PACS Products Explained

Katrina Exter

Zoltan Balog

Issue User . Version 1.0 -->
Mar 2015

PACS Products Explained

Katrina Exter
Zoltan Balog

Table of Contents

Chapter 1. Introduction

1.1. The scope

In this PACS Products Explained (PPE) document, the products that PACS spectroscopy and photometry provide to the astronomer are defined and explained. PACS observations are large and appear complex at first sight; along-side the raw and reduced astronomical data products, they also contain a lot of instrument and satellite data, all of which was/will be necessary to process/reprocess the science data. In addition, PACS science data consist of time-series measurements, during which the spatial and spectral ranges requested by the observer were gathered, and hence long series of data are required to produce every map or spectral cube. In this document we will explain what the astronomer needs to know to understand what makes up a PACS observation dataset, to locate the raw or fully-processed data products from within HIPE, and to understand the FITS files of the science-quality products.

- In Chapter 2 we discuss the **entire observation dataset**: the *ObservationContext*, which is what you get from the Herschel Science Archive (HSA) when you download an observation.
- In Chapter 3 we focus on the **pipeline products** that are most necessary for the science user.
- In Chapter 4 we explain the **Meta data and FITS keywords** of the Level 2/2.5 maps and cubes.
- In Chapter 5 we explain the **Standalone browse products**, which are found in the *ObservationContext* produced by SPG 13 and higher, also are also available directly from the HSA; these products are either exactly the same as (photometry), or different versions of (spectroscopy), the final-level pipeline products.

This document accompanies HIPE Track 13, and if you use the pipeline scripts of that version of HIPE you will be creating Track 13 products. There were some important changes in the higher level products produced by PACS between Track 12 and Track 13. Therefore, if you are working with data gotten from the HSA with a version number (the "SPG" version) of HIPE Track 12.x.x, there will be differences in what you see compared to data produced by HIPE Track 13.x.x. (With every new user release of HIPE, a bulk reprocessing of all the Herschel data is done and a new SPG version created: this takes several months to complete.) This is not, per se, a problem, but it does mean that the screenshots and some information in this document will be different if you are looking at SPG v 12 data, or SPG v 13 data or data you have reduced yourself in HIPE v 13. Therefore we will provide screenshots for both versions in this document.

A note about fonts used in this document. Italics are used to indicate a Product class: so *ObservationContext* is used to indicate a product of class "ObservationContext". Different product classes have different "capabilities", hold information of different types and in different ways, and can be accessed by different task and GUIs. For the user, the main reason to know the class of a product is to know what tasks and tools will work on them, and to know how to manipulate them on the command line.

Note

The "SPG" version of an *ObservationContext* can be determined either by looking at the columnar listing of observations in the HSA GUI before you download the data, or, once you have the *ObservationContext* in HIPE, by opening the Observation Viewer on it (double-click on it in the Variables panel, it is probably called "obs") and looking at the "Summary".

1.2. Other reference material

There is a large amount of documentation in HIPE about HIPE itself, about the tools provided to visualise and work with your data, and about the programing language used in HIPE. These documents are referred to as: *URM*: User's Reference Manual (the *HCSS URM* and the *PACS URM*), which is a dictionary of the tasks provided in HIPE; *PDRG*: the PACS Data Reduction Guides (PDRGs),

spectroscopy in *PACS Data Reduction Guide: Spectroscopy* or photometry in *PACS Data Reduction Guide: Photometry*; *DRM*: Developer's Reference Manual (the *PACS DRM* and the *HCSS DRM*), also known as java docs and which explains the capabilities and methods (i.e. how to use) all the classes in HIPE; the *PDD* (Herschel Product Definition Document), which is a reference guide for all the Herschel products.

Chapter 2. The Observation Context

2.1. Overview

The *ObservationContext* is a container of the entirety of the data for any Herschel—PACS observation. This includes: the raw and the SPG (Standard Product Generation: data reduced through the automatic pipelines) scientific data, the calibration files used in the SPG processing, the processing history, quality information, satellite and instrument data, pointing data, a lot of Meta data, and more. These data are held on disk as a collection of FITS files with a directory structure that HIPE can translate, and when you view the *ObservationContext* in HIPE you can browse through its contents using the **Observation Viewer**.

2.2. The layout of products in the ObservationContext

The first product you will work on is the *ObservationContext*, which is what you get from the HSA if you download an entire observation: see the Launch Pads, for spectroscopy in *PACS Data Reduction Guide: Spectroscopy* or photometry in *PACS Data Reduction Guide: Photometry*, for more information on downloading PACS data. An *ObservationContext* is a grand container of all the individual products and datasets that you get with each observation. To get an overview of its contents, open the *ObservationContext* with the Observation viewer (see the *HOG* chap. 15), accessed via a right-click on the observation as listed in the Variables or Outline panel; the viewer will open in the Editor panel.

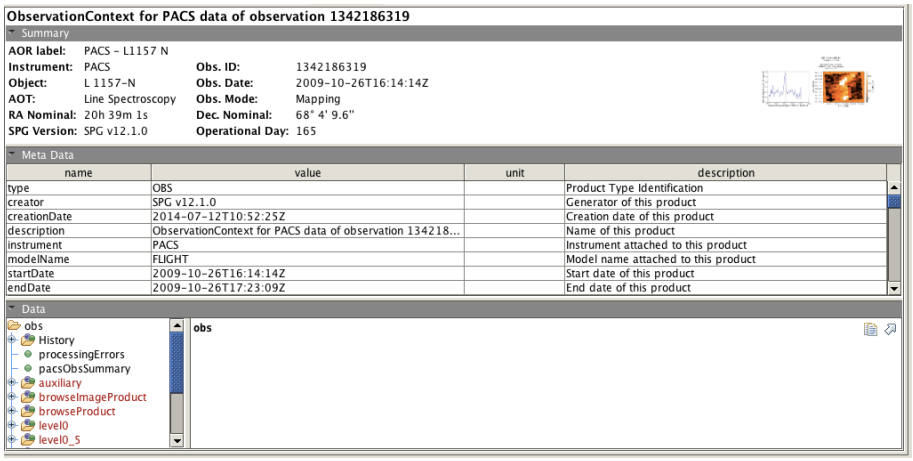


Figure 2.1. A spectroscopy ObservationContext, produced by SPG 12.1 and viewed with the Observation Viewer.

In the upper part you see a Summary tab and the Meta Data tab, at the lower left you see in the Data tab a directory-like listing of the layers in the *ObservationContext*. The small image in the **Summary tab** is a browse image, showing: (spectroscopy) a cube image made from each observer-requested wavelength range and the spectrum from the central spaxel of each of these cubes; (photometry) the Jscanam Level 2.5 maps, or the Level 2 maps if these are not available. Click on the image to enlarge it and click again to reduce. The **Meta Data tab** contains information about the observation; the Meta data shown belong to whatever is highlighted in the Data tab below, i.e. either to the entire *ObservationContext* or to any of the other products contained therein. The **Data tab** shows all the products, contexts, and data arrays that are held in the *ObservationContext*. To its right is an area where the default viewer for the selected object in the Data tab will open if the object is "viewable" (e.g. the Image viewer or the Spectrum Explorer). (Note that the default viewer will open whenever you click on an object, and for larger products it may take a while to open up.) From the Data tab you can extract products from the *ObservationContext* by dragging them to the Variables panel, to work on, view, or export to disk.

Let us go through these **products** in the Data tab. The "directory" for an observation called "obs" looks something like this:

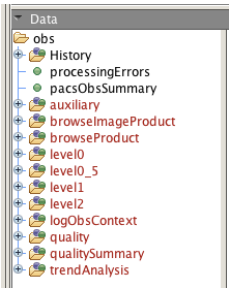


Figure 2.2. The Levels directory listing in an ObservationContext for a spectroscopy observation produced by SPG 12.1. An ObservationContext produced by a different SPG version will differ only slightly.

Listings in red have not yet been loaded into memory, those in black have been; to load a product into memory you only have to access it (e.g. click on it).

- The observer's science data are located in the *Contexts* called "+level0", "+level0_5", "+level1", "+level2", and for some observations "+level2_5". These are data at different stages of processing (0=raw, 0_5=partially processed, 1=more processed, 2=science-quality, 2.5=combination of level 2 cubes/maps, 3=for photometry, a super combination of observations into a single map).

- The "auxiliary" layer contains satellite and instrument data used in the pipeline, e.g. for calibrating the pointing and timing.

- The "History" contains a history of the reductions; there is a history for every layer you can look at, and its history is specific to that layer.

- "Quality" lists various quality information. For more detail on the History and Quality, see Section 2.4.

- For observations taken directly from the HSA there will be a "calibration" entry in the Data tab, containing the calibration tree that was used to reduce these data by the SPG pipeline (see chp. 2 of the PDRG: spec in *PACS Data Reduction Guide: Spectroscopy* for more information). For observations taken from someone else or which were saved to disk without specifically saving also the calibration tree, this will be absent.

  Note that the calibration tree for PACS can be gotten online, via HIPE, at any time, and you do not *need* to have that which comes with the observation.

- The "browseProduct" contains the standalone browse product (see Chapter 5), and the "browseImageProduct" contains the browse image (which you also see on the top-right side of the Summary tab).

- Finally, for SPG 13 products, the "pacsObsSummary" is the output of the pipeline task "obsSummary" and it contains a summary of the observation: AOT name, proposal details, wavelength ranges, and where they exist, quality information. To see this in a nice viewer, click directly on the "pacsObsSummary". For SPG 12 products this is located under the "browseProduct" entry.

2.3. Products in the pipeline-processed levels

Here we give an overview of the science products contained within the levels of PACS observations. Further detail on the datasets that make up these science various products can be found in Chapter 3.

