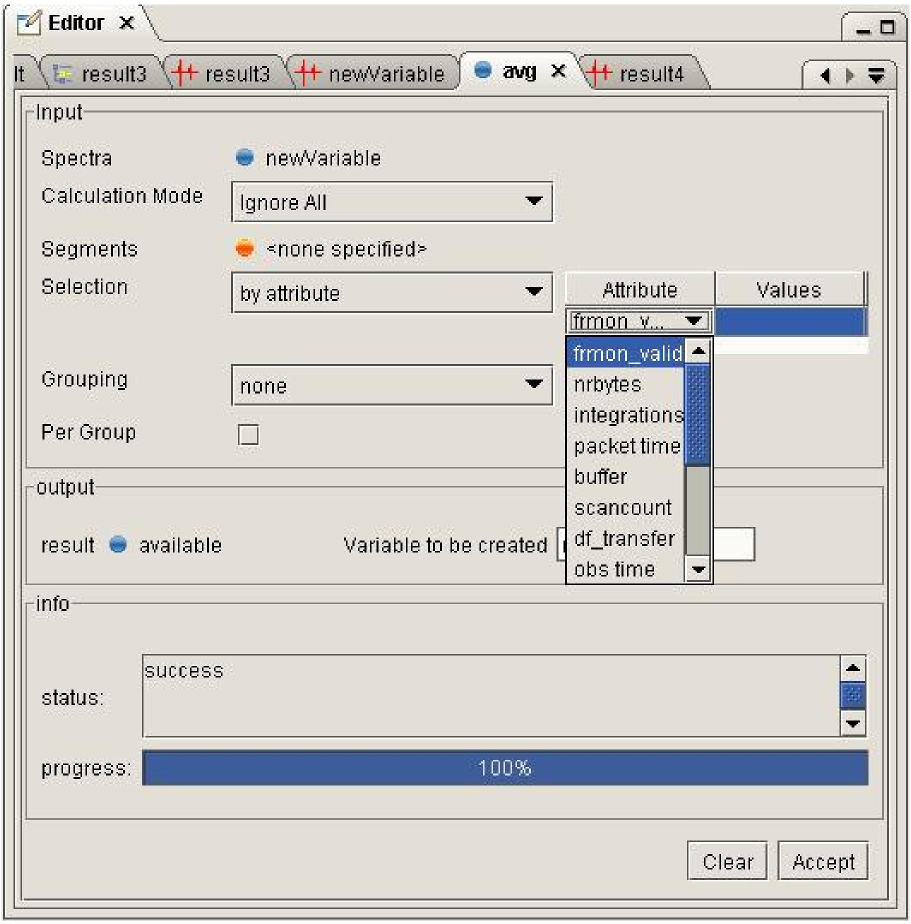


Figure 12.8. Using the `avg` task

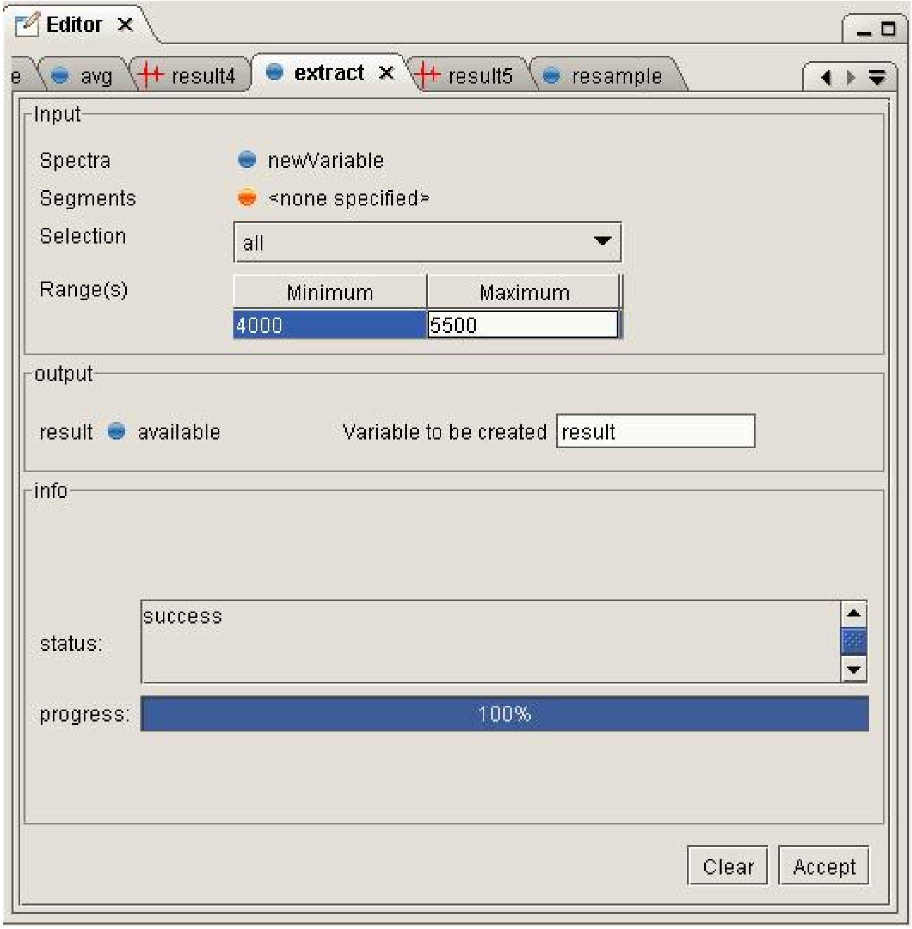


Figure 12.9. Using the `extract` task

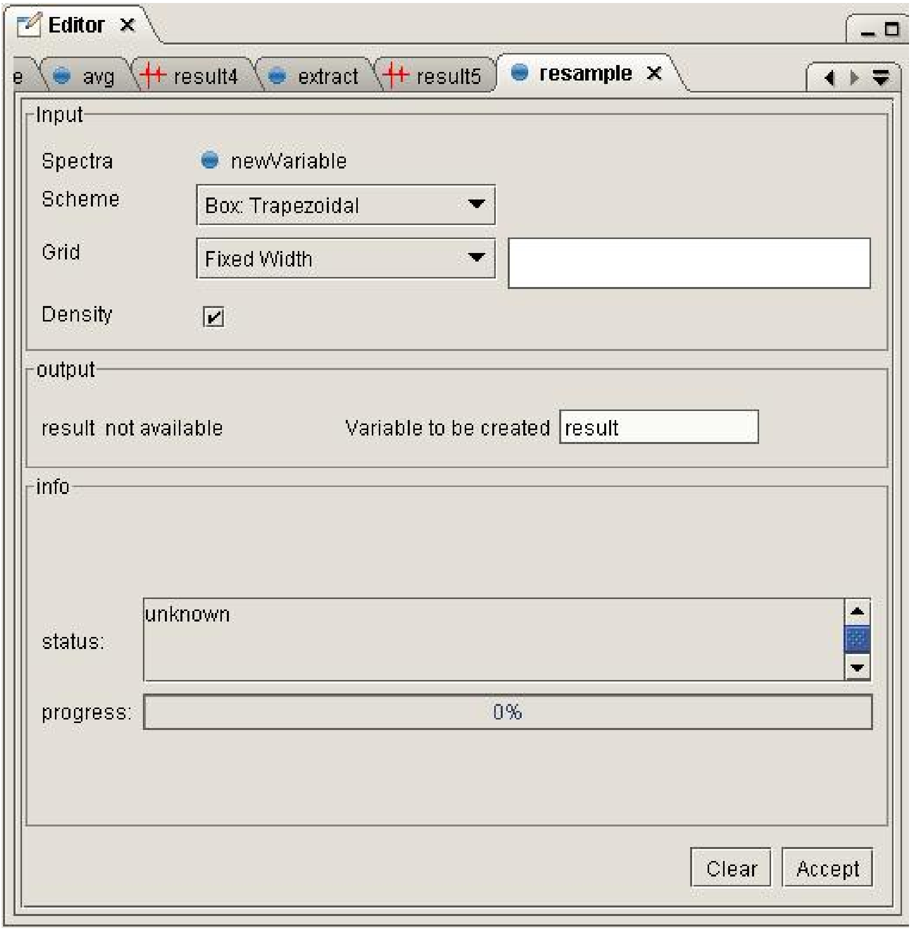


Figure 12.10. Using the `resample` task

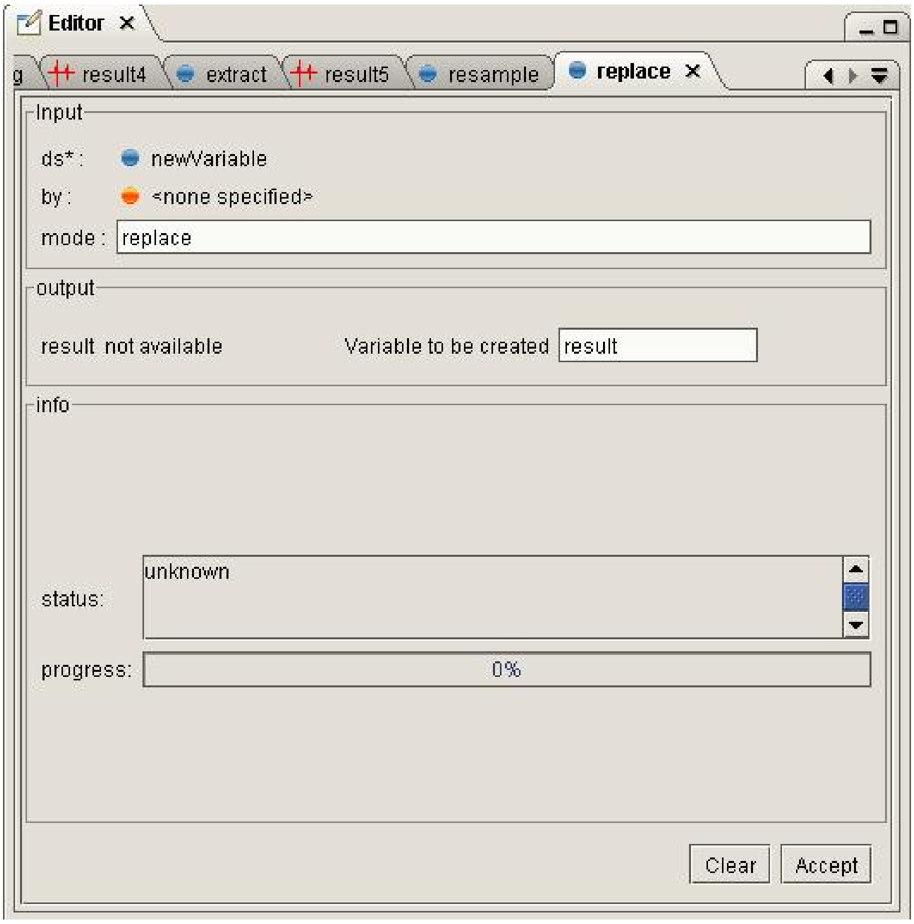


Figure 12.11. Using the `replace` task

- *select*: Provides a means of selecting those spectra that can be combined. A given attribute value or range of values can be used or simply the index number of the spectrum within the group (see Figure 12.5).

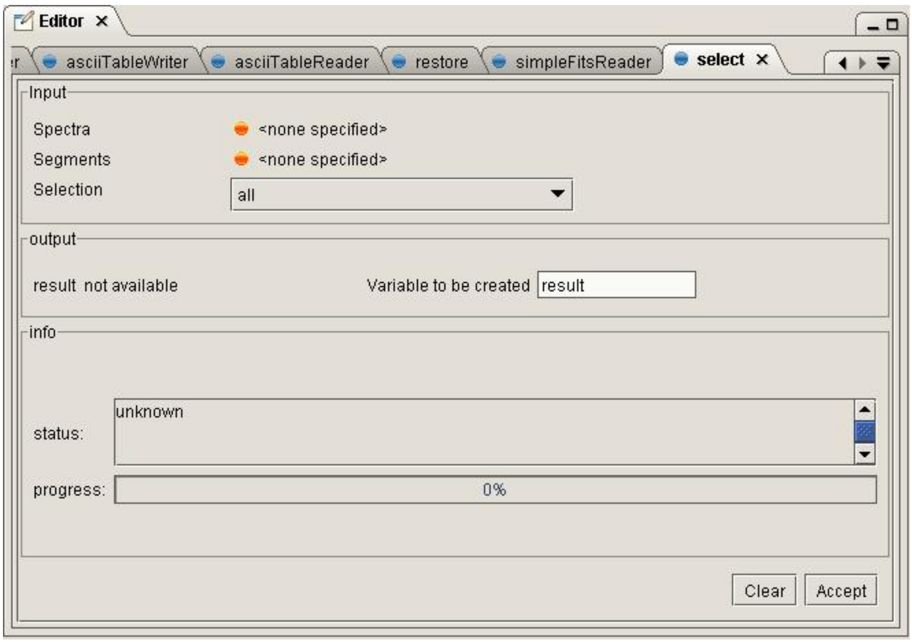


Figure 12.5. Using the `select` task

- *add/subtract/multiply/divide*: Provide means of adding/subtracting/multiplying/dividing groups of spectra or single spectra together (pair-wise), or adding/subtracting/multiplying/dividing a scalar value to/from all spectra in the selected dataset. Numbered segments, e.g., subbands, can be selected for addition if available within the dataset (see Figure 12.6 for adding the scalar value 200.0 to all spectra in our dataset)

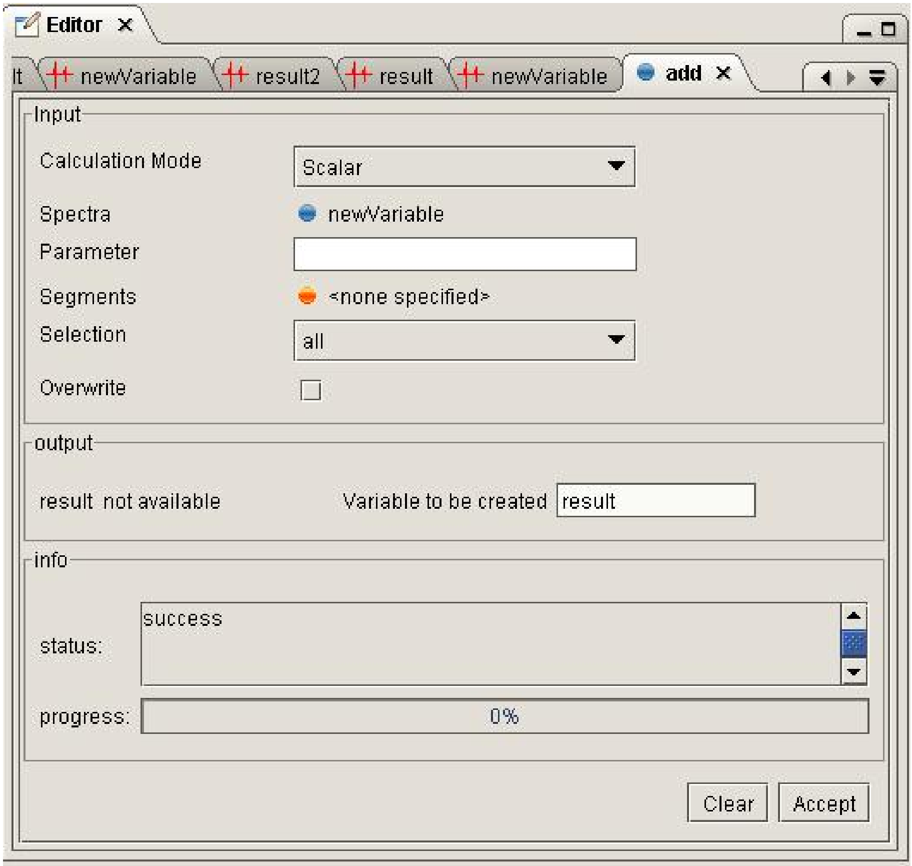


Figure 12.6. Using the `add` task

- *statistics* This allows for statistical operations to be performed on the datasets (it automatically works on individual sub-bands presently). It provide as mean, median, variance, standard deviation or percentiles for samples / selections of spectra from a dataset that can contain many datasets (spectra) when the "Accept" button is clicked. The result is an output that contains a number of datasets holding statistical information on the datasets. The main output is the "summary" table that is typically the last dataset listed of the set (double-clisk on output variable, e.g., "stats", in the Variables view. Use an appropriate viewer (Dataset viewer or Tableplotter to see the results).