2.3.1. Spectroscopy

Click on + next to the "level0/0_5/1" for a listing of their contents, e.g.,

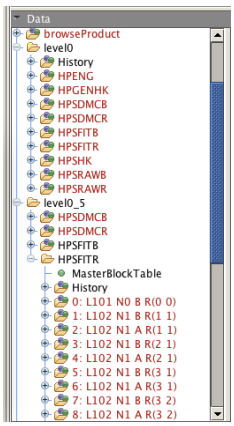


Figure 2.3. The level 0 and 0.5 contents for an ObservationContext produced by SPG 12.1 and higher

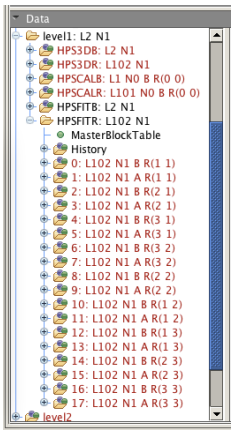


Figure 2.4. The level 1 contents for an ObservationContext produced by SPG 12.1 and higher

Each entry in these screenshots is a *context* containing other products. For example, many individual **H**erschel **P**acs **S**pectroscopy **Fit**ted "Frames" **R**ed products are held in a context called **HPSFITR**. The individual *Frames* products are different parts of the same observation. In the terminology of the PACS pipeline we call these individual *Frames* "slices". The same arrangement applies to every HPSXXX context in a PACS spectroscopy observation.

In the Data tab, each of these *Frames* has a title that indicates what is unique about it: in this case the 18 level 1 *Frames* listed under HPSFITR in the screenshot have:

- a sequence digit (0—18)
- a spectral line sequencer ("L 102")
- a nod-sequence number (only N1 for this observation)
- a nodding position A and B: for chop-nod observations these refer to the chopper position, for unchopped observations they refer on-source (B) or off-source (A)
- a raster position (R(1 1) is position row 1 column 1, R(2 1) is row 2 column 1, etc)

The total number of *Frames* or *PacsCube*s that are present at Level 1 is determined by: the number of lines the observer requested, the number of raster pointings, the number of nod-sequences, this combination then multiplied by 2 for their being repeated in nod A and nod B (unchopped line and chop-nod range) or by 1 (unchopped range). In the screenshot above there are 18 *Frames*: 1 line id, 9 pointings, 1 nod sequence, and 2 nods.

Hover with the cursor over any of the objects shown in the screenshot, and a banner comes up telling you what class they are and, when relevant, what type of information are contained therein.

The other *contexts* in the screenshot above are more science contexts: **HPSFITB** are the blue equivalent to the HPSFITR, and **HPS3DR|B** are blue and red cubes (**3D**imensional) of class *PacsCube*. There are the same number of *PacsCube*s as there are *Frames*.

The other layers in the screenshot above contain engineering data (HPSENG), satellite housekeeping data (HPSGENK and HPSHK), DecMec data (HPSDMC) which are used in some of the masking pipeline tasks, and data of the central detector pixel which is downlinked in raw format (HPSRAW). (You can also read the *PDD* for definitions of the various HPSXXX slices).

Click on + next to the "level2" for a listing of its contents:

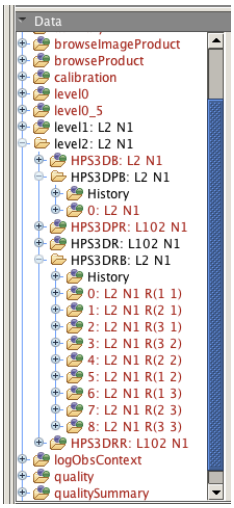


Figure 2.5. The level 2 contents for a spectroscopy ObservationContext produced by SPG 12.1 for a raster observation

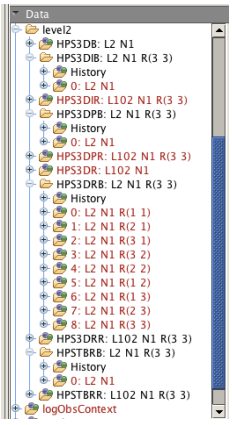


Figure 2.6. The level 2 contents for a spectroscopy ObservationContext produced by SPG 13.0 for a raster observation

In level 2 the science-quality end-of-pipeline products can be found. As can be seen from the screenshot above, there are some differences between the contents in SPG 12.1 and in SPG 13.0. Both contain the **HPS3D[B|R]**; which has the same number of slices as its equivalent in level 1. However, for line-scan observations (chop-nod or unchopped) these level 2 cubes have had a flatfielding applied to them, whereas those at level 1 have not. For range scan observations the level 1 and 2 are identical, since no flatfielding is applied in the SPG pipeline in either Track.

In addition there are:

- **12:1: HPS3DR[B|R]** and **HPS3DP[B|R]**. These *contexts* also contain cubes (hence the "3D"), of class *PacsRebinnedCube* and also called rebinned cubes, and of class *SpectralSimpleCube* but also called projected cubes.

The rebinned cubes are created from the preceeding *PacsCube*s with the pipeline task specWaveRebin, and where nod B and nod A were present in the *PacsCube*s, these have been combined (hence there are 9 cubes in the screenshot above). These cubes have the native footprint of PACS, that is a slightly irregular spatial grid of 5x5 spaxels of 9.4" size each.

The projected cubes are then created from the rebinned cubes and are intended for mapping/raster observations, whereby the individual rebinned cubes (all of a different pointing but the same line id) are mosaicked together into a single, "projected", cube with a regular spatial grid. This is done by the pipeline task specProject and is done to all observations by the SPG pipeline, whether they are mapping or a single pointing observations. For oversampled mapping observations the spaxel sizes are 3", for pointed observations they are 0.5", and all others have sizes of 1.5". There is only one projected cube (HPS3DP[B|R]) per line id, since all the pointings have been combined.

- **13.0:** In addition to the cubes mention above, in Track 13 we also provide two other *context*s of mosaic cubes, i.e. those with a regular spatial grid: **HPS3DD[B|R]** (drizzled cubes) and **HPS3DI[B|R]** (interpolated cubes). All observations have, in Level 2/2.5, the *PacsCube*s and *PacsRebinnedCube*s and in addition two of the projected, drizzled, or interpolated cubes. The rules are:

  - Pointed observations: the projected and the interpolated cubes. The projected cubes have spaxel sizes of 0.5" and the interpolated cubes have spaxels of 4.7".

  - Undersampled mapping observations: the projected and the interpolated cubes. The projected cubes have spaxels of 1.5" and the interpolated cubes have spaxels of 4.7"

  - Nyquist and oversampled mapping observations: for line scans, the drizzled and projected cubes, with spaxels dependent on the actual spatial sampling; for range scans and the full SED scans, the projected cubes (3" spaxels) and interpolated cubes (4.7" spaxels).

  Nyquist mapping observations have:

  - blue camera: raster step sizes of #16" with at least 3 steps in the raster (in both directions)

  - red camera: raster step sizes of #24" with at least 2 steps in the raster (in both directions)

  Oversampled mapping observations have:

  - blue camera: raster step sizes of #3" with at least 3x3 steps in the raster

  - red camera: raster step sizes of #4.5" with at least 2x2 steps in the raster

  In addition one of the cube *context*s per observation is provided in an "equidistant" format; this cube *context* has an HPS name with **EQ** added to it. These cubes have a wavelength grid that is linear and regular (for the standard cubes the dispersion of the wavelength grid depends on the resolution, which changes with wavelength). These cubes are also "standalone browse products", which are also offered as a download directly from the HSA. To learn more, go to Chapter 5.

  Another new product, **HPSTBR[B|R]**, has been created for Track 13. This contains all the information that is in the rebinned cubes (the fluxes, errors, positions, etc) but in a tabular format.

The Level 2.5: is found in the on-source observation of an unchopped range scan (on-source and off-source observations are separate for this observing mode). In the Level 2 of the on-source and off-source observation are the end products of the pipeline for that single observation. In Level 2.5 are the background-subtracted results: the Level 2 of the off-source observation subtracted from the Level 2 of the on-source observation. The layout and products contained in this level are exactly the same as in Level 2.

Note

For unchopped range raster observations, it is possible that the observer asked for only a single pointing for the off-source position. In this case, note that it will be normal that

> the type of cube produced at the end of the pipeline for the off-source observation may be different to the type produced for the on-source observation: the off-cube will be that for a single pointing (i.e. an HPS3DR[R|B] and an HPS3DI[R|B}) and the on-source cube will be that for a raster (HPS3DR[R|B]; and HPS3DI[R|B] or HPS3DP[R|B], depending on whether the raster is undersampled or has a higher sampling).

The table below summarises the contents of an *ObservationContext*. These contents are explained further in the next chapter.

Table 2.1. The contents of the Levels of an ObservationContext

Level	Context	Class	Contains	SPG version
ObservationContext, 0, HPSXXX of level 0.5, 1, 2, 2.5	History	*HistoryDataset*	A history of tasks that were run on the particular object the History is attached to; the most useful histories are held within the Level X HPSFITB|R or HPS3DXX layers themselves	12.1, 13.0
0	HPENG	a *context*	A table of engineering data	12.1, 13.0
0	HPGENHK	a *context*	A table of satellite housekeeping data	12.1, 13.0
0, 0.5	HPSDMC[R|B]	a *context*	A table of DecMec data, used in the pipeline in masking tasks	12.1, 13.0
0, 0.5, 1	HPSFIT[R|B]	*SlicedFrames*	Individual *Frames*, which contain all the signals gathered from the astronomical source. At level 0.5 it is "sliced" by nod position, nod cycle, wavelength, pointing: one *Frames* becomes many *SlicedFrames*, each containing one or more *Frames*.	12.1, 13.0
0	HPSHK	a *context*	Herschel PACS Spectroscopy housekeeping information table	12.1, 13.0
0	HPSRAW[R|B]	*SlicedRawFrames*	Raw data *Frames*, which contain data from the astronomical source but only in one spatial pixel, and only	12.1, 13.0

			used in the saturation pipeline task	
1, 2	HPS3D[R|B]	*SlicedPacsCube*	The first cubes produced by the pipeline	12.1, 13.0
2, 2.5	HPS3DR[R|B]	*SicedPacsRebinnedCube*	The second cubes produced by the pipeline, and one possible pipeline end-product to adopt	12.1, 13.0
2, 2.5	HPS3DP[R|B] and HPS3DEQP[R|B]	*SlicedPacsProjectedCube*, also *SpectralSimpleCube*	Projected cubes produced by the pipeline from the rebinned cubes; are one possible pipeline end-product to adopt. In Track 13 these are provided only for range scan observations. Also sometimes provided with an equidistant (evenly binned) wavelength grid ("EQ").	12.1, 13.0
2, 2.5	HPS3DD[R|B] and HPS3DEQD[R|B]	*ListContext* of *SpectralSimpleCube*s	Drizzled cubes produced by the pipeline from the preceding PacsCubes; are one possible pipeline end-product to adopt. Provided only for line scan observations. Also sometimes provided with an equidistant (evenly binned) wavelength grid ("EQ").	13.0
2, 2.5	HPS3DI[R|B] and HPS3DEQI[R|B]	*ListContext* of *SpectralSimpleCube*s	Interpolated cubes produced by the pipeline from the rebinned cubes; are one possible pipeline end-product to adopt. Also sometimes provided with an equidistant (evenly binned) wavelength grid ("EQ").	13.0

2, 2.5	HPSTBR[R|B]	*SlicedPacsSpecTable*	The data in the *SlicedPacsRebinnedCubes* presented as a table	13.0

2.3.2. Photometry

The L**evel0/0_5/1** contain products of class *Frames* at various stages of processing and calibration, with **level1** being fully calibration for astrometry and flux. *Maps* are found at the subsequent levels.