- *smooth*This allows a transformation of the data via a box or gaussian (of user-selected width) smooth of the spectra in a dataset. Flags and weights for the different spectral points can be added in the future. To run this tool, click on the dataset, e.g., "newVariable", in the "Variables" view to highlight. The Applicable Tasks in the "Tasks" view include `smooth`. Double-click on this to get the self-explanatory dialog shown in Figure 12.7. The task runs by hitting the "Accept" button.

- *avg*This allows the average of a selection of spectra from a dataset. Flags and weights for individual channels/pixels can be used if available. Spectra can be selected by their index number in the dataset or by attributes (such as buffer number -- a pull-down selection list is available.). To run this tool, click on the dataset, e.g., "newVariable", in the "Variables" view to highlight. The Applicable Tasks in the "Tasks" view include `avg`. Double-click on this to get the self-explanatory dialog shown in Figure 12.8. The task runs by hitting the "Accept" button.

- *extract*This allows the extraction of a data from a minimum to a maximum frequency/wavelength range for the complete set of spectra in a dataset. Flags and weights for individual channels/pixels can be used if available. Spectra can also be selected by their index number in the dataset or by attributes (such as buffer number -- a pull-down selection list is available.). To run this tool, click on the dataset, e.g., "newVariable", in the "Variables" view to highlight. The Applicable Tasks in the "Tasks" view include `extract`. Double-click on this to get the self-explanatory dialog shown in Figure 12.9, where the channels with frequencies 4000 to 5500 MHz have been selected. The task runs by hitting the "Accept" button.

- *resample*This allows the resampling of data using a Trapezoidal or Euler box, with a choice of variable or fixed width. Flags and weights for individual channels/pixels can be used if available. Spectra can also be selected by their index number in the dataset or by attributes (such as buffer number -- a pull-down selection list is available.). To run this tool, click on the dataset, e.g., "newVariable", in the "Variables" view to highlight. The Applicable Tasks in the "Tasks" view include `resample`. Double-click on this to get the self-explanatory dialog shown in Figure 12.10. The task runs by hitting the "Accept" button.

- *replace*This allows the replacement of certain frequency/wavelength channels. To run this tool, click on the dataset, e.g., "newVariable", in the "Variables" view to highlight. The Applicable Tasks in the "Tasks" view include `replace`. Double-click on this to get the dialog shown in Figure 12.11. The task runs by hitting the "Accept" button.

It is planned that the arithmetic toolbox will provide generic functionality for all instruments (HIFI, PACS and SPIRE). Instrument-specific behavior will be pre-configured by defaults in the system but will be able to be overwritten by the user.

Full command-line versions of the spectral arithmetic tools is also available and is described in the "DP Basic User's Manual."

Chapter 13. HowTo Fit Spectral Features

Spectral features (baseline, lines and noise) are fitted using the *spectrum fitting toolbox* in the HCSS.

The data that is used by toolbox can be any Java or Jython object, as long as it implements the SpectralSegment interface (e.g., extracted from a Spectrum1d object). An example of a SpectralSegment could be the spectrum from one subband of the HIFI WBS spectrometer.

13.1. How to fit spectra in HIPE

1. Select the spectrum to fit from the variable list and then double click on the task fitSpectrum (Figure 13.1).

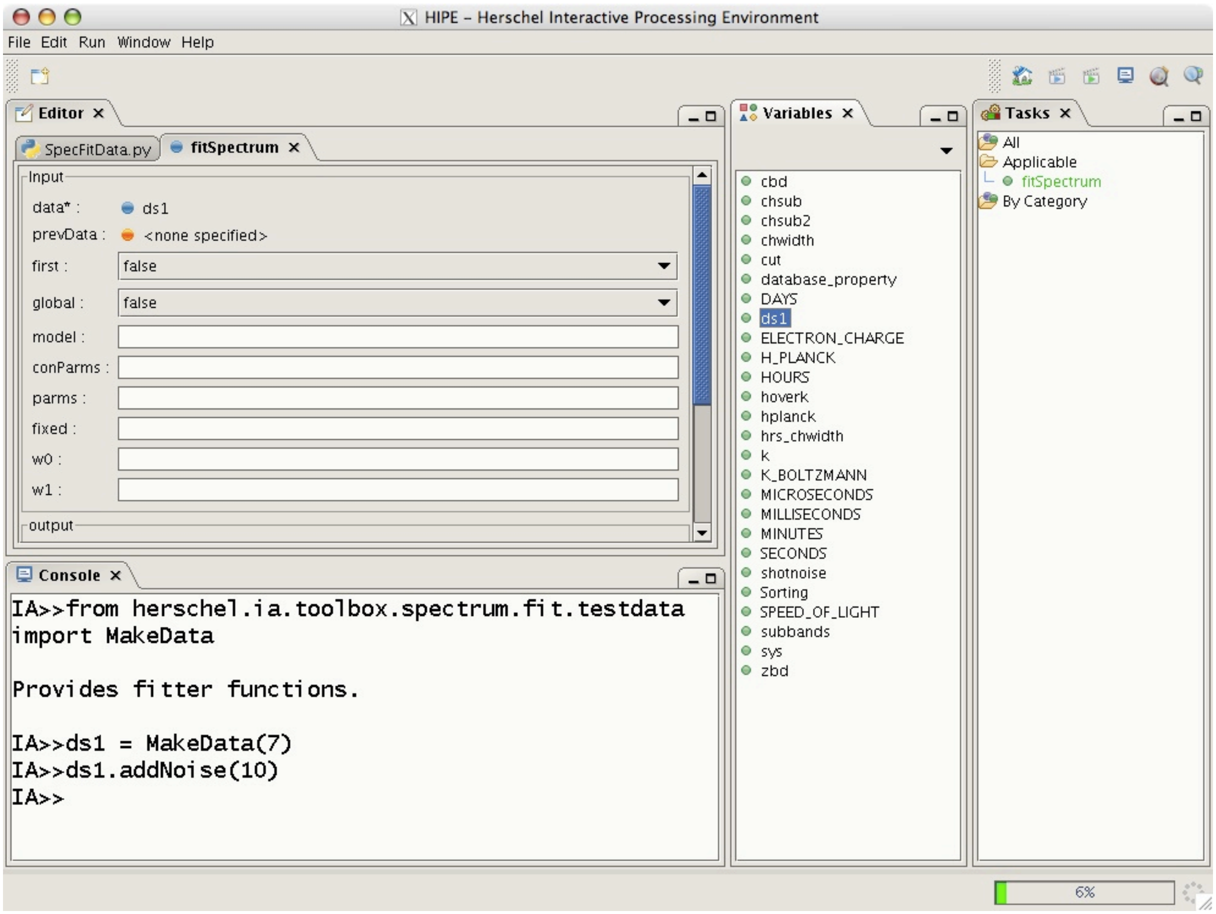


Figure 13.1. Starting fitSpectrum. Select "fitSpectrum" from the task list. This can most easily be found in the "Applicable Tasks" folder.

2. Apply a model to the spectrum via the GUI that pops up.

   All 1D models that are in ia.numeric.toolbox.fit can be selected from the drop-down box 'Use model'. The default is a Gaussian ("gauss") model, for which the 'Height of peak', 'X-Position of peak', and 'Width (sigma)' must be defined. The height and width have the value '1' already filled in, supply a value for the position and click RUN (Figure 13.2).

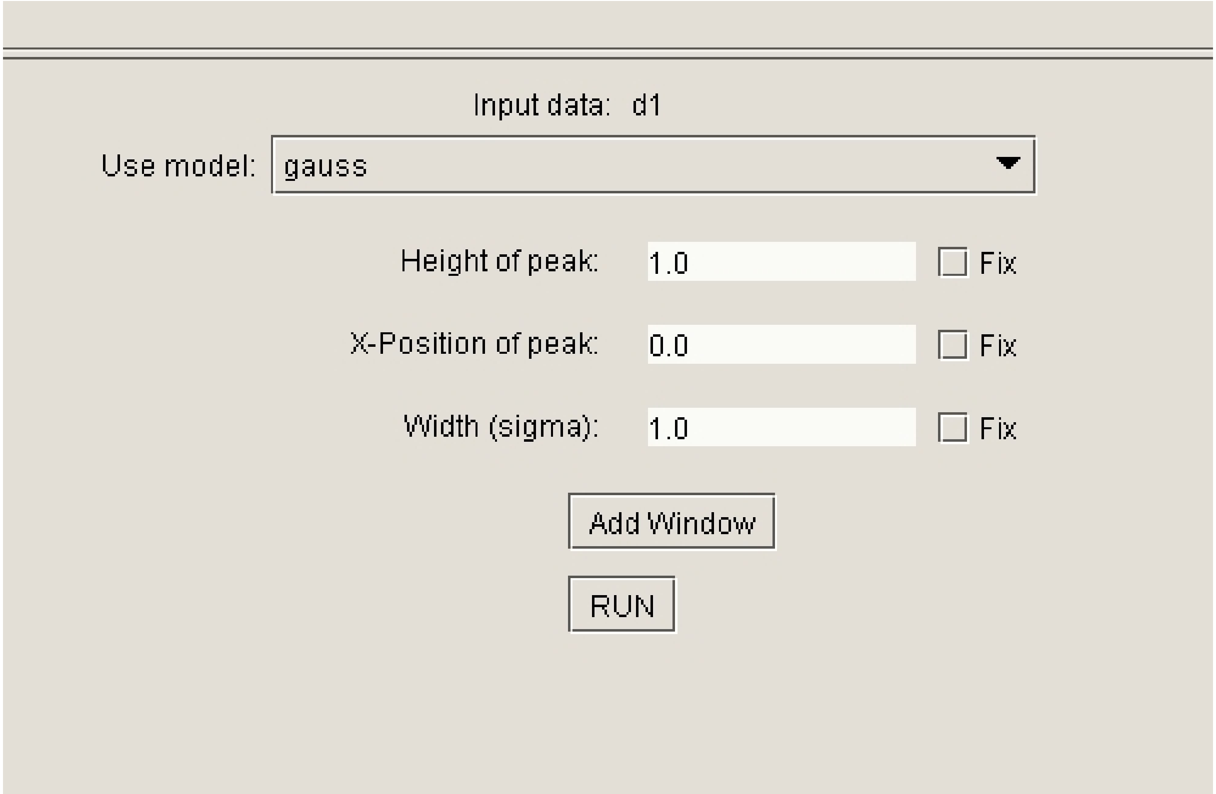


Figure 13.2. The Gaussian fit GUI. Supply the Gauss fit parameters.

A Lorentzian ("lorentz") model can be fitted in a similar way. If you select a polynomial ("poly") fit, then only the order of the polynomial needs be defined. The 'fix' tickboxes can be used to fix the value of the parameters (Figure 13.3).

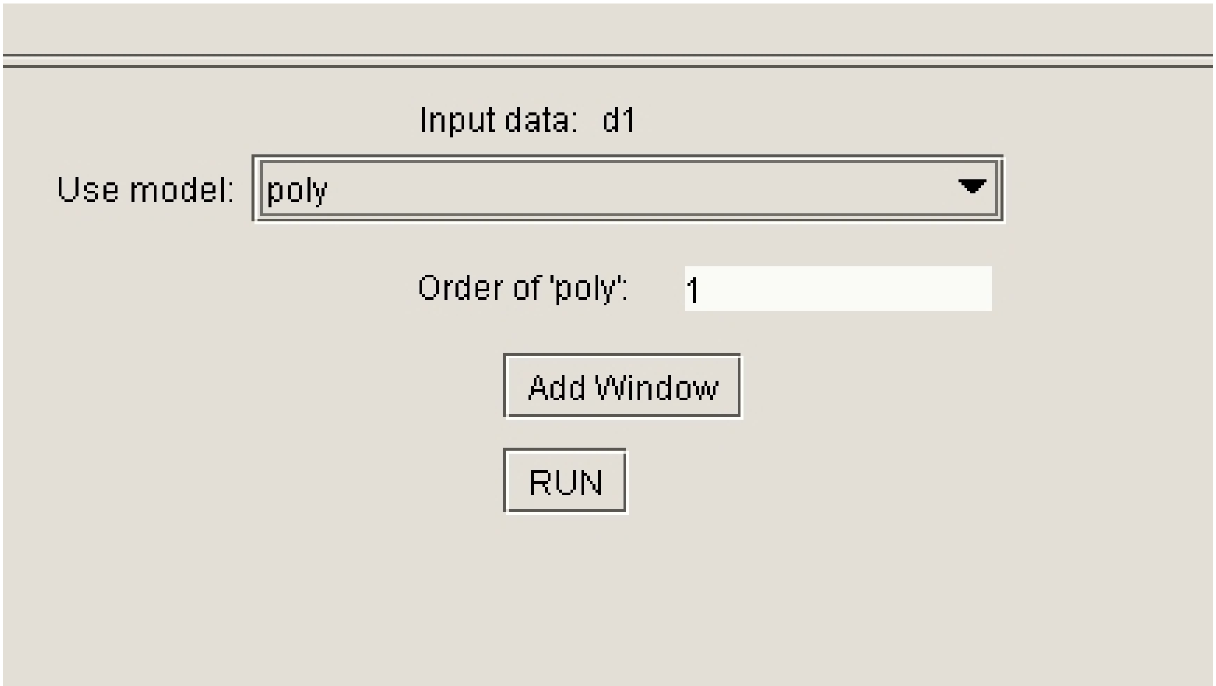


Figure 13.3. The polynomial fit GUI. Supply the order of the polynomial fit.

If any other model is selected, the GUI will look like Figure 13.4 . If the model requires constructor parameters (see the "DP Basic User's Manual" for details of all available models' parameters), the 'Constr Parms' field must be filled with a comma-separated list, for example, the "power" model needs a degree. If no constructor parameters are needed, as for the "sinc" model, leave the field empty. In the 'Parameters' field the fit parameters must be filled in with a comma-separated list - be sure to give the correct amount of parameters (see the JavaDoc). In the 'Fixed' field the parameters that must be fixed can be listed in (guess what) a comma-separated list. If the first and third parameter must be fixed, fill in: 0,2.