As in the case of the spectroscopy, if you hover with your cursor over these various layers, called HPPXXX (HPP=**H**erschel **P**ACS **P**hotometry), a banner comes up telling you what class the products are: for example, the level 0.5 HPPAVGR|B are *SlicedFrames*: this being a type of *ListContext*, which is a special container (list) of other products (in this case, a list of *Frames*). There are several products held in the other levels (calibration, auxiliary, satellite, housekeeping, pointing...), which generally have one or more datasets which contain the actual data.

The astronomer will be interested in the following *context*s: **HPPAVG[B|R], HPPMAP[B|R], HPPCOM[B|R]** and **HPPMOS[B|R]**. The first is the *ListContext* that contain the *Frames* mentioned previously, and the others hold the final projected maps, which are *SimpleImage*s and may be held in a *MapContext*, in all cases always for the blue and the red (B|R) camera. The *Frames* are found in Level 0 to 1, the *MapContext/SimpleImage*s are Level 2 and 2.5 products, and at Level 2.5 you also find combined maps of the same class. These are shown in the screenshot below: the other layers contain engineering (HPSENG), satellite housekeeping (HPSGENK and HPPHK), and DecMec (HPPDMC) data, which are used in some of the masking pipeline tasks. See also the *PDD* for definitions of the various HPPXXX slices.

By looking into the level0 HPPAVGB (click on the + next to it) you can see the layers inside it. At Level 0 there will always only be one layer (numbered 0), which is a single *Frames*. A *Frames* holds various other layers containing your astronomical and associated data. You can see some of these in the screenshot below: the Status table, Mask, the BlockTable, and the Signal.

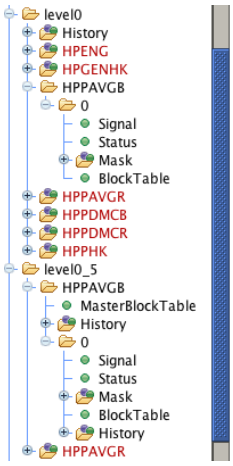


Figure 2.7. The layers in the level0 and 0.5 product of an ObservationContext (SPG 13)

At Level 1, shown in the figure below, you see the same data structure as in Level 0.5,

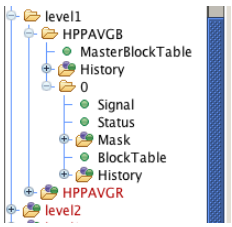


Figure 2.8. The layers in the level1 product of an ObservationContext (SPG 13)

Finally there is the **level2**, **level2.5** and sometimes a **level3**, inside which you will find maps. In **level2** you find the map of your actual observation: HPPMAPR|B,

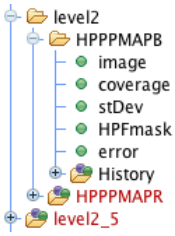


Figure 2.9. The layers in the level2 product of an ObservationContext (SPG 13)

In **level2.5** there are several maps which are the combination of a scan—cross scan pair (the most common observing mode of the PACS photometer). The difference between the various combined maps provided is the mapping technique used to create them, and note that between SPG 12 and SPG 13 there has been a slight change in which map makers are used (see below). The **HPPHPFMAP[R|B]** context contain *maps* which were all created using High Pass Filtering and PhotProject tasks; **HPPNAVMAP[R|B], HPPMADMAP[R|B], HPPCORMAP[R|B]** maps were created using MADMap; **HPPJSMAP[R|B]** maps were created using JScanam; and the **HPPUNIMAP[R|B]** maps were created by Unimap. See the PDRG in *PACS Data Reduction Guide: Photometry* for a description of the different mappers.

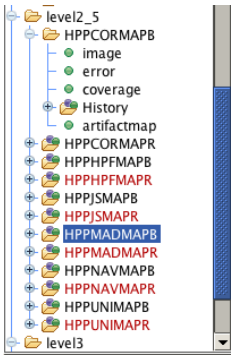


Figure 2.10. The layers in the level2.5 product of an ObservationContext (SPG 13)

Some observations also have a **level3.** The maps here are each the combination of all observations taken with the PACS photometer during the Herschel mission of the same object (the same coordinates). Note that since photometry observations always produce a red filter image and either a blue *or* a green filter image, for observations where not the same blue filter was used, only a red combined level3 image will be provided. The structure is similar to that of **level2.5** except that only one of the MADmap maps is retained, and there is no HPF map.

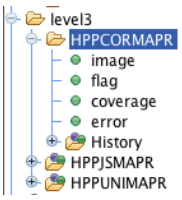


Figure 2.11. The layers in the level3 product of an ObservationContext (SPG 13)

The differences between the products found in an SPG 12 and an SPG 13 observation are the following:

- **SPG 12**: the maps created by the tasks Madmap, JScanam, and photProject+highPass filter are present

- **SPG 13**: the maps created by the tasks Madmap, JScanam, Unimap, and photProject+highPass filter are present

See Chapter 3 for details of the datasets contained in the various maps.

Table 2.2. The contents of the Levels (SPG for Track v 13 and higher)

Level; SPG	Product	Description
0; 12, 13	History	A history of tasks that were run on this object; the most useful histories are held within the Level X HPSFXIT or HPS3DXX layers themselves
0; 12, 13	HPENG	Herschel PACS engineering information table
0; 12, 13	HPGENHK	Herschel PACS general housekeeping information table
0, 0.5; 12, 13	HPPDMC[R|B]	Herschel PACS Photometry DecMec information table; used in the pipeline
0, 0.5, 1; 12, 13	HPPAVG[R|B]	*Frames or SlicedFrames*, holding individual *Frames*, red and blue
0; 12, 13	HPPHK	Herschel PACS Photometry housekeeping information table
2; 12, 13	HPPPMAP[R|B]	SimpleImage, holding the final map of the observation projected using PhotProject
2.5; 12, 13	HPPNAVMAP[R|B]	*MapContext* holding the final combined naive map of the scan and cross-scan pair created by MADMap
2.5; 12, 13	HPPMADMAP[R|B]	*MapContext* holding the final combined map of the scan and cross-scan pair created by MADMap
2.5; 12, 13	HPPCORMAP[R|B]	*MapContext* holding the final combined artifact corrected map of the scan and cross-scan pair created by MADMap
2.5; 12, 13	HPPJSMAP[R|B]	*MapContext* holding the final combined map of the scan and cross-scan pair created by JScanam
2.5; 12, 13	HPPHPFMAP[R|B]	*MapContext* holding the final combined map of the scan and cross-scan pair created by high-pass filtering and photProject
2.5; 13	HPPUNIMAP[R|B]	*MapContext* holding the final combined map of the scan and cross-scan pair created by UNIMAP
3; 13	HPPUNIMAP[R|B]	*MapContext* holding the final combined map of all the obser-

		vation taken on the same object created by UNIMAP
3; 13	HPPCORMAP[R|B]	*MapContext* holding the final combined map of all the observation taken on the same object created by MADMap
3; 13	HPPJSMAP[R|B]	*MapContext* holding the final combined map of all the observation taken on the same object created by JScanam

2.4. Tips for reading the Quality and History information

In an *ObservationContext* are history information: "+History", in the *ObservationContext*, in level0 and in the HPSXXX or HPPXXX of the other levels and quality information ("+quality", a listing within the *ObservationContext*). The History contains information about the processing history of the *ObservationContext* or whichever layer the History belongs to. In the Quality is information added by Quality Control at the HSC or added by the SPG pipeline processing in an automatic manner. A nice way to inspect these both is to click on them (in the Data tab) and the appropriate viewer will open.

- *History*. This contains the automatically generated script of actions performed on your data, a history of the tasks applied to the data, and the parameters belonging to those tasks. If, while in the Observation viewer, you click on the highest level +History, that of the *ObservationContext* itself, you will see only a few tasks listed in the table in the viewer. Go down to one of the HPSXXX/HPPXXX in the Levels (level0, 0.5 etc.) and the +History there are associated with those particular products, and so on down the various layers. This History is useful mainly if you want to see what tasks were run on the observation you are looking at, and what parameters were specified. The History should get updated when you do things yourself to your observation.

- *Quality: quality and qualitySummary*: Quality information is tabulated in a browser which is accessed by clicking on either of these *contexts* in the Data tab.

  If there is a "qualitySummary", this is what should be inspected, as it means that the observation has been checked for particular problems by a person at the HSC. As well as quality information added by the HSC, flags raised by the SPG or added automatically during the observations can be found. No comment = no problems. Where only the "quality" is present, then only the automatically-generated flags are provided.

  Much information about the quality of observations and known problems can also be found on the Herschel DP Known Issues page (http://herschel.esac.esa.int/twiki/bin/view/Public/DpKnownIssues). Information about the quality report is also given in Chap 6.2 of the *PDD*. Note that you can also get a quality report directly from the HSA: see chap 1.4.3 of the *DAG*.

Chapter 3. The Pipeline Products

3.1. Introduction

During the SPG processing of PACS data, i.e. while the pipelines are run in automatic mode, the intermediate and final produces are placed into the *ObservationContext* in the Level0_5, 1, 2 and 2.5 *contexts*. The names of the products added and how many of them are added will depend on which version of the SPG, i.e. which version of HIPE, the data are processed with. During interactive processing of PACS data by the astronomer the same set of intermediate and final products are created, and it is up to the astronomer to decide whether to add them to the *ObservationContext* or not. This interactive processing will, obviously, have the version number of the HIPE session being used, and this may be a more recent version than the version of HIPE that produced the SPG result. As was mentioned in Chapter 1, it is possible that you are using HIPE 13.0 to look at data which has an SPG version 12.1. Therefore we will explain the pipeline products in the *ObservationContext* as produced with SPG version 12.1, and explain also the products created by the interactive pipelines in HIPE version 13.0.

3.2. Spectroscopy

The spectroscopy pipeline products are *Frames*, *PacsCube*, *PacsRebinnedCube*, and different permutations of *SpectralSimpleCube*s. The lowest-level, raw products, are of class *Frames*, and as the pipeline proceeds the *Frames* are turned into *PacsCube*, then *PacsRebinnedCube*, and finally a projected cube (*SpectralSimpleCube)*, a drizzled cube (a *SpectralSimpleCube*) or an interpolated cube (a *SpectralSimpleCube*). The science-quality products are the rebinned, projected, interpolated or drizzled cubes: which cube is the most suitable for science analysis depends on the type of observation and on the astronomical source. More instruction can be found the PDRG chp. 1 in *PACS Data Reduction Guide: Spectroscopy* and chp. 8 in *PACS Data Reduction Guide: Spectroscopy*.

3.2.1. The science products

3.2.1.1. Slicing in the pipeline

A single PACS spectroscopy observation can contain data of several separate wavelength ranges and several pointings (e.g. for raster mapping observations), and we call these "slices". In addition, there is always red camera and blue camera data present, these contained in separate *context*s. At each stage in the pipeline all the pointing and all the wavelength range "slices" are held together in *ListContext*s: several individual *Frames* will be held together in a *SlicedFrames*, and ditto for the cubes. The slicing is done in the pipeline, the *Frames* from Level 0 is split into separate entities, the borders being wherever changes in the pointing (raster position, nod position) and wavelength occur. This makes handling larger observations by the pipeline easier.

The class of the containers of these slices is a type of *ListContext*: the *ListContext*s that contain the *Frames*, *PacsCube,* and *PacsRebinnedCube* slices are, respectively, *SlicedFrames*, *SlicedPacsCube*, *SlicedPacsRebinnedCube*, and for rest of the Level 2 cubes and the table, the class is a *ListContext*.

To work out what slice contains what data, you can use the pipeline task "slicedSummary" to see something like this (see the *URM* to learn more about this task):

```
# For a SlicedFrames taken straight from the ObservationContext
HIPE> slicedSummary(obs_level0_5_HPSFITB)
noSlices: 5
noCalSlices: 1
noScienceSlices: 4
slice#  isScience  onSource  offSource  rasterId lineId band  dimensions
 wavelengths
0   false  no    no     0 0   [0,1]  ["B2A","B3A"]   [18,25,1680]  50.389 - 76.325
1   true   yes   no     0 0   [2]    ["B2A"]         [18,25,1500]  71.932 - 73.922
2   true   no    yes    0 0   [2]    ["B2A"]         [18,25,1500]  71.932 - 73.922
```

```
3   true   no    yes    0 0   [2]    ["B2A"]         [18,25,1500]  71.932 - 73.922
4   true   yes   no     0 0   [2]    ["B2A"]         [18,25,1500]  71.932 - 73.922
```

(The output for the slicedDrizzed|Projected|InterpolatedCubes contains fewer columns, as some are redundant for these products.)

The *SlicedXXX* also contain organisational and other information: a History, Meta data, and for *SlicedFrames* and *SlicedPacsCubes* also a "MasterBlockTable". A single "BlockTable" is essentially a table of the data blocks, where a "block" is a group of datapoints taken while the instrument was in a particular setting: a particular wavelength range, or nod position, or position in a raster pointing pattern, or repetition number, for example. A BlockTable is the organisation table for a single *Frames* or *PacsCube*, and the MasterBlockTable is the concatenation of all the individual BlockTables of the *Frames* or *PacsCubes* slices.

There are various methods to interact with the sliced products (see the *PACS DRM* entries for the particular product to learn more). For the average user, the most commonly-used methods are:

```
# SlicedFrames, SlicedPacsCube, SlicedPacsRebinnedCube: extract a slice
# The example is from a sliceFrames, but replace "Frames" with "PacsCube"
#  or "PacsRebinnedCube" for the other products

# Get a slice number "slice" (where 0 is the first slice)
# The product returned is a Frames
slice = 1
frames = slicedFrames.get(slice)

# Get science (i.e. excluding the calibration slices) slice
# number "slice". The product returned is a Frames
frames = slicedFrames.getScience(slice)

# Another method to get slice number "slice"
# The product returned is a Frames
frames = slicedFrames.refs[slice].product

# Get a slice another way
# The product returned is a SlicedFrames
slicedFrames_sub=slicedFrames.selectAsSliced(slice)

# Using a task; see the URM to see all the parameters
# you can select on. A SlicedFrames is returned
# Here it is possible to specify more than one slice number for sliceNumber
slicedFrames_sub = selectSlices(slicedFrames, sliceNumber=[0,1,3])

# replace, add etc (will work for all PACS sliced products)
# Adds a new frame to the end
slicedFrames.add(myframe1)
# Inserts a new frame before the given slice no. (1 here)
slicedFrames.insert(1,myframe1)
# Removes slice no. 1 (which is the second slice, not the first)
slicedFrames.remove(1)
# Replaces slice no. 1 with a new Frames
slicedFrames.replace(1,myframe)

# concatenate any number of SlicedXXX using a task
sliceFrames_big = concatenateSliced([slicedFrames1, slicedFrames2])


# ListContext: the SpectralSimpleCube context called
#              slicedDrizzed|Projected|InterpolatedCubes
# The result is held in a basic ListContext
slice = 1
# The product returned is a SpectralSimpleCube
pcube = slicedPCubes.refs[slice].product
# Add a new cube on to the end
slicedPCubes.refs.add(ProductRef(newPCube))
# Add a new cube to a particular position (the first, position 0)
slicedPCubes.refs.add(1, ProductRef(newPCube))
# Remove a cube from a particular position
slicedPCubes.refs.remove(1)
```

For the full details of the methods that work on *SlicedFrames* etc., look up the class names in the *PACS DRM*. For the *ListContext*s of *SpectralSimpleCube*s you need to consult the *HCSS URM*. You can also extract slices from a *SlicedXXX* list in a more varied way by selecting on the MasterBlockTable: see Section 3.2.2.3 for a guide.

Tip

As with all other products in HIPE, their names are not the actual "thing" itself, but rather a pointer to the address in memory where the "thing" is held. That means that if you extract a slice out of a *SlicedXXX* list and change it, you are not only changing the copy, but also the original, as they both point to the same address.

Thus, if you want to work on a copy and not have any changes you make be done also to the original, you need to make a "deep copy" of the original by adding .copy() to the end of the sentence:

```
# A deep copy of a SlicedFrames
slicedFrames_new = slicedFrames.copy()
```

This will work for the *SlicedFrames*, *SlicedPacsCubes*, and *slicedPacsRebinnedCubes*, as well as for the *Frames*, *PacsCube*, and *PacsRebinnedCube*. It will work in the *SpectralSimpleCubes* also, but not for the *ListContext* that contains these (projected, drizzled, or interpolated) cubes.

The pipeline task selectSlices, which reads a *SlicedXXX* in and writes a *SlicedXXX* out, also creates a fully independent product.

3.2.1.2. Level 0 to 1: not yet fully-reduced data

The science data-products at these lower levels are *Frames* and *PacsCube*s, the HPSFIT[R|B] and HPS3D[R|B] as discussed in Section 2.3. The datasets that make up the *Frames* and *PacsCubes* can be seen by clicking on the + next to any of the HPSFIT[R|B] or HPS3D[B|R] listed in the Data tab of the Observation Viewer,

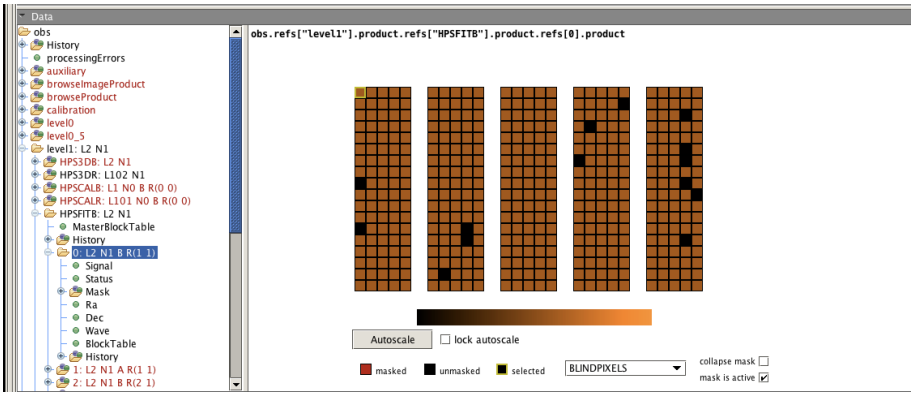


Figure 3.1. The datasets in a Frames (SPG 12.1), and the top part of the Pacs Product Viewer is visible in the Data viewing tab