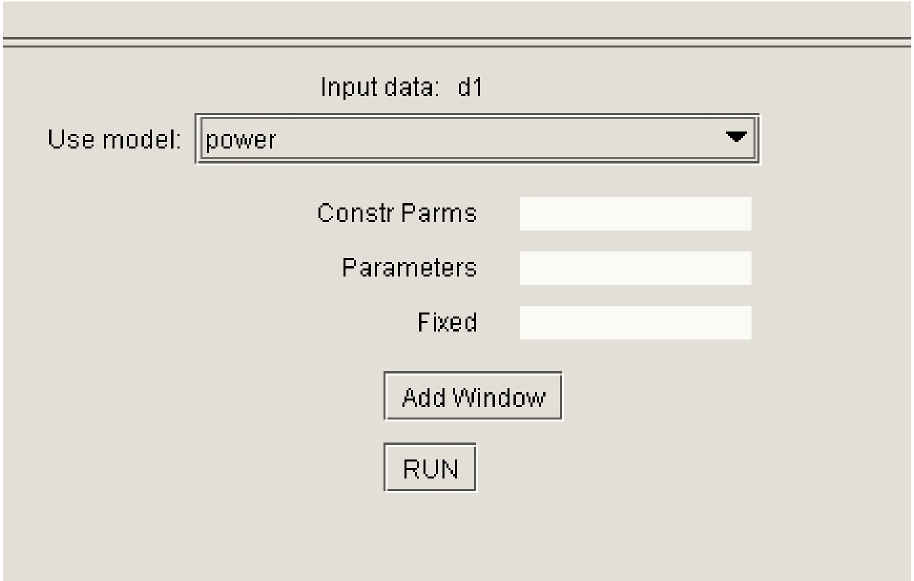


Figure 13.4. Other fit models. Supply all the model fit parameters.

3. A fit can be applied over a specified range.

   Click on 'Add Window' to define the fit X-range (between 'from' and 'to'). Up to five ranges can be defined by clicking on 'Add Window' again (Figure 13.5).

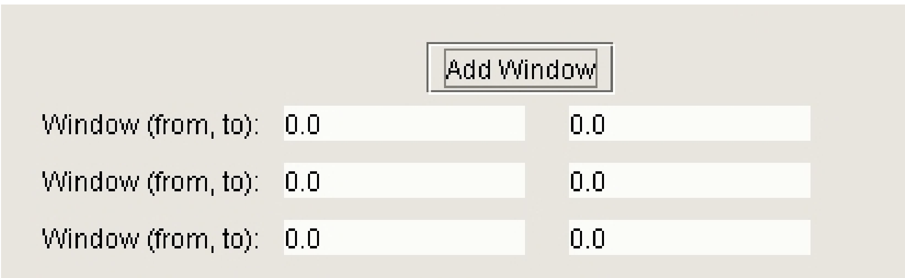


Figure 13.5. fitSpectrum can be applied over a specified range. Set the fit X-range

4. The result of a fit is a "fitResult" variable. Double clicking this variable opens a view window with the data and fitted model (top) and the residual (bottom). A table of the fitted parameters and their standard deviations can be seen by then clicking on the "ShowFitResult" task (Figure 13.6). If the fitResult contains several models, the parameters for all models are listed here.

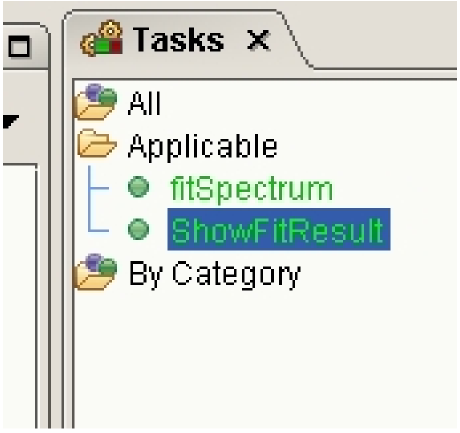


Figure 13.6. Tasks available after "fitSpectrum". Click on "ShowFitResult" to see fitted parameters and their standard deviations.

5. Another model can be applied to the result of a fit. So you can, for example, fit the baseline and then fit a spectral feature.

   Click on a 'fitResult' and then again on the 'fitSpectrum' task and follow the proceedure given above. This results in another fitResult variable to which you can apply another fit model, and so on.

6. Once satisfactory models for all spectral features have been found, all the models can be applied to the original data.

   Click on your final fitResult variable and then click again on the fitSpectrum task. The GUI contains a checkbox 'global fit' (Figure 13.7), check this and click on 'RUN'. No new model can be added at this stage.

Figure 13.7. Global fit. Check global fit to apply all models to original data.

13.2. How to fit spectra from the command line

1. Download the toolbox into the session, note that in JIDE it is called SpectrumFitter rather than fitSpectrum!

```
from herschel.ia.toolbox.spectrum.fit import SpectrumFitter
from herschel.ia.toolbox.spectrum.fit.testdata import MakeData
```

   For demonstration purposes, we will use MakeData to create some test data to fit.

```
data=MakeData(7)
data.addNoise(10)
#instantiate the fitter
sf=SpectrumFitter(data)
```

A plot window should look similar to that shown in Figure 13.8.

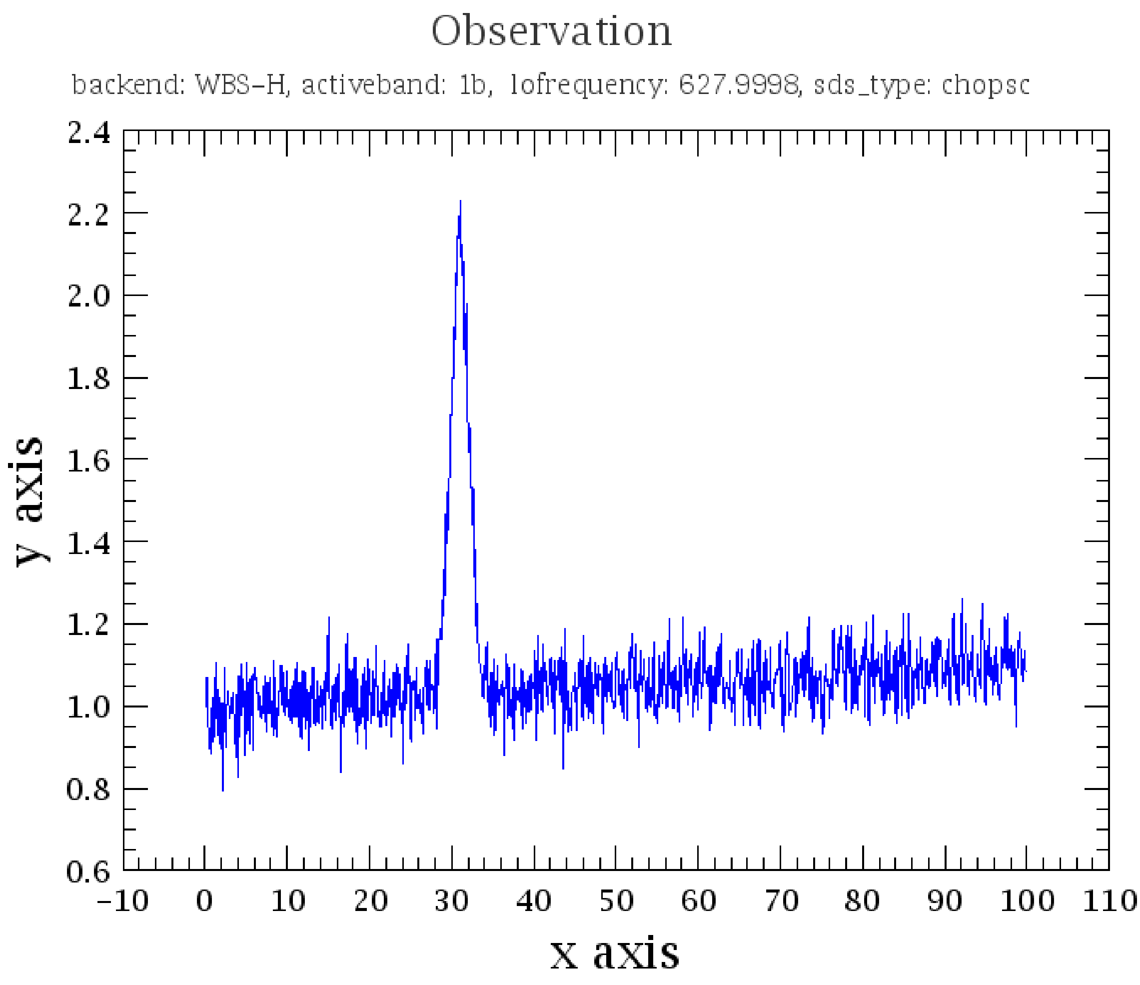


Figure 13.8. Test data to fit. Start the SpectrumFitter

2. The SpectrumFitter is an interactive tool and is best used in conjunction with the SpectrumModel tool, which allows you to select (and change) models and fitting parameters. The three models you are most likely to use are Gaussian, Lorentzian and Polynomial; the model fits, their parameters, and their usage in the SpectrumFitter tool are summarized in Table 13.1:

Table 13.1. Model fits, their parameters and usage in the SpectrumFitter tool

Model	Mathematical fit	Parameters	Usage
Gaussian	$f(x) = a_0 \, exp\left\{\frac{-(x-x_0)^2}{2s_0^2}\right\}$	a_0 = amplitude of line	**`sf.addModel('gauss', [a0,x0,s0])`**
		x_0 = location of line peak	
		s_0 = width of line (sigma)	
Lorentzian	$f(x) = p_0 \left[\frac{p_2}{(x-p_1)^2 + p_2^2}\right]$	p_0 = amplitude of line	**`sf.addModel('lorentz', [p0,p1,p2])`**
		p_1 = location of line peak	
		p_2 = half width at half maximum of line	
Polynomial	$f(x) = c_0 + c_1 x + ... + c_n x^n$	n = order of polynomial	**`sf.addModel('poly', [n], [c0,c1, ..., cn])`**
		c_0 .. c_n = polynomial coefficients	

Note that you must know (roughly) where you expect a spectral feature in your data to be, in addition to its expected shape and approximate shape parameters. So, an initial guess is required - if this guess is completely wrong you may end-up fitting noise rather than your spectral lines.

Now, fit first the baseline with a polynomial and then fit the line with a Gaussian.

```
#First the baseline
# Apply the model
model=sf.addModel('poly', [2],[0,0,0])
# Do the fit
sf.doFit()
# Inspect the residual after the baseline is removed
sf.residual()
# Keep the fit
sf.fitOK()
#Now the line
sf.addModel('gauss', [1.0,30,0.1])
sf.doFit()
sf.residual()
sf.fitOK()
```

These steps result in the plot below. A black line (not seen here) displays the model and is replaced by a green line showing the fit (the Gaussian model here). The red line is the final fit for the entire spectrum. The residual is shown in a separate plot.

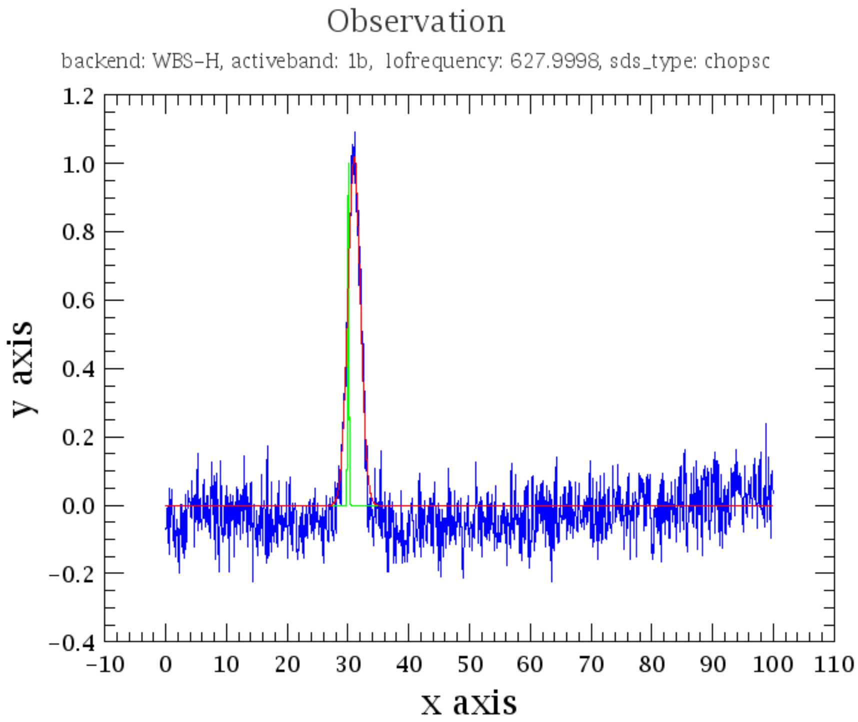


Figure 13.9. Fit result. Fit results for spectrum

3. It is possible to do both fits at the same time, globally, since the instance of our SpectrumFitter remembers what it has done so far.

```
sf.doGlobalFit()
```

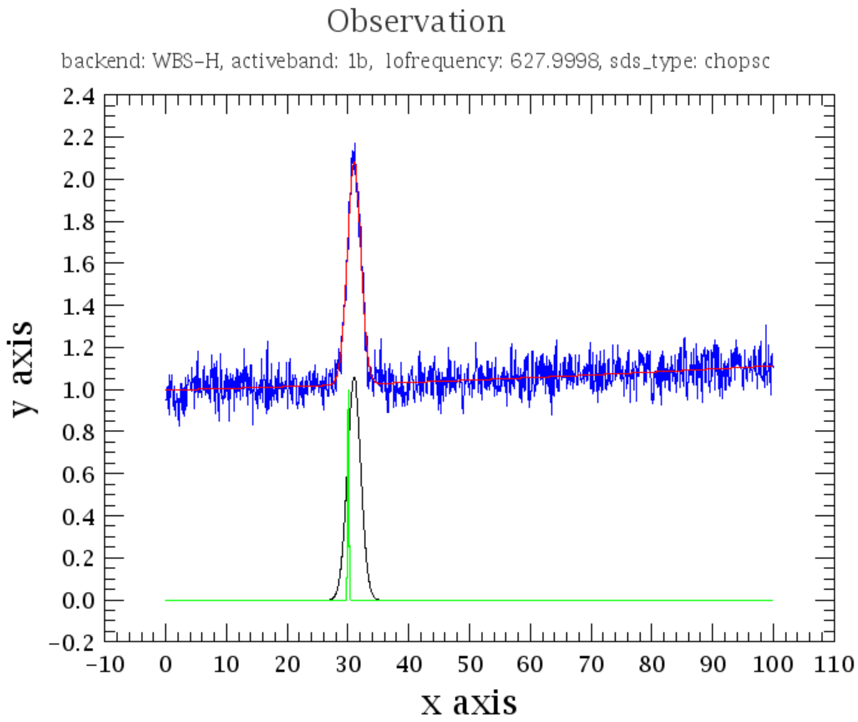


Figure 13.10. Global fit. Use the models together in a global fit

4. It is also possible to mask data. The following will do a polynomial fit only using data from 0 to 20 and from 40 to 100.

```
model=sf.addModel('poly', [2],[0,0,0])
#after you've created the model, now add the masks.
model.setMask(0,20)
model.setMask(40,100)
```

   To best see how this works, include this masking in the example given above.

5. The fitted model parameters and their standard deviations are printed to screen with:

   `print sf`

6. It is possible to manipulate the models produced by SpectrumFitter in various ways:

   - If you wish to change the initial parameters of any of the models (`model = sf.addModel(...)`), use setParameters:

     `model.setParameters([...])`

     A new fit will be made on the fly.

   - There are two ways to remove models:

     `sf.removeModel(m)`

     Or:

     `m.remove()`

- Subtract the model from the dataset:

  ```
  sf.subtractModel(m)
  ```

  This also removes the model from the fitter tool.

- Once you are satisfied with a fit, you can set the fitted parameters as the default for the models:

  ```
  m.useResults()
  ```

  This may be useful when using the same models for a following dataset.

- To apply them to a different dataset:

  ```
  sf.setData(otherData)
  ```

  Note that this replaces the data held in the SpectrumFitter with the SpectralSegment held in the variable 'otherData'. Once again, the fit will be redone on the fly.