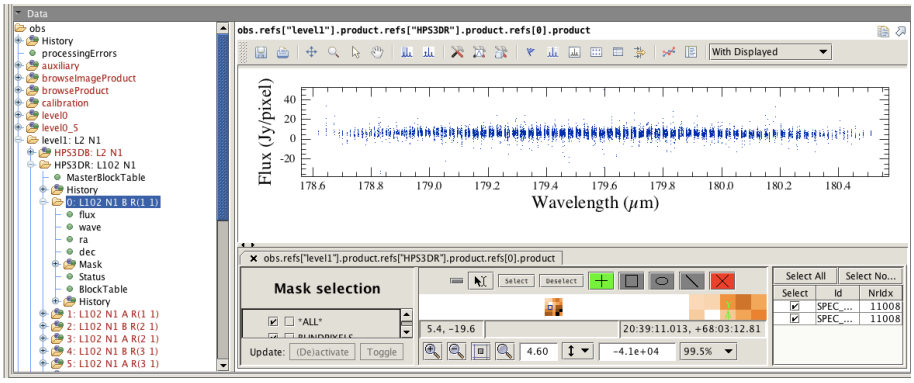


Figure 3.2. The datasets in a PacsCube (SPG 12.1) and the top part of the Spectrum Explorer is visible in the Data viewing tab

- The **3D data-arrays** of signal (*Frames*) or flux (*PacsCube*), wave, ra, dec. You can browse through these by clicking on them in the Data tab
- The **BlockTable** is an organisation table of data-blocks. See Section 3.2.2.3 for more detail.
- The **Status** is a table containing data about where the different parts of the instrument were set to (grating position, nod position, chop position, datapoint number, ra, dec, time, temperatures, velocity...), for each time-point in the observation. These data are used in some of the pipeline tasks, in particular when turning raw data into calibrated data. See Section 3.2.2.1 for more detail.
- The **Masks** contain various masks created by pipeline tasks that flag datapoints as bad for different effects, e.g. saturation. A flag value of 0 means "is not bad" and 1 means "is bad". These masks can be viewed along with the science data using the Pacs Product Viewer, via a right-click on an individual *Frames* or *PacsCube* which contain the masks. See the PDRG (e.g. chp. 9 in *PACS Data Reduction Guide: Spectroscopy*) for more detail.

To inspect one of the *Frames* or *PacsCube*s in a viewer, a single click on the product while in the Observation Viewer will open the default viewer (or double-click to get a list of viewer and action options). For the *Frames* this is the Pacs Product Viewer, for the *PacsCube* it is the Spectrum Explorer. How to use these viewers is explained in the PDRG (chp. 9 in *PACS Data Reduction Guide: Spectroscopy*): the screenshots above show the top view of both of these viewers.

At level 0, 0.5 and 1 there are also *context*s that contain engineering, housekeeping, and auxiliary data. These are used by the pipeline but we do not explain them in more detail here.

3.2.1.3. Level 2 and 2.5: science-quality data

The Level 2 products are the end products for all observing modes except the unchopped range, for which the on-source observation will also have a Level 2.5: the off-source Level 2 subtracted from the on-source Level 2, producing a background-subtracted, end product.

The science data-products of the higher levels are cubes: *PacsCube*, *PacsRebinnedCube*, and three types of *SpectralSimpleCube* called, in the pipeline, projectedCubes, drizzledCubes and interpolatedCubes, and these are contained in the *ObservationContext* in *context*s called HPS3D[R|B] (*SlicedPacsCube*), HPS3DR[R|B] (*SlicedPacsRebinnedCube*), HPS3DP[R|B] (*ListContext* of projected cubes), HPS3DD[R|B] (*ListContext* of drizzled cubes), and HPS3DI[R|B] (*ListContext* of interpolated cubes). The first cube is the same product as found in Level 1, except that for "line scan" observations a flat-fielding has been applied to the Level 2 cubes (this is not done for "range scan" observations). The latter four cubes are science-quality products which are each ideal for different types of observations and sources.

ObservationContexts from SPG 12 contain only the *PacsCube*, *PacsRebinnedCube*, and the projected cubes *(SpectralSimpleCube)*. For SPG 13 they contain the *PacsCube*s and *PacsRebinnedCube*s and also the following:

- drizzled and projected cubes for Nyquist and oversampled mapping line scans;
- interpolated and projected cubes for Nyquist and oversampled mapping range scans;
- projected and interpolated cubes for pointed and undersampled mapping line and range scans.

More detail can be found in Section 2.3.1. One type of cube (depending on the observing mode) is provided additionally with an equidistant wavelength grid (the other cubes have a grid that scales with wavelength), these "equidistant cubes" also being the PACS spectroscopy "standalone browse products" (see Chapter 5 for more detail). In the *ObservationContext* the letters "EQ" are added to the HPS name of the cube *context*.

To see how many cubes are contained within any *ListContext*, or to see the datasets within the cubes, click on the appropriate product in the Observation Viewer Data tab, e.g.,

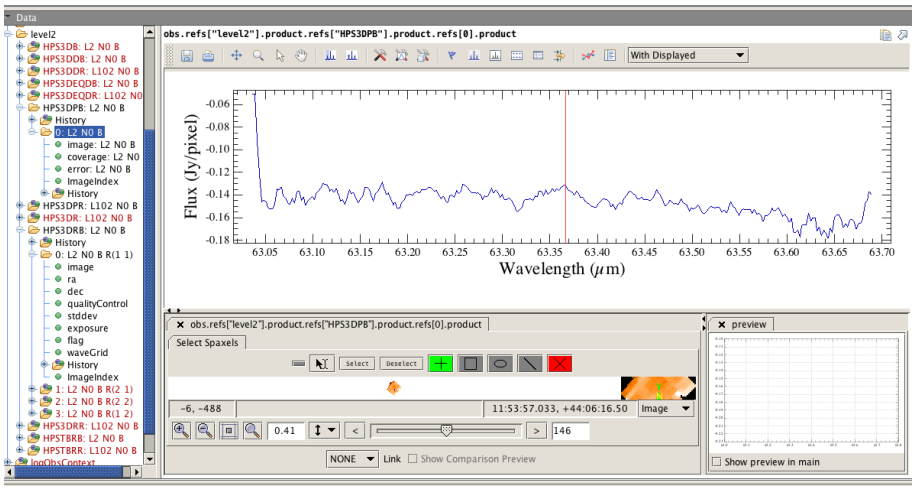


Figure 3.3. The datasets in a Level 2 *PacsRebinnedCube* and a projected cube, taken from an SPG 13 observation. The Spectrum Explorer is open on the cube in Data viewing tab

The screenshot shows the datasets of the rebinned and projected cubes. The datasets of all the cubes include:

- **ImageIndex**: a table of the wavelengths. These are the same for each spaxel. Because the bin sizes scale with wavelength, this spectral information is carried in a dataset rather than via WCS keywords. This is found in the projected, interpolated, and drizzled cubes.
- **Image**: is the flux data (units of Jy/pixel [though actually Jy/spaxel]) for the entire 3D array. This is found in the rebinned, projected, interpolated, and drizzled cubes.
- **Error**: are the errors of the fluxes, as propagated through the pipeline. See the PDRG chp. 6 in *PACS Data Reduction Guide: Spectroscopy* for more information on what is included in these errors. This is found in the projected and drizzled cubes.
- **Coverage**: is a 3D dataset found in the re-mapped cubes: the projected and drizzled cubes, which are created by re-mapping a raster sequence of rebinned cubes into a single cube. The dataset contains one value, per wavelength and per spaxel of the re-mapped cube, that is the sum of the weights of the spaxels of the rebinned cube(s) that contributed to each new spaxel of the re-mapped cube.
- **Flag**: This is found in the rebinned and drizzled cubes. When these cubes are created, the data masked as bad in the Masks found in the *PacsCube*s (which the rebinned and drizzled cubes are both created from) are usually excluded. However some of the Masks are still useful to carry forward: for example, whether you decide to include or exclude saturated data, you may still want to know that those datapoints were flagged as being saturated. The flag data are held as 3D arrays, but inspecting and understanding the entries is not so straightforward. See the PDRG chp. 9.6 in *PACS Data Reduction Guide: Spectroscopy* for a short summary of working with the flags.

- **Stddev** and **exposure**: these datasets are found in the rebinned cubes. The *PacsRebinnedCube*s are created from *PacsCube*s: the spectrum of each spaxel of the rebinned cube is created by combining the multiple spectra contained in each spaxel of the *PacsCube*. The resulting spectra are combined on a wavelength grid created from the wavelength datasets in the *PacsCube*s. The stddev is the scatter in the *PacsCube* datapoints that fall into each wavelength bin of the rebinned cube. The exposure is a measure of the number of times a wavelength bin in the rebinned cube was visited.
- **Ra, dec**: the values of RA and Dec for each spaxel and wavelength. These are found in the rebinned cubes—in the other cubes the spatial grid is regular and the sky coordinates are found in each cube's WCS.
- **QualityControl**: to be ignored.
- **Weight**: This is found in some equidistant cubes. If there is an error array in the cube that the equidistant cube is made from, then an error is created for the equidistant cube, otherwise a blank array is created.

To view a cube with the Spectrum Explorer within the Observation Viewer (in the space to the right of the Data tab), single click on the cube. To open the Spectrum Explorer in a larger tab, instead double-click on the cube (or right-click and select the viewer).

In addition to the cubes, in SPG 13 there is a new *context* called HPSTBR[R|B]. This contains the data of the rebinned cubes in a tabular format. This product is also one of the standalone browse products, and is explained in more detail in Chapter 5.

3.2.1.4. A table of the datasets in the science products

The names of the products and their datasets found at all the levels of the pipeline are listed, in alphabetical order, in the table below. In the following sections we show how to navigate and view the various products.

Table 3.1. The contents of PACS pipeline products from Level 1 and 2/2.5

Name	What it is / task that made it	What product it is found in
BlockTable	organisation table of data blocks / findBlocks and added to by a few other tasks	*Frames, PacsCube*
coverage	a measure of the amount of data from the input *PacsRebinnedCube*s that became data of the *SpectralSimpleCube*, one value per wavelength and per spaxel / specProject and drizzle	*SpectralSimpleCube*s
error	values of the flux error (Jy/pixel), propagated from the stddev dataset of the input *PacsRebinnedCube* and modified by the specProject	*SpectralSimpleCube*
error	values of the flux error (Jy/pixel) based on the data in the input *PacsCube* and modified by drizzle	*SpectralSimpleCube*
exposure	a measure of the amount of data from the input *PacsCube* that became data of the output *PacsRebinnedCube*, one value per wavelength bin and per and spaxel / specWaveRebin	*PacsRebinnedCube*

flag	an HCSS-compliant array that contains values based on the Masks of the *PacsCube*s that created the *PacsRebinnedCube*—included in this "flag" are the following masks: SATURATION, RAWSATURATION, NOISYPIXELS, BLINDPIXELS, INVALID, and UNVALID (but ignore this one); for information on the state of these masks when the *PacsRebinnedCube* was created, see the Meta data attached to the "flag" dataset / specWaveRebin	*PacsRebinnedCube*
flag	an HCSS-compliant array that contains values based on the Masks of the *PacsCube*s that created this *SpectralSimpeCube* via the pipeline task drizzle—included in this "flag" are only the following masks: NOD; to see the status of these Masks at the time this *SpectralSimpleCube* was created see the Meta data attached to the flag dataset / drizzle	*SpectralSimpleCube*
flux	the flux dataset in units of Jy/pixel (where a pixel is actually a spaxel) / specFrames2PacsCube	*PacsCube*
image	the flux dataset in Jy/pixel (where a pixel is actually a spaxel) / specWaveRebin	*PacsRebinnedCube, SpectralSimpleCube*s created by specProject, drizzle and specInterpolate
ImageIndex	the wavelength dataset in micrometres (created from the "wave" and "waveGrid" datasets) / specWaveRebin	*PacsRebinnedCube, SpectralSimpleCubes* created by specProject, drizzle and specInterpolate
Mask	contains all the individual masks, 0 and 1 values for *is not bad* and *is bad* data for different instrumental or data effects / present from the Level0 and added to as the pipeline proceeds	*Frames, PacsCube*
qualityControl	to be ignored	*PacsRebinnedCube*
ra, dec	RA and Dec datasets / specAssignRaDec	*Frames, PacsCube, PacsRebinnedCube*
signal	signal dataset, in units of counts / already there at Level0 but changed by specConvDigit2VoltsPerSecFrames and rsrfCal	*Frames*

Status	a table of the status of PACS during the entire observation / there from the Level0 and added to as pipeline proceeds	*Frames*, *PacsCube*
stddev	the standard deviation dataset (not an error dataset, but a measure of the scatter in the spectrum) / specWaveRebin	*PacsRebinnedCube*
wave	wavelength dataset in micrometres / waveCalc	*Frames*, *PacsCube*
waveGrid	the wavelength grid dataset created by wavelengthGrid and used to create the *PacsRebinnedCube* / specWaveRebin	*PacsRebinnedCube*
weight	found only in the equidistant cubes where no error dataset is present in input cube (and then the array contain only 0s) / specRegridWavelength	*SpectralSimpleCube*s which are the equidistant interpolated cubes

3.2.2. Using the Status and BlockTables to perform selections on data

Most astronomers will not need to interact with the Status or BlockTable. However, these two can be used to select data chunks out of an observation, for example to work with or plot, and advanced users may wish to do so. Here we outline the methods that can be used to do this, starting by listing the columns of information that the Status and BlockTables contain.

3.2.2.1. The Status Table

The Status table generally has the same length as that of the readout/time axis of the flux dataset in a *Frames* or *PacsCube*, so for each astronomy datapoint, each column of the Status will have an entry. Most, but not all, of the Status entries are a single value, others are arrays.

The Status changes as the pipeline processing proceeds, it is added to as information is calculated (e.g. RA and Dec, spacecraft velocity) or modified (e.g. when the length of the *Frames* is reduced after the pipeline task specDiffCs), and information in the Status is used by other pipeline tasks.

To inspect the Status you can use the Dataset viewer or one of the plotters (access via the usual right-click on "Status" when you view your product via the Observation Viewer). The entries in the Status table are of mixed type—integer, double, boolean, and string. The ones that are plain numbers can be viewed with the Table/OverPlotter, the others you need to look at with the Dataset Viewer. (Note: over-plotting to see clearly the entries that have very different Y-ranges can be a bit difficult.)

Most of the Status columns are given in this table. For those not listed, hovering the mouse over the column label will bring up a tooltip about that column.

Table 3.2. Status information

Name	**(S)pec/(P)hot,Type,Dim,Unit**	**Description**
AbPosId	S+P Bool 1	on and off raster nod information (0=A, 1=B)
AcmsMode	S+P String 1	Attitude Control and Measurement System mode
AngularVelocityX/Y/Z|Error	S+P Double 1 arcsec/sec	Angular velocity along the spacecraft axes

Aperture	S+P String 1	instrument aperture used
BAND	S+P String 1	band
BBID	S+P Long 1	Building block identifier
BBSEQCNT	S+P Int 1	building block sequence counter
BBTYPE	S+P Int 1	building block type
BLOCKIDX	S+P Int 1	a reference to the BlockTable: value taken therefrom
BSID	P Int 1	a 1 byte field (1 word used in the packet) containing instrumental information of the bolometers
CALSOURCE	S+P Int 1	calibration source number (0=not, 1 or 2)
CAPA	S Double 2 pF	Electric capacitance
CHOPFPUANGLE	S+P Double 1 degree	chopper position angle wrt the optical axis of the focal plane unit
CHOPPER	S Int 1	combination of CHOPPER-PLATEAU and CALSOURCE
CHOPPOS	S String 1	the chopper position (e.g. +/-throw)
CHOPSKYANGLE	S+P Double 1 arcmin	chopper position angle wrt as an angle on the sky
CPR	S+P Int 1	Chopper position as encoded by the MEC
CHOPPERPLATEAU	S+P Int 1	a number identifying that the chopper is in one of its set-positions
CRDC	S Int 1	Number of readouts since the last SET_TIME command to the DMC—used with TMP1 and 2 to calculate the FINETIME. Current ReadOut Count: current values of the readouts counter, starting from Nr and decreasing, a value of 0 signals a destructive readout and the end of an integration interval.
CRECR	S Int 1	CRDC number of readouts since the last SET_TIME command to the DMC.
CRDC	P Int 1	5x the number of OBT (on board) clock ticks since the last SET_TIME command to the DMC—used with TMP1 and 2 to calculate the FINETIME
CRDCCP	P Int 1	Current Readout Count in Chopper Position. This counter is reset each time the chopper starts moving

DBID	P	Data Block ID: of the block of detector arrays whose data are included in this packet (data in this timestamp).
DecArray/Err	S+P Double 1	Dec of the boresight (centre of the detector), from pointing product and FINETIME
DMCSEQACTIVE	S+P Int 1	DecMec sequence active? 0=no 1=yes
DP_WHICH_OBCP	S+P Int 1	On Board Control Procedure
DXID	S+P Int 1	Detector selection table identifier (this is a table which is used to select which detector pixels to read out)
FINETIME	S+P Long 1	Time in units of microsecond. Atomic time (SI seconds) elapsed since the TAI epoch of 1 Jan. 1958 UT2.
GRATSCAN	S Int 1	counter of grating scans
GPR	S Int 1	grating position an encoded by the MEC
GratingCycle	S Int 1	the index within the grating cycle, within the lABBA or ABAB chopping cycle
IndexInCycle	S Int 1	the index within the ABBA or ABAB chopping cycle
IsAPosition/IsBPosition	S+P Bool 1	at B or A nod position>
IsConstantVelocity	S+P Bool 1	is the satellite velocity constant?
IsOutOfField	S+P Bool 1	out of field?
IsOffPos	S+P Bool 1	off position flag
IsSerendipity	S+P Bool 1	serendipity mode; was the instrument active during e.g. slew periods
IsSlew	S+P Bool 1	slew of the satellite
LBL	S+P Int 1	Label
Mode	S+P String 1	observing mode
NodCycleNum	S+P Long 1	pointing nod cycle number
NrReadouts	Int 1	number of readouts that were used for the on-board-reduced value
OBSID	S+P Long 1	Observation identifier
ON/OFFSOURCE	S Bool 1	whether on or off source
OFF_RESETIDX	S Int 1	reset indices of the used off-source frames
OnTarget	S+P Bool 1	on target flag
PaArray/Err	S+P Double 1	Position angle of the boresight (centre of the detector), from

		pointing product and FINE-TIME
PIX	S+P Int 1	Counter for synchronisation to SPU housekeeping
PIXEL	S+P Int 1	the detector pixel number
RaArray/Err	S+P Double 1	RA of the boresight (centre of the detector), from pointing product and FINETIME
RasterLine/ColumnNum	S+P Long 1	raster line and column number
RAWSAT	Boolean	value of the rawsaturation mask (which contains values for one spectral pixel of the PACS detector only)
Repetition	S+P Int 1	repetition number
RESETINDEX	S+P Int 1	a counter of resets
RCX	S+P Int 1	(detector) Raw Channel Index
RollArray	S+P	obsolete; see PaArray
RRR	S Int 1	Readouts in Ramp (number)
SCANDIR	S Int 1	grating scanning direction
ScanLineNumber	S+P Long 1	scan line number
TMP1	S+P Int 1	Timing parameter: "time stamp" of the SET_TIME to the DMC, set only at the beginning of the observation. Unit of seconds
TMP2	S+P Int 1	see TMP1. In 1/65536 fractional seconds
Utc	S+P Long 1	UTC (not used during pipeline processing but is set by pipeline task)
VLD	S+P Int 1	Science data is value (0xff) or is not (0x00)
VSC	S Double 1	the spacecraft velocity to correct the wavelengths; negative is moving towards the source
WASWITCH	S Bool	wavelength switching mode or not
WPR	S+P Int 1	Wheel position encoded by the MEC