Chapter 14. HowTo Display and Manipulate Images in HIPE

Herschel Editorial Board

14.1. Introduction

All image display tasks work on a `SimpleImage` that can be derived from a FITS file import (see HowTo chapter on FITS and ASCII input/output) or even from an image file such as a JPEG -- which is what we will use for illustrative purposes in this chapter.

Images can come with associated flux information (in header or, in Herschel DP, meta data). The flux information/units can also be applied to a given image by hand.

Images either have a World Coordinate System (WCS) stored in the meta data information, or a WCS may be applied. This chapter will also include information on the WCS parameters that can be found or applied to a given image.

Throughout this chapter illustrations are given from the *Full Work Bench* perspective.

14.2. Creation of a `SimpleImage` for Display

The `SimpleImage` format data is the standard map/ image data format that comes from the pipelining of Herschel data following standard pipeline processing. Images downloaded from the Herschel Science Archive are in this format. The following short script can be adapted to create a `SimpleImage` from any JPEG file and associate a very simple WCS to it. The following can be copied and pasted into the Editor view after opening a Jython script window, or copied into the Console view and run from there.

```
# Create some fake WCS information
myWcs = Wcs(crpix1 = 29, crpix2 = 29, crval1 = 30.0, crval2 = -22.5, \
 cdelt1 = 0.00028, cdelt2 = 0.00028, ctype1="RA---TAN", ctype2 = "DEC--TAN")
# Create a SimpleImage with WCS in it
myImage2 = SimpleImage(wcs = myWcs)
#Put the image into the SimpleImage
# *.jpeg, *.jpg, *.tiff, *.tif, *.png, *.fits, *.fts or *.fit
# files are accepted.
importImage(image = myImage2, filename="directory name/ngc6992.jpg")
```

A SimpleImage called "myImage2" is created and is available in the "Variables" view (See Figure 14.1). It should be emphasised that it is possible to use ANY image created in an instrument pipeline for the examples given in this chapter.

The importing of the image is also possible via the "importImage" task available in the Tasks view list. Click on "myImage2" in the "Variables" view then double-click on the appropriate task, "importImage". A name can be typed in or a selection made by Browse...ing the system.

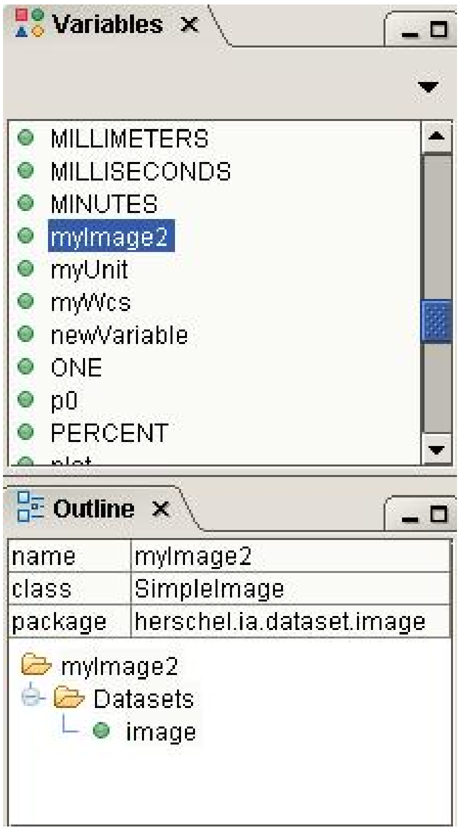


Figure 14.1. The Variables view shows the "myImage2" highlighted. A double click on this automatically brings up the image in a new Editor window (top left). In the Tasks view the folder "Applicable", when opened, shows the tasks that can be applied to this image.

Double-clicking on the variable "myImage2" in the Variables view will automatically display the image in a new Editor window. A single right click in the same place will indicate that this can be "Open(ed) with..." a Product viewer as well. This shows header (metadata) information for the whole image product, which can have a number of datasets. For the `SimpleImage`we have created for our example there is a single image dataset.

14.3. Viewing the Metadata and Array Data Associated with an Image Dataset

An image can have several datasets. For example, we can include a flag image dataset for flagging bad pixels (see "DP Basic User's Manual" for more information). Each of these datasets have associated metadata, which has the same role as header information in a FITS file. It indicates associated flux and coordinate information plus processing history (if appropriate) etc.

To view the metadata (and array data) associated with an image dataset requires opening a Dataset viewer. This can be done in two ways.

- First a right-click on your image variable name in the "Variables" view (e.g., on "myImage2"). A short menu including "Open With...." appears. Choose the product viewer. The product view is shown which includes some overview information/metadata plus a list of datasets (at the bottom of the datasets -- and could include a number of image layers). Do a right-click on one of the datasets to see the "Open With..." in the short menu. Select Dataset viewer.
- A single click selection of the image in the "Variables" menu list shows its outline in the "Outline" view. Opening the folder in the Outline view to see the datasets in it and right-click on a dataset to see the short menu with "Open With...." and the dataset viewer selectable.

Any of the above will provide a view of the metadata plus the data values of the array making up the dataset within a window in the "Editor" view. View of either the metadata or array data can be toggled using the arrows to the left of the metadata/array data names in the "Editor" window (see Figure 14.2).

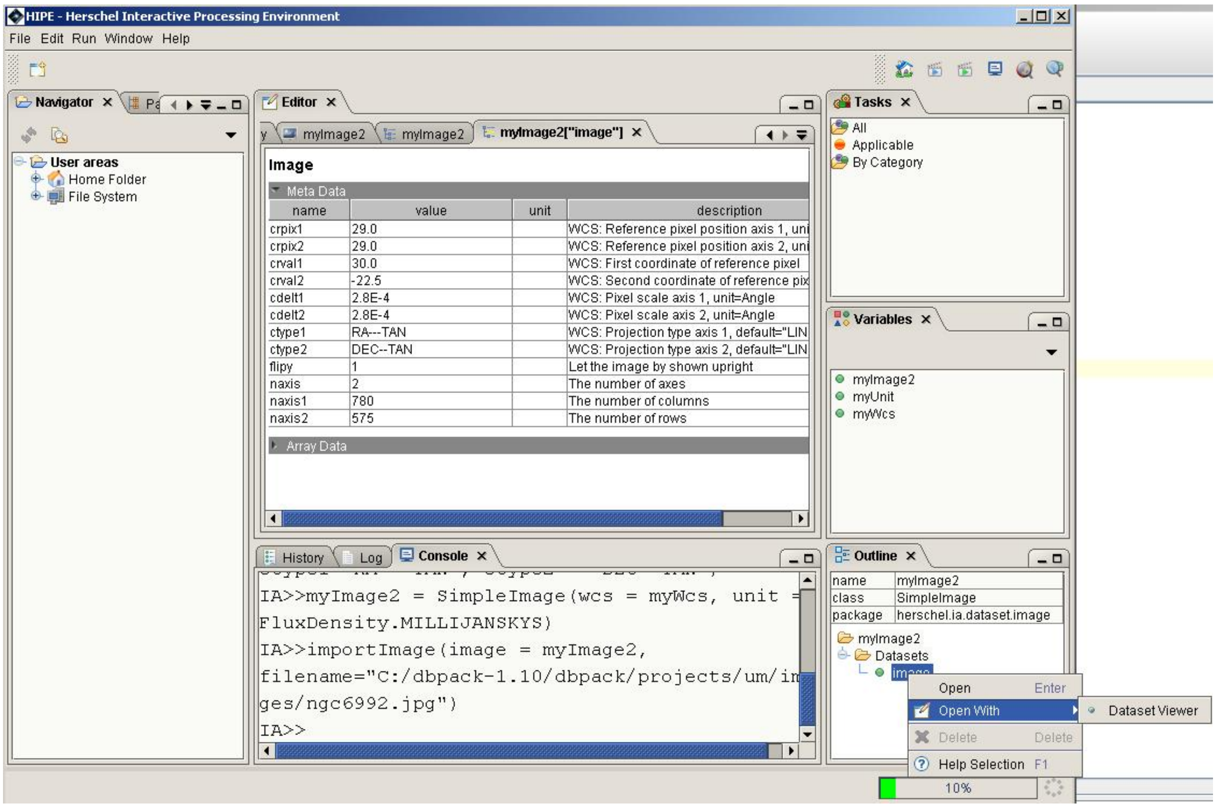


Figure 14.2. Metadata and Array data view using the Dataset viewer with an image.

14.4. A Simple Display of an Image

The simplest way to display an image in HIPE is to double-click the image name (e.g., in the "Variables" list). The default activity for this is then the display of the image in a new window in the "Editor" view (see Figure 14.3).

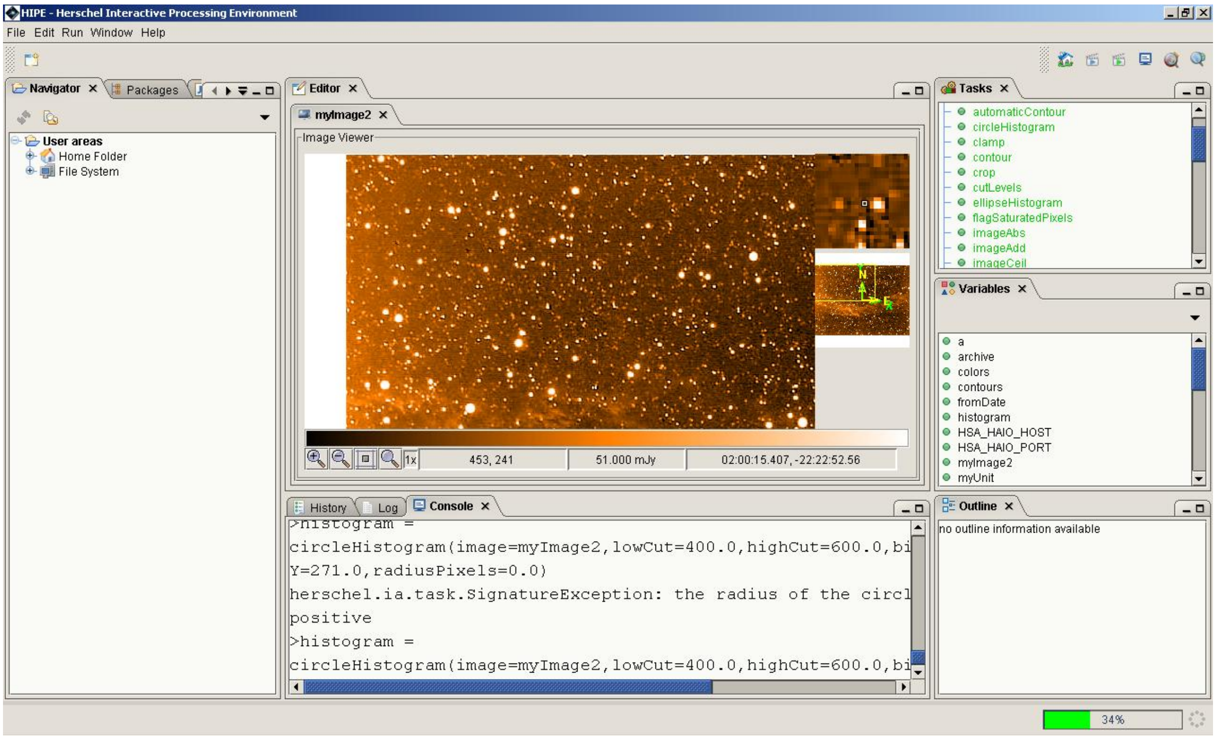


Figure 14.3. Automatic Display obtained via double-click of a SimpleImage variable appearing in the "Variables" view.

The display shows a zoom/pan image as the main image, plus two smaller images that show,

- A zoomed image is shown around the mouse position on the main image (at top right).
- An overview of the full image showing the zoom/panned position of the main display (at bottom right) outlined by a rectangle. This box also illustrates the directions N and E on the display based on the WCS coordiantes of the image. The position of the rectangular zoom/pan region can be adjusted by clicking on the box and dragging it to another part of the display. In Figure 14.3 the box has been dragged to the top left of the image.

14.4.1. Magnifying an Image

To bottom left of the view (see Figure 14.3) are a set of magnifying glass images that, in order left to right, zoom in, zoom out and go back to the original image size. In between the magnifying glass images is an icon with a small square surrounded by a box -- which allows an image to be displayed that fits the whole SimpleImage into the viewing area.

14.4.2. Image Coordinates and Pixel Intensity

The mouse position over the image is constantly updated at the bottom of the image displayed with both the pixel coordinates and the world coordinates (if a WCS is available in the SimpleImage being viewed) being presented to the right of the magnification icons.

In between the two pieces of coordinate information the pixel intensity for the pixel falling under the mouse position is also constantly updated (see again Figure 14.3) as the mouse is moved across the image.

14.5. Editing and Printing Images

We can edit an images in a number of ways. The following are available after doing a right-click of the mouse button while the mouse is over a displayed image (see Figure 14.4).

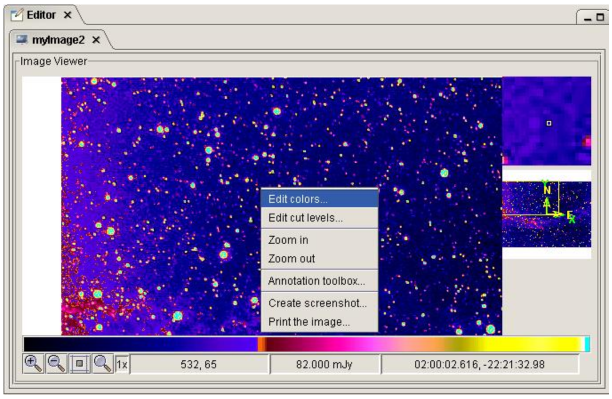


Figure 14.4. Edit functions available via a right-button mouse.

- Edit colors -- the colour lookup table can be adjusted to a number of different types plus linear/ log/exponential scalings.
- Edit cut levels -- the cut levels for which the colour lookup table is to be applied can be adjusted for an image.
- Zoom in/out -- as per the zoom icons discussed earlier.
- Annotate the image -- an annotation can be placed at the position of a mouse click.
- Create screenshot/print image -- the displayed image can be saved to an image file or printed on a user-selected printer.

14.5.1. Editing the Colour Look Up Table (LUT)

The standard colour scheme for image display is for "Real" colours shown in a "Ramp" intensity with a "linearScale". Selection of the "Edit colors..." from Figure 14.4 displays the colour menu Figure 14.5. Hitting the "Reset" button always enables the default colour display.

To select any other colour scheme simply click on the colour type and/or intensity or algorithm to create a new colour scheme. The scheme is applied to the image immediately. The window (Figure 14.5) can be dragged away from the image.

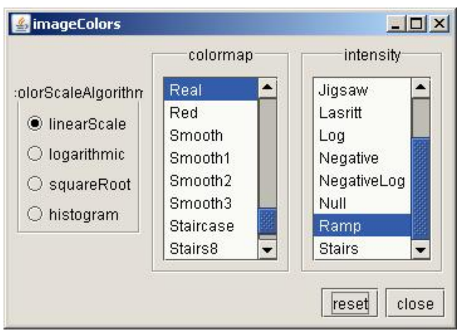


Figure 14.5. Colour table selection menu. Hitting "Reset" takes you back to the original colour table.

14.5.2. Editing the Cut Levels

The default cut levels for images is 99.5 per cent of pixel values. Selection of the "Edit cut levels..." from Figure 14.4 displays the a cut level selection including a histogram of the current pixel intensity values Figure 14.6. Hitting the "Reset" button always enables the default cut levels.

To select any other cut level the user can do one of two things.

- A button selection of cut levels (90, 95, 98, 99, 99.5 or 100 per cent). Note that selection of any of these will adjust the histogram display above.
- Adjustment of the upper and/or lower-level cutoffs of the histogram by click-and-drag of the yellow arrows (left or right) shown at either end of the histogram view.

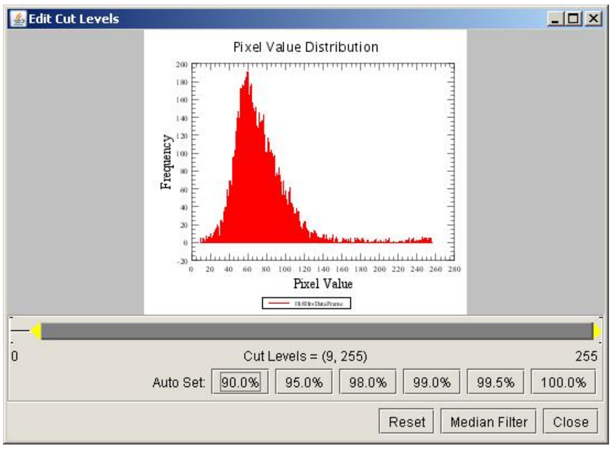


Figure 14.6. Cut level selection window. Hitting "Reset" takes you back to the original cut levels of 99.5 per cent.

14.5.3. Zoom In/Out

Selection of either of these provides an increase or decrease in zoom by a factor of 2.

14.5.4. Annotation Toolbox

The annotation toolbox is shown in Figure 14.7.

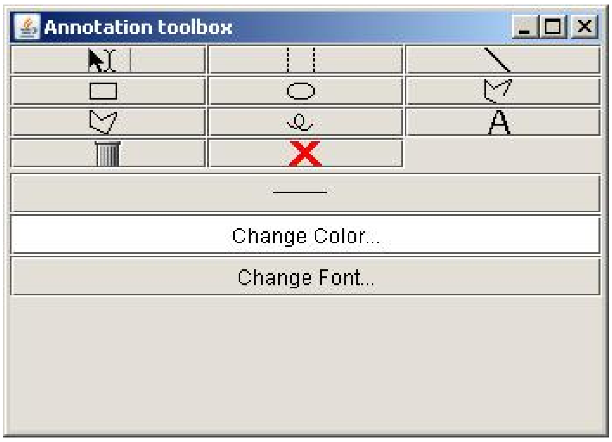


Figure 14.7. The annotation toolbox.

The icons in the annotation toolbox appearing in Figure 14.7 have the following usage (from left to right and from top to bottom):

- Select annotation
- Select all annotations in a region
- Draw a line
- Draw a rectangle

- Draw an ellipse
- Draw a polyline
- Draw a polygon
- Draw with the free hand on the image
- Add a text annotation
- Remove the selected annotation(s)
- Remove all annotations

Letting the mouse linger over an icon also displays its function.

The polygon and polyline methods will enable you to select points on the image which should be used as a corner of the polygon using the mouse. Double-clicking the mouse will end the selection procedure.

The three buttons below the ones already described change the view of the annotation. From top to bottom:

- Change the thickness of the line
- Change the colour of the annotation. The present colour of annotations is shown in the background.
- Change the font of the text annotation

14.5.5. Screenshots and Printing Images

The last 2 possibilities within the image edit menu allows screenshots to be created in JPG, PNG or BMP format or a printing to a user-selected printer. The user is also given the choice of whether the image produced includes all overlays and annotations or not.

14.6. Image Transformations

Image representations can be adjusted in the following ways:

- Clamp: or clipping an image.
- Crop: extract a seubsection of an image.
- Clamp: or clipping an image.
- Rotate: rotate image by an arbitrary angle
- Scale: image rescaling in user-selected factors in X and Y.
- Translate: move positive or negative pixel or sky amount of image within the frame.
- Transpose: flip or rotate by n x 90 degrees

14.6.1. Applying Image Transformations

All image transformations can be applied in the same way. First, select a `SimpleImage` in the "Variables" view, then go to the "Tasks" view and select -- from the Applicable Tasks folder -- the appropriate image transformation (*crop*, *clamp*, *rotate*, *scale*, *translate* or *transpose*). To select one of the transformation tasks, double-click on its name on the Tasks view. This will bring up a dialog for the task.

Dialogs work in a similar fashion for all image transformations. Options are presented in a pull-down menu (e.g., the form of the interpolation of pixel values when rotating an image, see Figure 14.8) or with an editable input such as the rotation in degrees and the option for the name of the output variable created following the transformation. Hitting "Accept" will run the task.

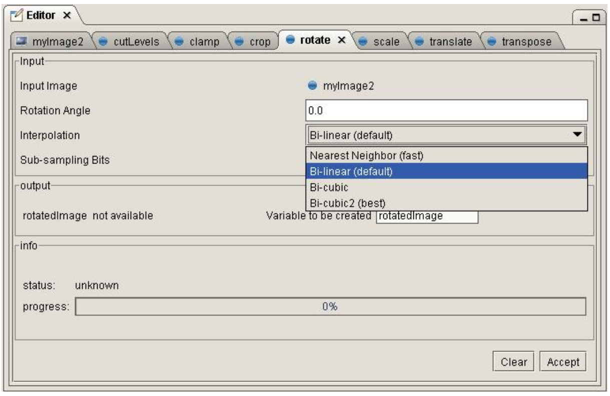


Figure 14.8. Example image transformation dialog. Rotating an image using the "rotate" task. Several interpolation options are available.

14.6.2. Image Transformation Options

For each image transformation there are a set of options for the user.

14.6.2.1. Clamp Options

This allows the floor and ceiling of an image to be set. Values above the max or below the min input by the user are set to the max and min values assigned by the user respectively.

14.6.2.2. Crop Options

A section of the image to be extracted to another SimpleImage. The range of X and Y pixel coordinate values are input by the user.

14.6.2.3. Rotation Options

When rotating the image, several types of interpolation are possible. By default, bi-linear interpolation is used. There are four types of interpolation possible.

- **Bi-linear [default]** -- *the default* - interpolates one pixel to the right and one below.
- **Nearest neighbour [fast]** -- direct pixel copying, the fastest.
- **Bi-cubic** -- uses interpolation via a piecewise bi-cubic polynomial.
- **Bi-cubic2 [slow]** -- variant of bicubic interpolation that can give sharper results than bicubic.

14.6.2.4. Scale Options

Allows for different magnification in X and Y pixel directions. Possible interpolation types are as for the "rotate" task.

14.6.2.5. Translate Options

Allows either X and Y pixel translations or sky translations (coordinates input as strings of the form "hh:mm:ss.s" and "dd:mm:ss.s") can be input by the user.

14.6.2.6. Transpose Options

Allows for different simple transpositions of images. The following transpositions can be done with this task.

- Flip vertical (flips top and bottom)
- Flip horizontal (flips from side to side)
- Flip diagonal (bottom left to top right)
- Flip antidiagonal (top left to bottom right)
- Rotate 90 degrees (clockwise rotation)
- Rotate 180 degrees
- Rotate 270 degrees

14.7. Image Arithmetic

Images can be arithmetically manipulated (scalar or pair-wise combinations) to provide changed versions of the original. In all cases, image arithmetic can be done by opening a dialog, filling in the dialog and then clicking "Accept" to run the task (e.g., Figure 14.9).

Possible arithmetic tasks are:

- Absolute value (imageAbs). To obtain the absolute value image from the input.
- Add/Divide/Multiply/Subtract (imageAdd, imageDivide, imageMultiply, imageSubtract). This allows either a scalar or a second image as the amount to be added/divided/multiplied/subtracted. The second image can be input into the dialog by click-and-dragging of it from the "Variables" view to the orange dot position in the dialog for the second image (see Figure 14.9) For images, the combination is by pixels or WCS reference.
- Exponent of the image. Including to the power N and 10 (imageExp, imageExpN, imageExp10).
- Log of the image. Including base 10 or N (imageLog, imageLog10, imageLogN).
- Image to the power n (imagePower).
- Image rounding (imageRound).
- Square and square root of the image (imageSquare, imageSqrt).

Most of the above are self-explanatory. One example is shown in Figure 14.9.

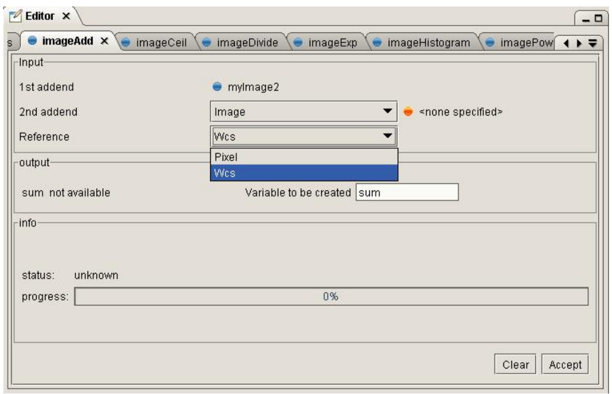


Figure 14.9. Example image arithmetic dialog.

14.8. Working with the World Coordinates System (WCS)

The WCS information for an image is stored in its metadata which can be viewed using the Dataset viewer.

The Wcs class enables the user to define a transformation between the pixel coordinates and world coordinates. The following illustrates how we can type in (at a command-line) a WCS. We create our Wcs() object which we then add to a fake `SimpleImage` we set up to start with.

```
i = SimpleImage()
i.image=RESHAPE(Double1d.range(200*300), [200,300])
# create a fake image 200x300 pixels in size

myWcs = Wcs() # set up the Wcs() object
myWcs.ctype1 = "LINEAR"  # start adding things to it....
myWcs.cdelt1 = 5
myWcs.crval1 = 200
myWcs.cunit1 = "K"
myWcs.crpix1 = 0

myWcs.ctype2 = "LINEAR"
myWcs.cdelt2 = .05
myWcs.crval2 = 2.0
myWcs.cunit2 = "V"
myWcs.crpix2 = 0

i.wcs = myWcs  # apply the set of WCS information to our image
print i.wcs #to see the WCS of the image
```

The above example will create a coordinate system, where the temperature and current are set for the axes. The x-axis is LINEAR (ctype1), has the central pixel in column 0 (crpix1), has a value of 200 in the central pixel (crval1), uses steps of 5 (cdelt1) and has as unit Kelvin. The y-axis is also LINEAR (ctype2), has the central pixel in row 0 (crpix2, this is the top of the image), has a value of 2 in the central pixel (crval2), uses steps of 0.05 (cdelt2) and has as unit Volts.

It is also possible to use the Wcs class to define transformations between pixel coordinates and sky coordinates. This can be done using the standard Wcs parameters. An example is given below. It also indicates how we can "set" WCS values in our WCS object :

```
wcs2 = Wcs()  # ❶
wcs2.setCrpix1(128)
wcs2.setCrpix2(128) # ❷
wcs2.setCrval1(101.676612741936)
wcs2.setCrval2(0.829427624677429) # ❸
wcs2.setCtype1("RA---TAN")
wcs2.setCtype2("DEC--TAN") # ❹
wcs2.setRadesys("ICRS")
wcs2.setEquinox(2000.0) # ❺
wcs2.setParameter("cd1_1", -1.9064468150235E-6, "")
wcs2.setParameter("cd1_2", 3.39797311269006E-4, "")
wcs2.setParameter("cd2_1", 3.39811958581193E-4, "")
wcs2.setParameter("cd2_2", 1.580446989748E-6, "") #❻
```

❶ A `Wcs` is created.

❷ The central pixel is set. In this case, the central pixel is at (128, 128).

❸ The value of the central pixel is set. In this case, the first central pixel is located at 6h46'42.387" and the second pixel at 0 degrees 49'45.94".

❹ The type of the axes is set. The first axis defines the right ascension (in a gnomonic projection) and the second axis defines the declination (in a gnomonic projection).

❺ The coordinate system is set (here, we use the standard ICRS type). The equinox is also set.
❻ The linear transformation matrix is set. This defines the pixel size and the rotation of the images.

For more information on the WCS see Chapter 4 of the "DP Basic User's Manual."

Chapter 15. HowTo Do Basic Image Analysis in HIPE

Herschel Editorial Board

15.1. Introduction to Interactive Image Analysis with HIPE

Basic image analysis described in this chapter involves the following tasks that are available within the HIPE environment.

- aperture photometry
- image/area histograms
- 1D profile plotting
- contour plotting and overlays

All tasks work on a `SimpleImage` that can be derived from a FITS file import (see HowTo chapter on FITS and ASCII input/output) or even from an image file such as a JPEG -- which is what we will use for illustrative purposes in this chapter.

15.2. Setup and Display of Images for Analysis

In the chapter on Image Display we note how to create image coordinate systems and how to formulate the `SimpleImage` format from external sources. `SimpleImage` format data is the standard map/ image data format that comes from the pipelining of Herschel data during standard pipeline processing. Images from the Herschel Science Archive (HSA) are in this format. The following short script can be adapted to create a `SimpleImage` from any JPEG file and associate a WCS to it. The following can be copied and pasted into the Editor view after opening a Jython script window, or copied into the Console view and run from there.

```
# Create some fake WCS information
myWcs = Wcs(crpix1 = 29, crpix2 = 29, crval1 = 30.0, crval2 = -22.5, \
 cdelt1 = 0.00028, cdelt2 = 0.00028, ctype1="RA---TAN", ctype2 = "DEC--TAN")
# Create a SimpleImage with WCS in it
myImage2 = SimpleImage(wcs = myWcs)
#Put the image into the SimpleImage
# *.jpeg, *.jpg, *.tiff, *.tif, *.png, *.fits, *.fts or *.fit
# files are accepted.
importImage(image = myImage2, filename="directory name/ngc6992.jpg")
```

A `SimpleImage` called "myImage2" is created and is available in the "Variables" view (See Figure 15.1). It should be emphasised that it is possible to use ANY image created in an instrument pipeline for the following tasks.

The importing of the image is also possible via the "importImage" task available in the Tasks view list. Click on "myImage2" in the "Variables" view then double-click on the appropriate task, "importImage". A name can be typed in or a selection made by "Browse..."ing the system.

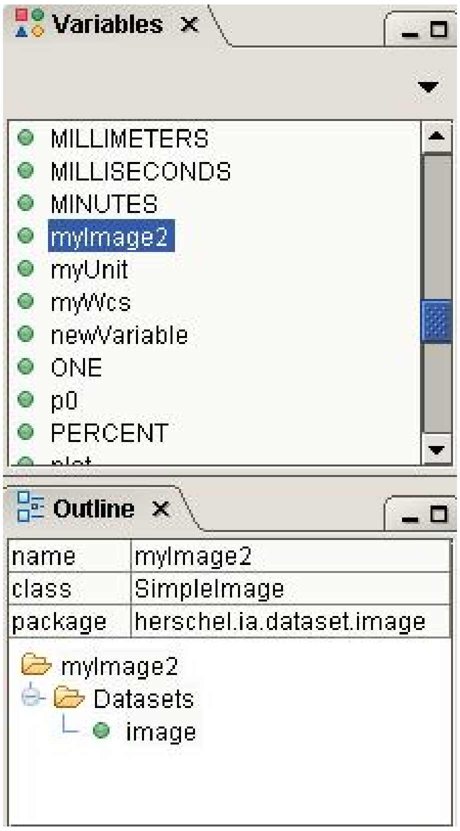


Figure 15.1. The Variables view shows the "myImage2" highlighted. A double click on this automatically brings up the image in a new Editor window (top left). In the Tasks view the folder "Applicable", when opened, shows the tasks that can be applied to this image.Variables view with `SimpleImage` variable highlighted.