3.2.2.2. Selecting discrete data-chunks using the Status

Here we show a few ways to work with Status data, including using it to select on the astronomical data (the fluxes). **Note**: The intension is to show the methods that can be used to work with these data: the code itself will only produce a plot at the end for a Frames taken from Level 0.5 (from Level 1 there is no "off" chop data), for a chop-nod AOT (not for wavelength switching or unchopped range scans, which do not have "on" and "off" chop within the same observation), and for a "large" throw (other choices are "small" and "medium").

```
# Preamble: to test the following on an example observation
obs = getObservation(1342238131, useHsa=1)
myframe = obs1.level0_5.red.fitted.product.refs[1].product
# a red frames, level 0.5 -> change as you wish
```

```
# What are the dimensions of a Status column in a Frames
print myframe.getStatus("RaArray").dimensions

# Two ways to read Status data
# what are all the "BAND"s in a Frames
print UNIQ(myframe.status["BAND"].data)
print UNIQ(myframe.getStatus("BAND"))

# Extract the Status column called CHOPPOS from a Frames
# You can do the same for other columns
# Result will be a Double1d, Int1d, or String1d, in most cases
stat = myframe.getStatus("CHOPPOS")

# Extract out certain data using Status values
# get the on chops: here the throw is "large";
# "small" and "median" are also values
on = (stat == "+large") # a boolean, true where +large
on = on.where(on) # turn that into a Selection array
# get the off chops
off = (stat == "-large")
off = off.where(off)
wave = myframe.getWave(8,12)
offwave = wave[off]
# To check this, you could plot the selected data
p=PlotXY(myframe.getWave(8,12)[on],myframe.getSignal(8,12)[on])
p.addLayer(LayerXY(myframe.getWave(8,12)[off],myframe.getSignal(8,12)[off]))
```

More scripts following the example here can be found in the PDRG in *PACS Data Reduction Guide: Spectroscopy*. You can also look up the *PacsStatus*, *Frames* and *PacsCubes* classes in the *PACS DRM* to see what (java) methods they have. And to learn more about scripting in HIPE, we refer you to the SG.

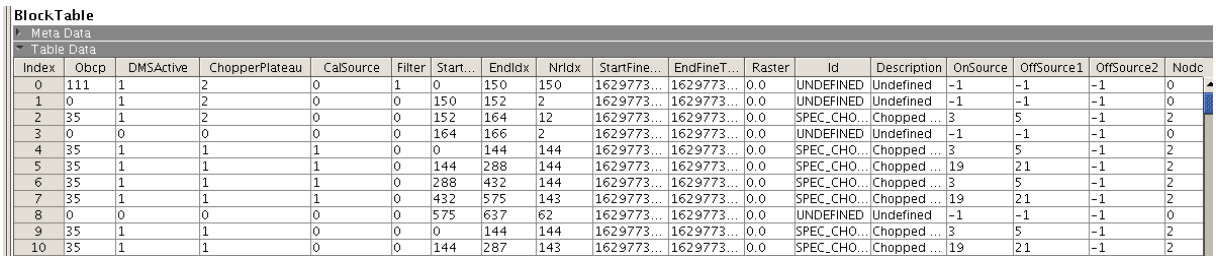
3.2.2.3. BlockTable and MasterBlockTable

The BlockTable is a listing of the data blocks of your *Frames* and *PacsCube*s. In the spectroscopy pipeline we slice our data according to a (mainly) astronomical logic (pointing, wavelength, etc.), therefore, as well as the BlockTables of the individual slices of the *SlicedFrames* or *SlicedPacsCube*s, there is a MasterBlockTable that is a concatenation of the individual BlockTables of all the slices.

3.2.2.4. Selecting discrete data-chunks using the block table

Blocks are chunks of data grouped following an instrument and/or astronomical logic: for example two grating scans will be two blocks, and two pointings in a raster will be two blocks. The data themselves are not organised in this block structure, rather the BlockTable contains pointers to the data so that tasks can identify where the blocks are by querying with the BlockTable. It also gives a handy overview of your observation. Blocks are created in and used by the pipeline.

To find the BlockTable to look at it, click on the "+" next to a *Frames* or *PacsCube* listed in the Observation Viewer. To see the MasterBlockTable for a *SlicedFrames*, click on the + next to the HPSFIT[R|B] in the Observation Viewer. Open the Dataset inspector (right-click on the BlockTable to access the viewer via the menu) to see something similar to this screenshot:


BlockTable
Meta Data
Table Data

Index	Obcp	DMSActive	ChopperPlateau	CalSource	Filter	Start...	EndIdx	NrIdx	StartFine...	EndFineT...	Raster	Id	Description	OnSource	OffSource1	OffSource2	Nodc
0	111	1	2	0	1	0	150	150	1629773...	1629773...	0.0	UNDEFINED	Undefined	-1	-1	-1	0
1	0	1	2	0	0	150	152	2	1629773...	1629773...	0.0	UNDEFINED	Undefined	-1	-1	-1	0
2	35	1	2	0	0	152	164	12	1629773...	1629773...	0.0	SPEC_CHO...	Chopped ...	3	5	-1	2
3	0	0	0	0	0	164	166	2	1629773...	1629773...	0.0	UNDEFINED	Undefined	-1	-1	-1	0
4	35	1	1	1	0	0	144	144	1629773...	1629773...	0.0	SPEC_CHO...	Chopped ...	3	5	-1	2
5	35	1	1	1	0	144	288	144	1629773...	1629773...	0.0	SPEC_CHO...	Chopped ...	19	21	-1	2
6	35	1	1	1	0	288	432	144	1629773...	1629773...	0.0	SPEC_CHO...	Chopped ...	3	5	-1	2
7	35	1	1	1	0	432	575	143	1629773...	1629773...	0.0	SPEC_CHO...	Chopped ...	19	21	-1	2
8	0	0	0	0	0	575	637	62	1629773...	1629773...	0.0	UNDEFINED	Undefined	-1	-1	-1	0
9	35	1	1	0	0	0	144	144	1629773...	1629773...	0.0	SPEC_CHO...	Chopped ...	3	5	-1	2
10	35	1	1	0	0	144	287	143	1629773...	1629773...	0.0	SPEC_CHO...	Chopped ...	19	21	-1	2

Figure 3.4. The MasterBlockTable

The entries that identify nods, grating scan (direction), raster pointing, wavelength range, etc. can be found in the table below. Many of the BlockTable columns come directly from the Status, and so can be similarly understood. If you hover with the cursor over the column titles, a banner explanation will pop up. The BlockTable entries are:

Table 3.3. BlockTable columns

Name	Description
Band	spectral band
CalSource	0,1,2 for neither, calsource 1, calsource 2
ChopperPlateau	taken from the Status table
Description	explanation of keyword
DMSActive	DecMec sequence active? 0=no 1=yes
Filter	filter
FramesNo	slice number, the *Frames* number in the *Sliced-Frames* that this block comes from
GenericOnOffSource	0=not science, 1=on, 2=off, -1=not defined (i.e. the block has chopping, so both on and off are included)
GPRMin/Max	smallest and largest grating position
Id	keyword describing this block
index	a simple counter
IsOutOfField	true=out of field
LineDescription	line description taken from the Meta data
LineId	line identification number (a simple counter)
Min/MaxWave	smallest and largest wavelength
NodCycleNum	nod cycle number
NoddingPosition	0=none, 1=A, 2=B
NrIdx	number of indices in this block
Obcp	on board control procedure number
OffSource1/2	first and second off-source position label (some AOTs have 2 off positions)
OnSource	on-source position label
Raster	raster number
RasterColumn/LineNum	raster column and line numbers
Repetition	repetition cycle number (used in photometry)
ResLen	reset length (number of resets that fed into each readout)
ScanDir	scanning direction (0,1)
Start/EndFineTime	start and end FINETIME of this block
Start/EndIdx	starting/end index of this block
WaSwitch	wavelength switching active or not

In the MasterBlockTable, be aware that, per block, the StartIdx and EndIdx entries are relative to the slice that is that block. The entry that identifies the slice number is "FramesNo".

To use the [Master]BlockTable to query your data you can use the following examples

```
# Preamble: to test the following on an example observation
```

```
obs=getObservation(1342209728, useHsa=1)
slicedFrames = obs.level0_5.red.fitted.product
myframe = slicedFrames.refs[1].product
slicedCubes = obs.level1.red.cube.product # red cubes, level 1
mycube = slicedCubes.refs[0].product


# Get listing of all columns of the BlockTable of a Frames
# and the MasterBlockTable of a SlicedFrames
print myframe.blockTable
print slicedFrames.masterBlockTable
print mycube.blockTable
print slicedCubes.masterBlockTable


# Select the blocks "SPEC_UPSCAN_ALL" and "SPEC_DOWNSCAN_ALL"
# and create two new Frames from them
# This command will overwrite "myframe" so copy it first
myframe_cp = myframe.copy()
myframe_up=myframe.select(myframe.getBlockSelection("SPEC_UPSCAN_ALL"))
myframe = myframe_cp.copy()
myframe_down=myframe.select(myframe.getBlockSelection("SPEC_DOWNSCAN_ALL"))

mycube_cp = mycube.copy()
mycube_up=mycube.select(mycube.getBlockSelection("SPEC_UPSCAN_ALL"))
mycube = mycube_cp.copy()
mycube_down=mycube.select(mycube.getBlockSelection("SPEC_DOWNSCAN_ALL"))

# To be sure this has worked, plot the data from one
# detector pixel
p=PlotXY(myframe_up.getWave(8,12), myframe_up.getSignal(8,12),line=0)
p.addLayer(LayerXY(myframe_down.getWave(8,12), myframe_down.getSignal(8,12),line=0))
p[0].setName("up")
p[1].setName("down")
p.getLegend().setVisible(True)

p=PlotXY(mycube_up.getWave(2,2), mycube_up.getFlux(2,2),line=0)
p.addLayer(LayerXY(mycube_down.getWave(2,2), mycube_down.getFlux(2,2),line=0))
p[0].setName("up")
p[1].setName("down")
p.getLegend().setVisible(True)


# The easiest way to select from a SlicedFrames, a slicedPacsCubes
# or SlicedPacsRebinnedCubes, i.e. from an entire observation, not just
# working on one slice at a time, is with the pipeline helper task
# "selectSlices". This is taken from the pipeline scripts: to select
# on any of these parameters, enter the selection keywords in the relevant
# []. The output of this task is another SlicedFrames|PacsCubes.
verbose     = 1
lineId      = []
wavelength  = []
rasterLine  = []
rasterCol   = []
nodPosition = ""
nodCycle    = []
band        = ""
scical      = ""
sliceNumber = []
sCubes = selectSlices(slicedCubes, lineId=lineId, wavelength=wavelength,\
         rasterLine=rasterLine,\
         rasterCol=rasterCol, nodPosition=nodPosition, nodCycle=nodCycle,\
         band=band, scical=scical,\
         sliceNumber=sliceNumber, verbose=verbose


# A listing of the PacsCubes (or Frames) in a slicedCubes or slicedFrames
slicedSummary(slicedCubes)


# More general examples for querying a SlicedCubes(or Frames) to identify
# which slices satisfy a condition, and then select them out.
```

```
#
# Get the raster column numbers from a SlicedFrames
nodSlicesCol = slicedCubes.getMasterBlockTable().getRasterColumnNumber()
framesNo = slicedCubes.getMasterBlockTable().getFrameNumbers()
# select on it, e.g. you want the slices from raster column 2:
select = UNIQ(framesNo.get(nodSlicesCol == 2))
# Now select out the slices that have this column number and place the
# in a new SlicedfCubes
select = select.getArray()
sCubes = selectSlices(slicedCubes,sliceNumber=select)
```