Double-clicking on the variable "myImage2" in the "Variables" view will automatically display the image in a new Editor window. A single right click in the same place will indicate that this can be "Open(ed) with..." a Product display as well. This shows header information and the fact that there is a single image dataset in the `SimpleImage` product we have created.

The image appearing in the Editor view is displayed with the standard zoom/pan and editing capabilities associated with it that are discussed in the chapter "HowTo Create, Display and Manipulate Images."

15.3. Getting a `SimpleImage` a product out of the Herschel Science Archive (HSA)

When downloading a product out of the science archive we access images from an `ObservationContext`. An `ObservationContext` contains all the information associated with a single observation and its processing (including all associated calibration files). In a download (see chapter on HowTo Access Data) from the HSA we have products made available from several levels of processing at using the Herschel Science Center's Standard Product Generation pipelines.

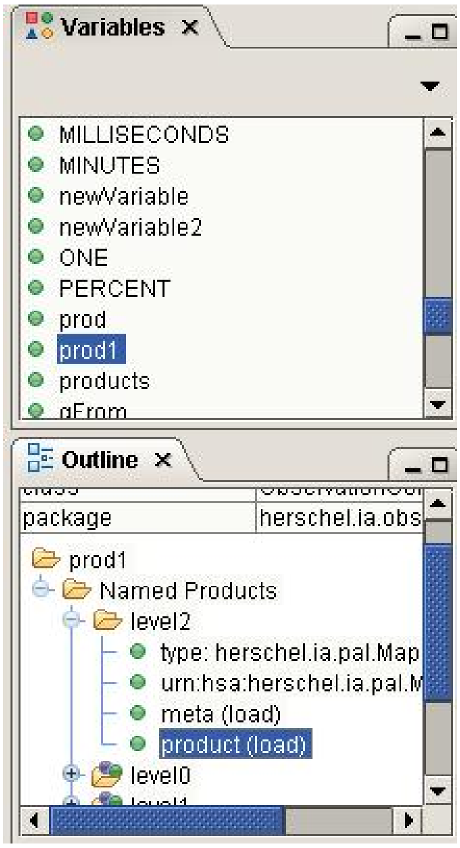


Figure 15.2. An `ObservationContext` called "prod1" has been obtained from the HSA. Clicking on the folders it contains in the "Outline" window allows us to get at the Level 2 product -- the final pipeline output for this observation.Contents of an `ObservationContext`

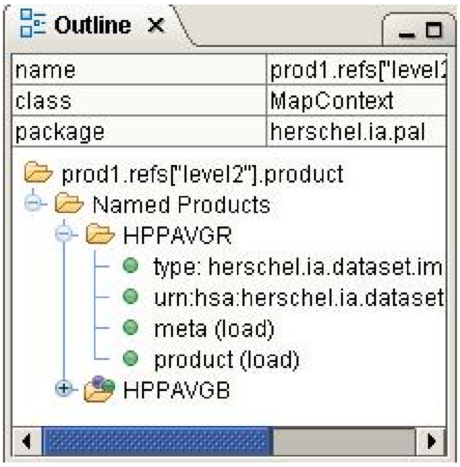


Figure 15.3. A double-click on the product highlighted in blue in Figure 15.2 provides this Outline view. A double-click on the product highlighted displays the image from the green channel of this PACS observation.PACS green channel image access

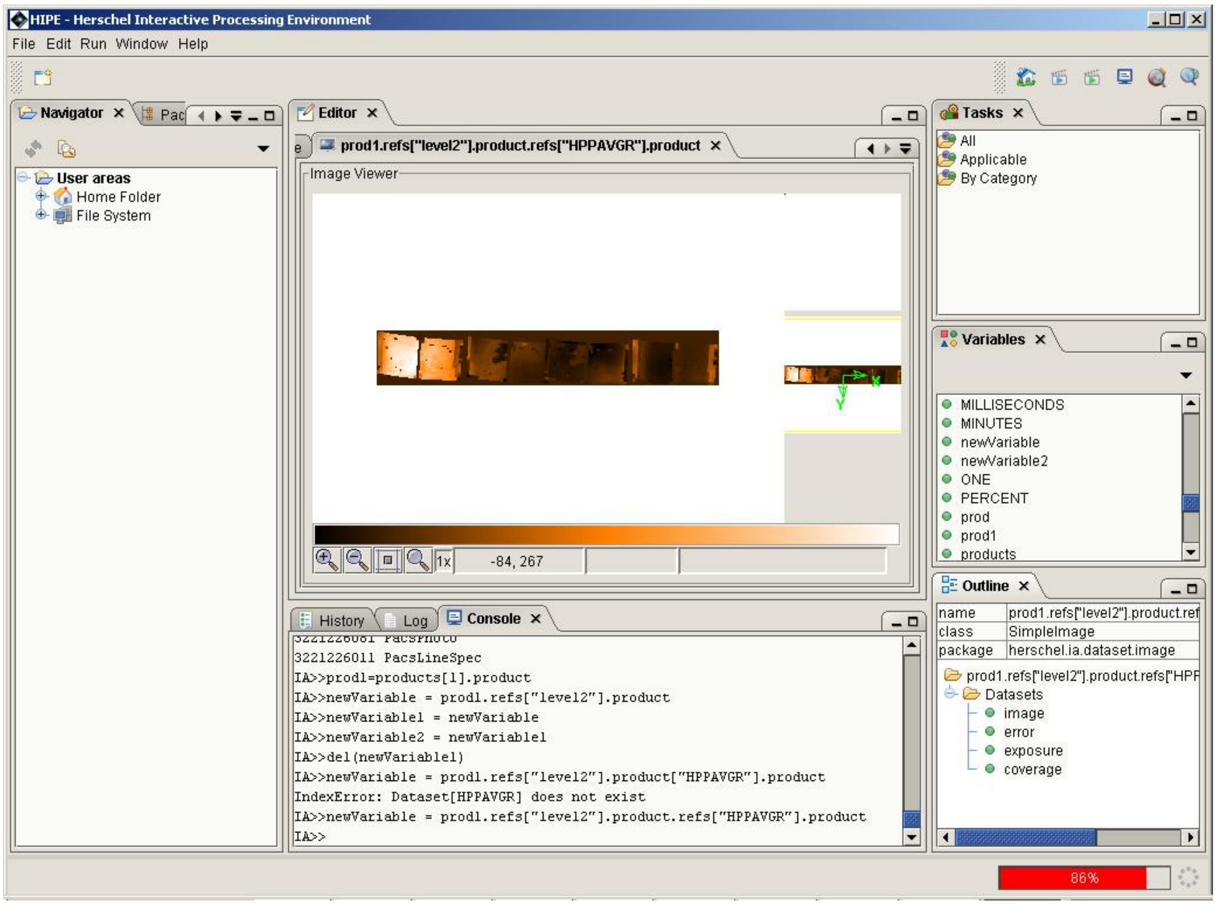


Figure 15.4. The PACS green channel image displayed in the full work bench of HIPE.

In the "Variables" and "Outline" displayed in Figure 15.2 and Figure 15.3 we see first an `ObservationContext` called "prod1" which is a PACS photometer test observation -- which has been expanded in the "Outline" view. A double click on the "Level2" product will show the outline of the final processed image (which contains two PACS images in two channels of the photometer taken simultaneously, a green channel and a blue channel). This is shown in Figure 15.3. We can also get the `SimpleImage` (e.g., name it "image1") by extracting it from the `ObservationContext`. The line below can do this from the command-line of the "Console" view.

```
image1 = prod1.refs["level2"].product.refs["HPPAVGR"].product
```

A double click on the product automatically opens up an image display of the test image. In the "Outline" window we can actually see that there are several datasets which include an error map, a coverage map and exposure map associated with the image (see Figure 15.4). A right click on any of the associated datasets and going to "Open With..." allows a Dataset viewer to appear which shows metadata and array data for the particular dataset.

15.4. Basic Analysis Capabilities

It should be noted that the overview and zoomed images displayed to the right of the displayed image during basic image analysis are the reverse for those when just dislaying the image, as illustrated in the "HowTo Display and Manipulate Images" chapter.

The basic analysis capabilities described in this chapter -- for application to `SimpleImage`s are;

- *1D profile plotting. Slices can be taken through the image*

- making a *histogram* of the whole image or of a certain region of interest, which is bounded by a circle, an ellipse, a rectangle or a polygon (the user should draw the bounding figure on the image)
- *aperture photometry* with a circular target aperture and an annular or a rectangular sky aperture
- *contour plotting and overlays*

Note

Note that all these functionalities are also available via the command line in the HIPE "Console" view. Using GUI/dialog interaction will copy the equivalent command to the "Console" view. This can be copied and pasted into a script (if wanted) for possible use in further, batch, processing.

15.4.1. 1D Profile Plotting

The 1D profile plotting capability allows the user to draw a straight line on an image and plot the intensity along that straight line.

After double-clicking on the Variable "myImage2" in the previous section the image was displayed in an "Editor" window. The Applicable Tasks are also available including the task `profile` (see Figure 15.5). A double-click on this item in the tasks list brings up another display of the image and allows interaction with the mouse

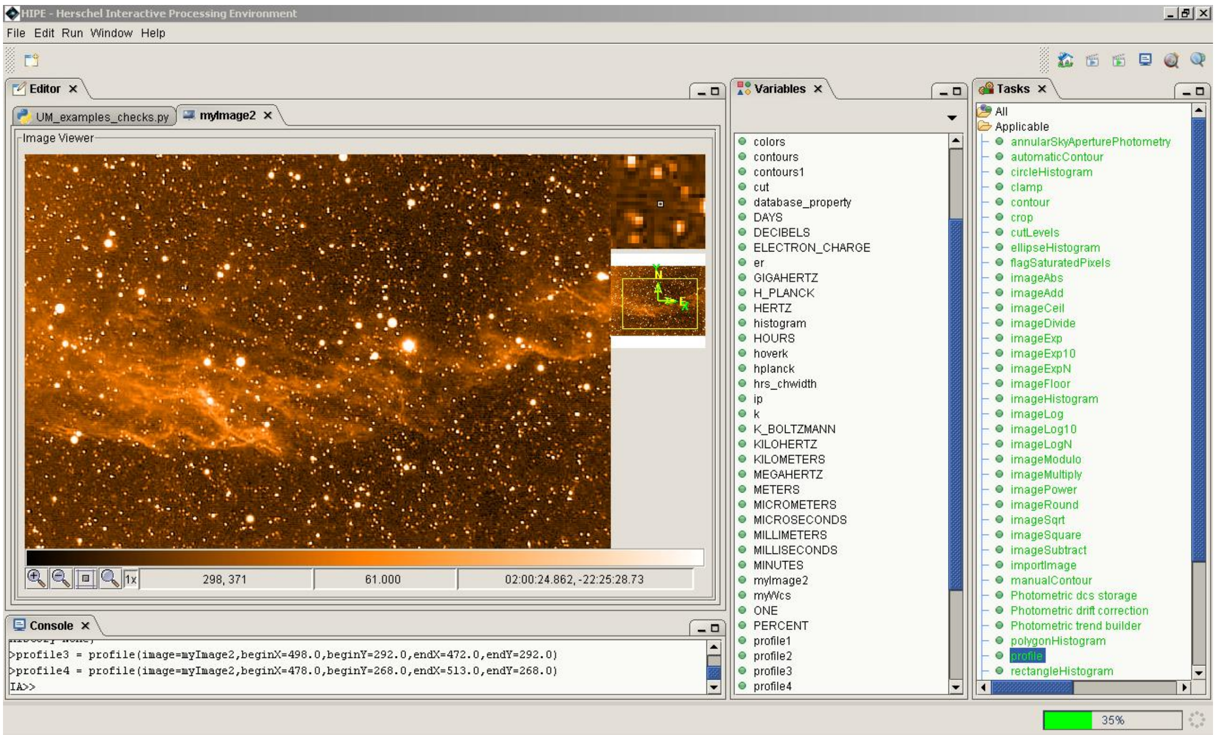


Figure 15.5. The available tasks show `profile` is available for "myImage2". Double-clicking on profile after first highlighting myImage2 in the Variables window creates a new display of the image together with the `profile` tool capabilities.Accessing the `profile` task

Now we can start drawing the straight line on the active image. The beginning of the line can be fixed by clicking once on the image. While moving the mouse over the active iamge, the straight line will be updated, until the end of the straight line is fixed by clicking a second time on the image. Simultaneously, the intensity plot along the straight line in an extension of the window below the displayed image (see Figure 15.6). The window is scrollable so the whole profile display can be seen by scrolling down (see Figure 15.7).

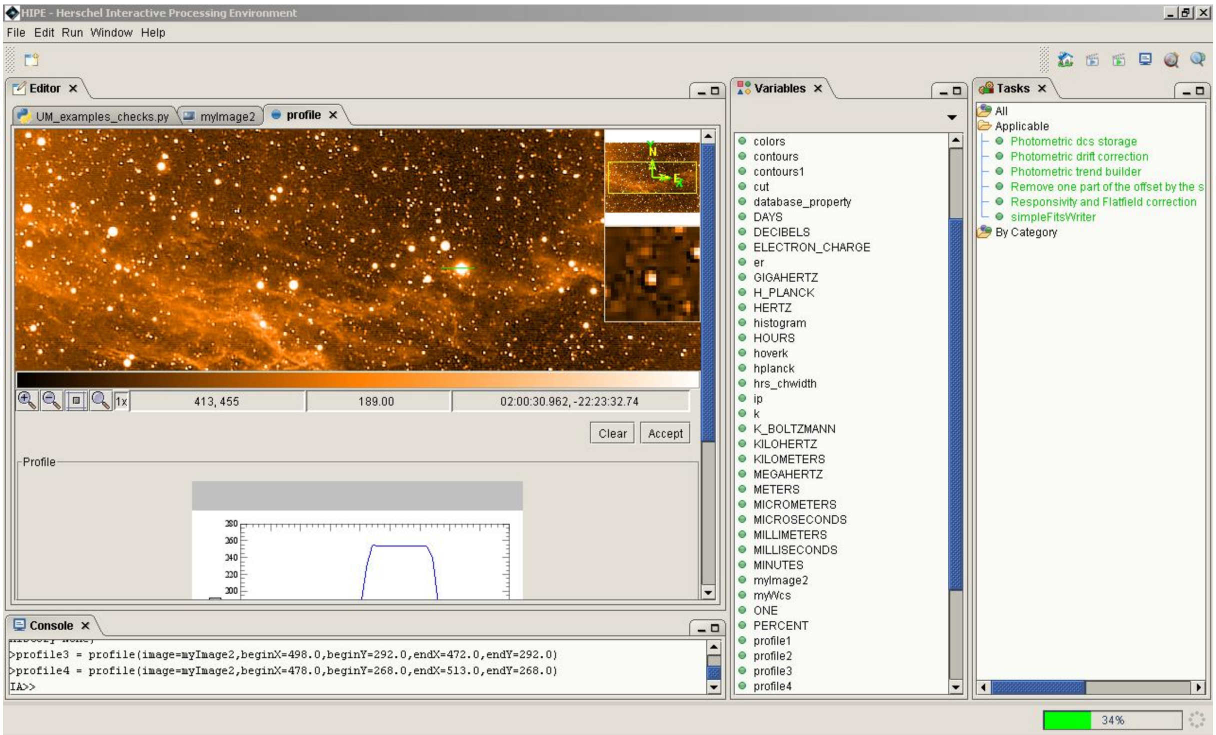


Figure 15.6. A 1D profile plot interaction

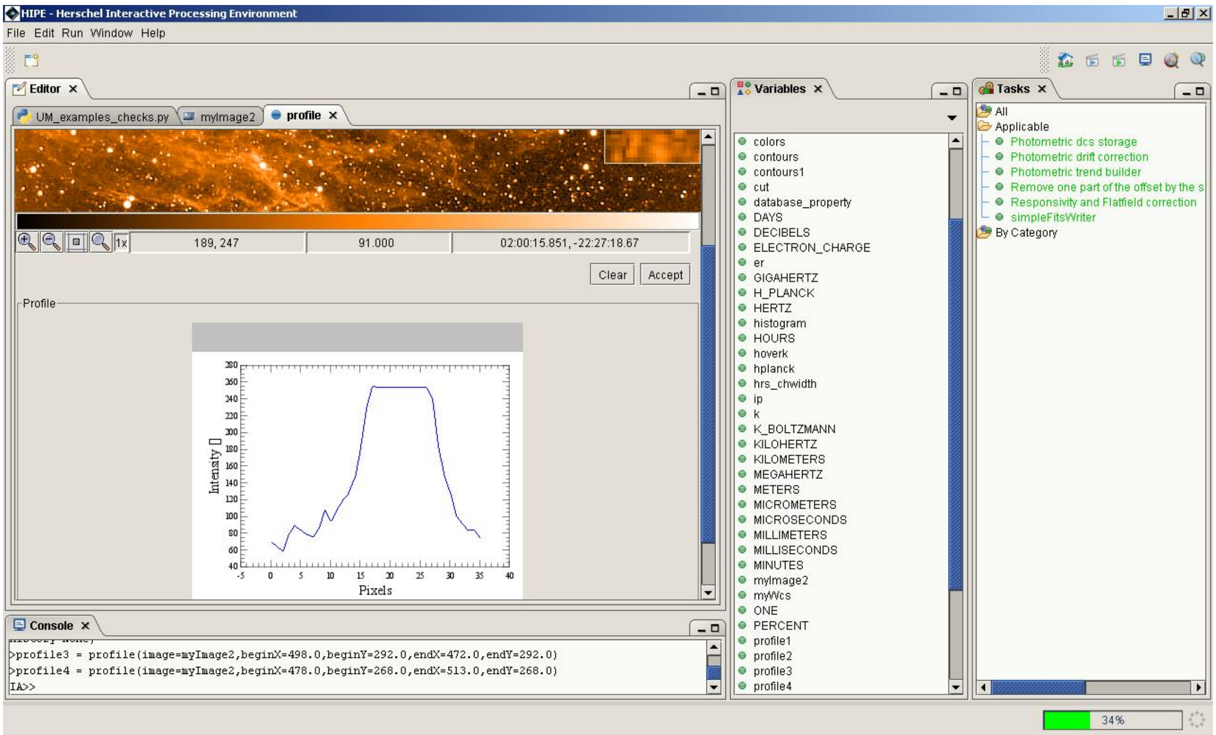


Figure 15.7. Same as for Figure 15.6 but scrolling down to show the profile display.

15.4.2. Area Histogram

One can make a histogram of an image as a whole, or of a certain region of interest which is specified by the user. This region can be bounded by a CIRCLE, an ELLIPSE, a RECTANGLE or a POLYGON, which has to be drawn on the image, or can be for the whole image.

We start the procedure for making an area histogram by choosing one of the `imageHistogram`, `polygonHistogram`, `circleHistogram`, `ellipseHistogram` or `rectangleHistogram` tasks.

First click on a `SimpleImage` in the list of Variables, e.g. "myImage2". Then double-click one of the histogram tasks in the Tasks view. This brings up a new image and activates the mouse so that an image area can be selected. To cover an area with a circle, ellipse or rectangle do a click-and-drag. On release, the area selected is shown overlaid on the image.

The histogram is constructed from the intensity values of the selected pixels and the input of the min and max cut levels in the boxes provoided, plus the number of bins for the histogram. Hitting the "Accept" button does several things.

- A histogram is formulated in the Editor window (scroll down).
- The equivalent command line is shown in the Console view which includes a named output object.
- The histogram values are placed in a dataset that appears in the Variables list. In Figure 15.8 this is called "histogram2".
- The output (e.g., "histogram2") appears highlighted in the Variables view and appears in the Outline view. Double-clicking this output value in the Variables view provides the histogram together with key information (see Figure 15.9).

For the `polygonHistogram` the only difference is that each corner is indicated by a single mouse click. The polygon area completion is indicated by a mouse double-click. Otherwise, this works in the same way as the other histogram tasks.

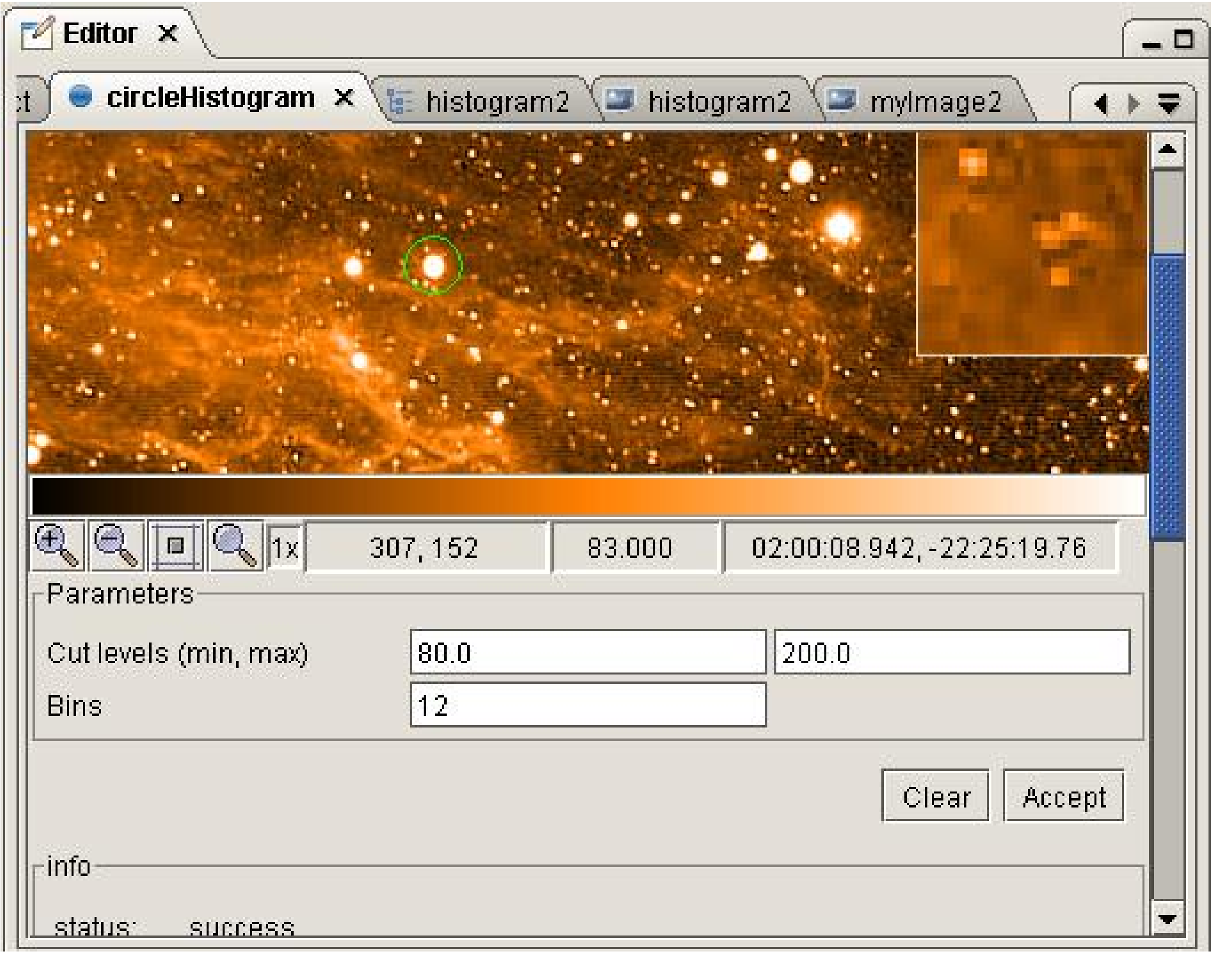


Figure 15.8. Circle histogram area selection and parameter selection. These appear in the HIPE "Editor" view.Circle histogram

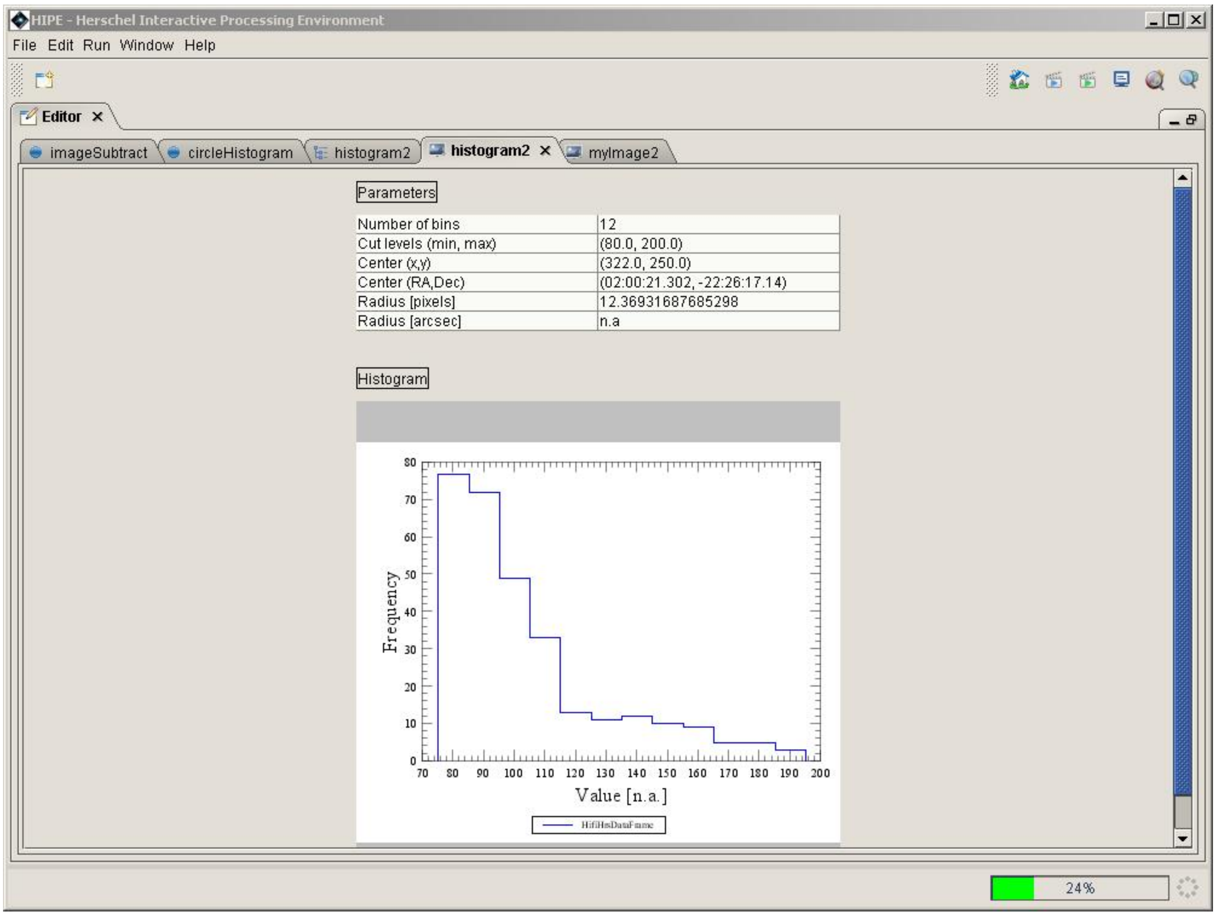


Figure 15.9. Display of the histogram results held in the histogram output in an expanded Editor view.Histogram display

15.4.3. Aperture Photometry

One can also perform aperture photometry on an image, using a circular target aperture and an annular or a rectangular sky aperture. There are five algorithms that can be used to estimate the sky : average, median, mean-median, the synthetic mode and daophot. In the mean-median method all values further away from the median than a specified number of times the standard deviation (i.c. 1.5) are discarded and the remaining values are averaged. The `daophot` method is a translation of the algorithm used in the aophot package from IDL to Java.

A start to doing annular aperture photometry can be made by choosing the `annularSkyAperturePhotometry` item in the Tasks menu following selection of the appropriate image in the "Variables" menu. This provides the image in the "Editor" view below which is the dialog for the aperture photometry task options (see ???). This is easiest seen by expanding the "Editor" view window and scrolling down below the image.

There are three mechanisms by which the photometry area can be identified.

- By click-and-drag mouse interactions.
- By pixel region selection.
- By sky coordinate selection.

The default is by mouse interaction. A single click on the image allows places a circle on the image at the mouse point. The user then inputs a value for the object aperture radius and inner and outer radii for sky subtraction. A selection should be made for the appropriate fitter (e.g. daophot) and pressing accept leads to an output in the variable "result". The circular radii are shown on the image (see Figure 15.10). To redo -- press the "Clear" button.

A sky position or pixel position can also be selected. Selection of either of these possibilities enables an update to the input screen allowing the sky/pixel values to be input. For the sky position (at present) the format of input is "02:00:39.4" for RA and "-22:27:20.6" for Dec. Note that the quotations are necessary as the input is a string. This is likely to be changed to allow various input types in the future.

The results can be displayed by double-clicking on the "result" variable shown in the "Variables" menu (see Figure 15.11)

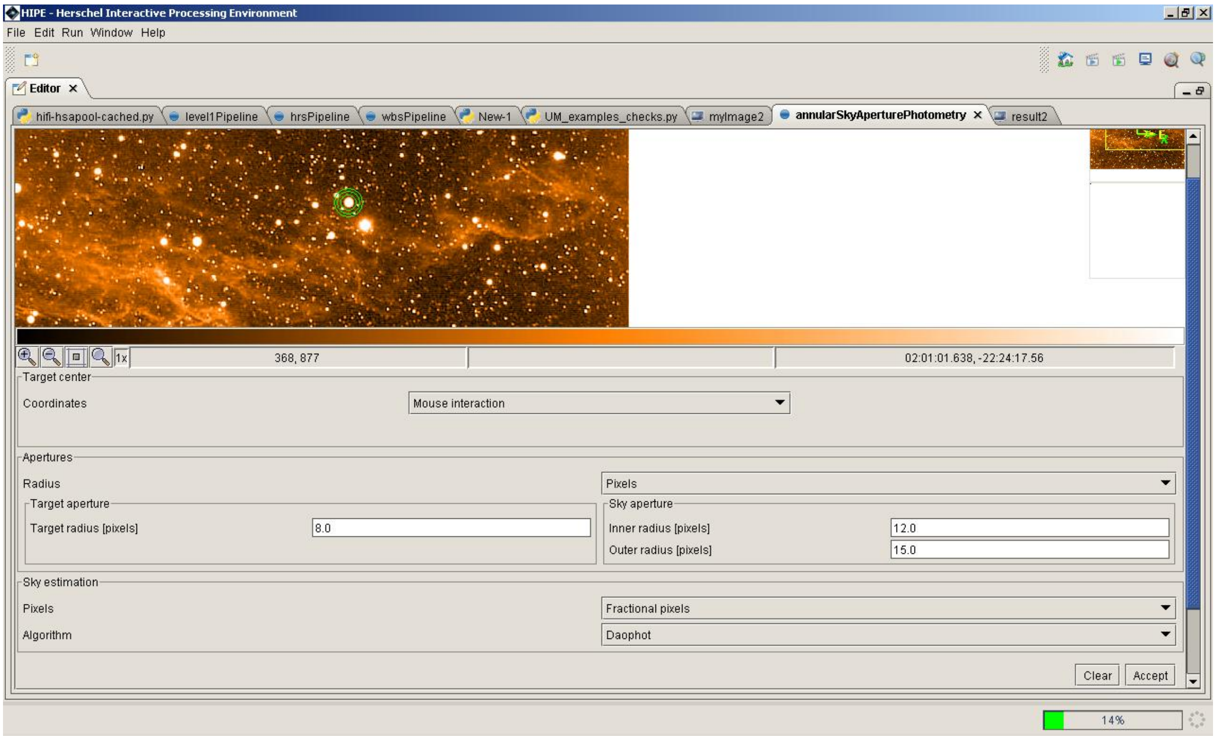


Figure 15.10. Aperture photometry with an annular sky aperture as displayed in HIPE.

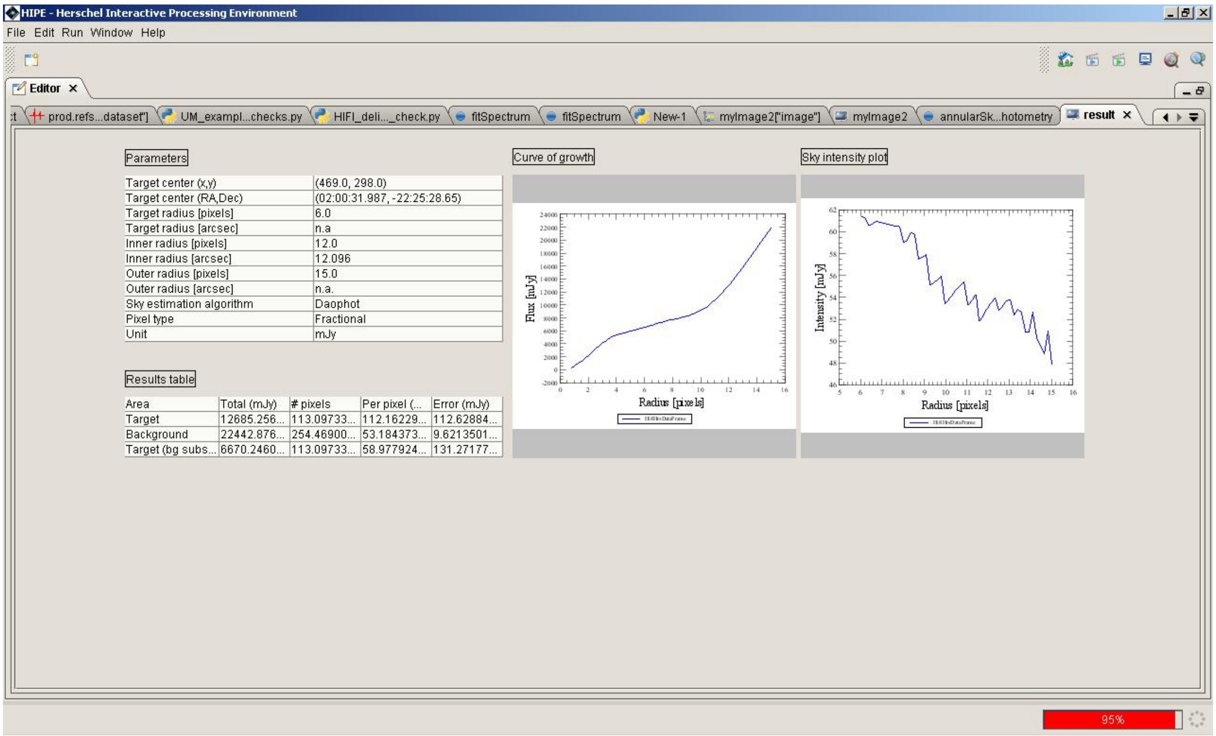


Figure 15.11. Aperture photometry results plot and tables. Note that n.a. relates to "not applicable" and typically will occur when units are not assigned to the image.Results of sky aperture measurement.

A similar capability is available for using a rectangular sky aperture. Rectangular aperture photometry can be done by choosing the `rectangularSkyAperturePhotometry` item in the Tasks menu following selection of the appropriate image in the "Variables" menu. Similar to the above, a single mouse click can be used to identify the target or a sky or pixel position can be indicated by the user. A rectangular sky aperture can then be selected by a click-and-drag across a region of the image (see Figure 15.12). Following the calculation for the first position, the same rectangular box can be used for the sky and a further single click on the image picks out a new object. Hitting the "Accept" button allows another result for this new position.

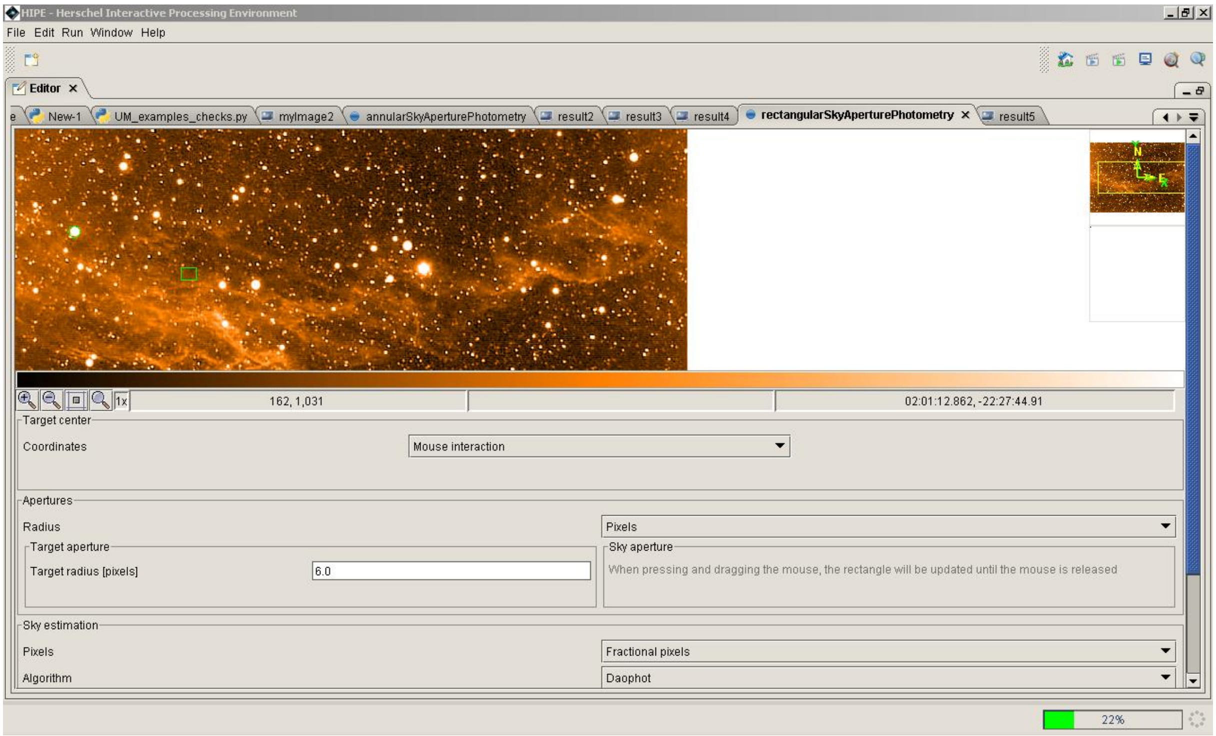


Figure 15.12. Aperture photometry with an annular sky aperture as displayed in HIPE.

The results for both aperture photometry tasks provide the curve of growth. This is a plot of the target flux as a function of the target radius. Such a plot can be used to see whether a valid target radius has been given. When an annular aperture is used to estimate the sky, a sky intensity plot is also shown. This plot shows the intensity per sky pixel as a function of a varying inner radius (the outer radius is fixed).

15.4.4. Contour Plotting

Another functionality of the toolbox is contour plotting. A contour plot connects all points in the image with the same intensity, like isobars on a weather map.

First we create the contours. We then, later, overlay these contours on any image we wish.

There are two methods for providing a set of contours for display. The first is an `automaticContour` where the number of contour levels and a min and max contour level are selected and the intermediate levels are generated automatically with linear, ln or log intervals of intensity. The second is a `manualContour` where the values of each contour level are individually put in by the user.

In either case we, as usual, start by clicking the name of the image we want to be countoured from the "Variables" list. Then we choose either `automaticContour` or `manualContour` by double-clicking these items in the "Tasks" list (see Figure 15.13for example).

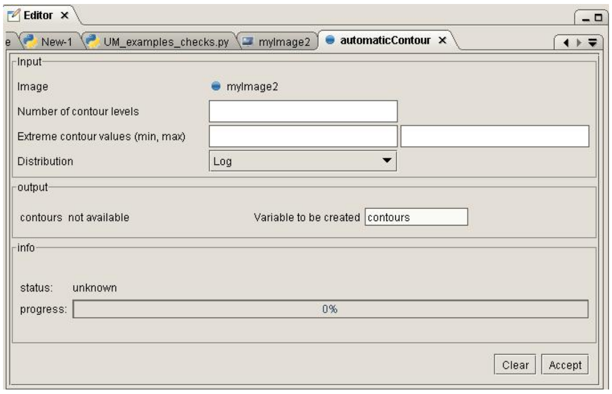


Figure 15.13. Dialog for `automaticContour`.

In either case, we create an output (editable value for user), e.g. "contours". When hitting the "Accept" button this is the variable that will store the contour results.

Chapter 16. How to Save/Play Back Scripts in HIPE

Herschel Editorial Board

16.1. Introduction

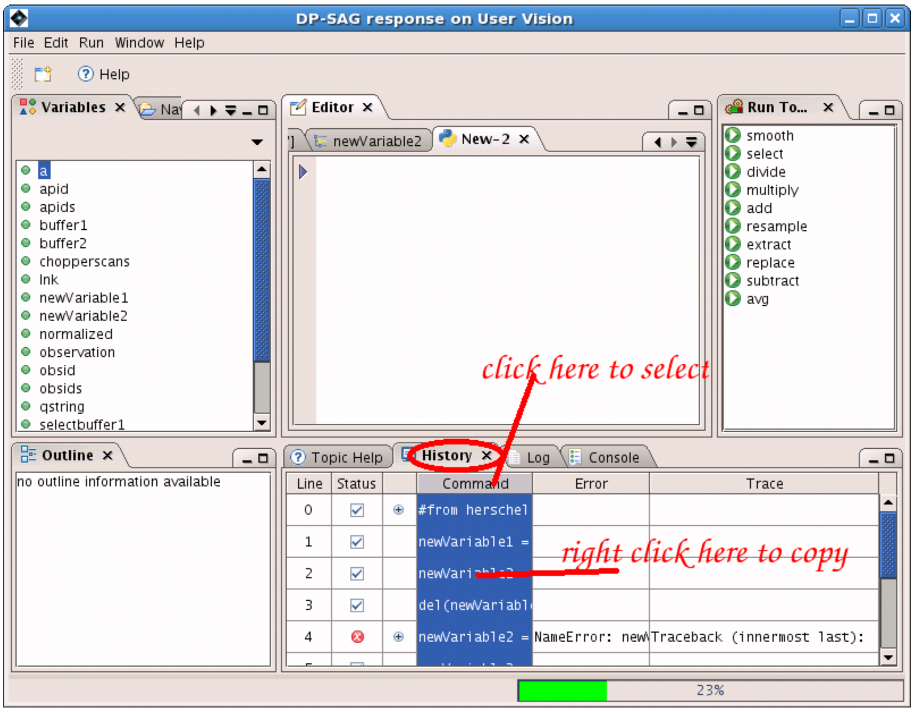
Hipe keeps a running record of all items typed or actiona taken using the graphical interface (Mouse points and clicks). The purpose of this article is to identify the steps you can take to save this information. The goal is to be able to keep a record of all actions and create a Jython script which can be resused or slightly modified.

16.2. How to save/ play back a script in HIPE

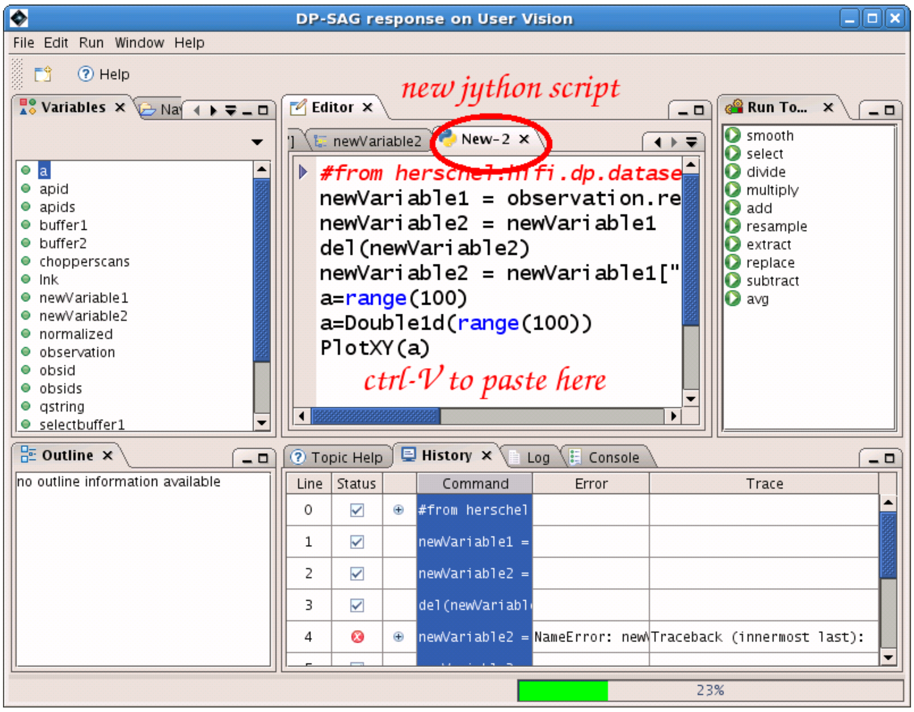
These are the steps to follow using to save all the commands which were given to HIPE during your session. In the tab bar with the Console view there is a tab called history. You can also bring this History view up from the Window--Show view pull down menu.

1. In the History view. Mouse left mouse click on the column called "Command". The entire column should be now selected.
2. From the "File" button at the top left of the HIPE window, create a new (blank) Jython script.
3. Then right-mouse click on the selection and Choose copy.
4. Move the cursor to the blank Jython script page and either select the "Paste" command in the Edit pull down menu or type Ctrl-V. The script will appear in the Editor window.
5. Your new Jython script can be saved for later importing (via the Navigator view).

The following screen shots show what is described above.



And...



To save the script to file (default extension .py), click on the Editor view tab that contains the script -- which brings the script to the foreground -- then hit CTRL-S. The file will be saved. If you have not already provided a directory and name for the script then you are prompted for one, otherwise the previous version is overwritten at the same place in your directory structure.

Alternatively -- click on the appropriate Editor tab (as above) and then go to the "File" pull-down menu at top left of HIPE. Go to "Save" or "Save As...".

16.3. How to Play Back a Script from the Command Line

The main way in which a script is developed and run in the HIPE environment is via the Editor screen, as described above. It can also then be saved -- as noted above. However, it is also possible to play back a saved script that is on the disk using the `execfile` command. Enter something similar to the following on the command-line of the Console view (do not forget the quotation marks).

```
execfile("<full path name><file name>")
```