You can look up the *Frames* and *PacsCubes* classes in the *PACS DRM* to see the other methods that work with these classes. Bear in mind that since the data at Level 0.5 onwards is already sliced by raster position, wavelength, etc., within any one slice you cannot usefully select for these aspects. However, similar methods to those given here will work on a *SlicedFrames* or *SlicedPacsCubes*, allowing you to select on the entire observation. To learn the syntax of these methods, look up these classes in the *PACS DRM* to see what methods are provided for them. And if you completely extract the [Master]BlockTable from a *[Sliced]Frames* (e.g. HIPE>status=myframe.getBlockTable()), the *PACS DRM* entry for the its class (*ObcpBlockTable* or *MasterBlockTable*) can be consulted to see what methods you can use on that. Finally, to learn more about scripting in HIPE, we refer you to the SG.

3.3. Photometry

3.3.1. The science products

3.3.1.1. Level 0 to 1: not yet fully-reduced data

The photometry pipeline products are *Frames* and *Simpleimage*s. The PACS pipeline products, as you would have seen in some of the screenshots in 2.3.2, are multi-layered, these variously including: a BlockTable, Meta data, a Status table, Masks in *PACS Data Reduction Guide: Photometry*, and various datasets that have been produced by the pipeline.

Table 3.4. The contents of PACS pipeline products

Name	What it is / task that made it	What product it is found in
signal	the flux dataset in units of digital units/pixel	*Frames*
Mask (PDRG in *PACS Data Reduction Guide: Photometry*)	contains all the individual masks / there from the Level0 and added to as pipeline proceeds	*Frames*
BlockTable	organisation table of data blocks / created by findBlocks and added to by a few other tasks	*Frames*
Status	a table of the status of PACS during the entire observation / there from the Level0 and added to as pipeline proceeds	*Frames*
image	the flux dataset in Jy/pixel	*SimpleImage*
coverage	a measure of the number of readouts that were used to assign flux to a spatial map pixel. In mappers using photProject, the general level of the coveage values strongly depends on the parameter pixfrac in the task.	*SimpleImage*

error (PDRG in *PACS Data Reduction Guide: Photometry*)	the error dataset	*SimpleImage*
stDev	the standard deviation of the projected pixels in JScanam and HPF maps	*SimpleImage* (only JScanam and HPF)
artifactmap	MADmap processing produce some artifacts on bright sources that are removed in the corrected map (HPPCORMAP)	*SimpleImage* (only for HPPPCORMAP)
HPF Mask	the mask used for high-pass filtering	*SimpleImage* (only for HPF)
naive (PDRG in *PACS Data Reduction Guide: Photometry*)	the naive map of UNIMAP	*SimpleImage* (only for UNIMAP)
gls (PDRG in *PACS Data Reduction Guide: Photometry*)	the map processed with UNIMAP (still contains gls artifacts)	*SimpleImage* (only for UNIMAP)
pgls (PDRG in *PACS Data Reduction Guide: Photometry*)	the final map after removing gls artifacts in UNIMAP	*SimpleImage* (only for UNIMAP)

The methods you will use to access these parts of these products can be found in their *PACS DRM* entries.

3.3.1.2. Level 2 and 2.5: science-quality data

The photometry pipeline product from **level2** are *Maps*. These maps contain the image in Jy/pixel, the coverage, some measure of error plus several mapmaker-related auxiliary images (e.g. the artifact map of MADmap). There are two ways to access these layers. The easy way is to right-click on the desired product in the Observation viewer and select *Open with#Image Viewer for ArrayDatasets*. The other way is to use the Display tool from the command line:

```
Display(obs.refs["level2_5"].product.refs["HPPCORMAPB"].product["error"].data)
```

here you need to type in the exact reference to your product. The syntax for this can be taken from Name entry in the table above (e.g. "error" in the example), the level you want to take it from, and the *context* name from the table above (e.g. HPPCORMAPB in the example).

If your observation is of point sources, e.g. as will usually be the case with mini-map observations of individual stars, the maps at **level2** are of science quality, however it is recommended to use the **level2.5** if you can, since these maps have a better S/N (being a combination). For extended sources or large maps, the **level2** products should not be used for science.

Chapter 4. The Meta Data and FITS Keywords

4.1. Introduction

An *ObservationContext* has Meta data that give information about the observation: details taken from the observation planning tool (HSPOT), coordinates, the instrument configuration, some processing information, wavelength information, dates, etc. Each *context* and each product that is contained within an *ObservationContext* also has Meta data, some of which is copied from (or to) that of the *ObservationContext*, and other of which is specific to that *context* or product. For products which can be exported to FITS files—maps, cubes, *TableDatasets*—the FITS keywords are created from the Meta data. When these FITS files are read back in (whether Herschel data or that from another observatory) the FITS keywords become Meta data. HCSS and each instrument has a dictionary that links the Meta data to the FITS keywords: any Meta datum that is not in that dictionary is translated as META_## in the FITS header.

In this chapter we list the Meta data and FITS keywords that are most likely to be encountered by the astronomer. We list these in alphabetical order for the *ObservationContext*, the Levels in the *ObservationContext*, and for the science products that astronomers will use, for both photometry and spectroscopy.

4.2. Where are the Meta data?

The Meta data of the layers in the *ObservationContext*, in *SlicedFrames*, *Frames*, or any other object in an observation, will not be the same. Some are repeated, some are not. To see the Meta data of any particular product, open the observation with the Observation Viewer (Chapter 2), the Meta data panel is located at the top:

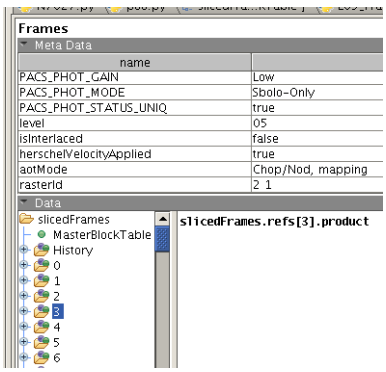


Figure 4.1. Meta data in the Observation viewer

You can also type Meta data out on the console, for example to see some of the Meta data added by the pipeline,

```
# On the individual slices (i.e. Frames)
print myframe.meta["nodPosition"]
print myframe.meta["lineId"]
print myframe.meta["rasterId"]
print myframe.meta["lineDescription"]
print myframe.meta["band"]
# Or also
print myframe.meta["band"].string
# Or, on slice 1 of the entire SlicedFrames (or Cubes)
print slicedFrames.getScience(0).meta["lineId"]
```

4.3. Meta data and FITS keyword tables

Here we list the entries of the more useful (for the astronomer) of the Meta data; the technical instrument and satellite Meta data are not includes. Meta data have a name, a value, and a description: the first and last are tabulated here. Some Meta data are also explained in the *PDD*. Some Meta data have the description "HSPOT": this is information that was entered into the AOR by the observer, rather than information gathered during the observing. The note "SLICE-INFO" indicates information related to the slicing of the product.

4.3.1. The *ObservationContext*

For photometry and spectroscopy

Table 4.1. Meta Data of an ObservationContext

Name	Description	FITS header keyword
algo number	internal: related to algorithm	ALGONUM
algorithm	indicates whether the data are raw ramps, averaged ramps (Floating average: 4), or fitted ramps (least squares fit)	ALGORITHM
aorLabel	name given to the AOR submitted by the astronomer	AOR
aot	astronomer observing template name	AOT
aotMode	aot mode, concatenation of other AOT mode Meta data	AOTMODE
band	filter band chosen	BAND
bandUsedForRaDec	band used in PhotAssignRaDec	BNDRADEC
BLINDPIXELS	in the flag dataset of a cube: either a T/F indicating whether this cube was created including data with the BLINDPIXELS flag, or a number indicating how many input data-points so affected were included	no keyword yet
blue	filter used for the blue camera (phot)	BLUEBAND
bluWave	HSPOT: blue wavelength range limit(s)	BLUEWAVE
BOL_VD_B_1/2/3/4	bias for the blue detector group 1/2/3/4	BOLVDB1/2/3/4
BOL_VD_R_1/2	bias for the red detector, group 1	BOLVDR1/2/3/4
calBlock	false or true for whether this product is the calBlock	CALBLKID
calTreeVersion	The calibration tree version used	CALTREE
calVersion	The calibration tree versionused	CALVERS
camera/camName/camera_signature	camera name, for this product	CAMERA/CAMNAME/CAMSIG
capacitance	detector integration capacitance setting	CAPACIT

chopAvoidFrom/To	HSPOT entry: chop avoidance zone	CHPAVFRM/CHPAVTO
chopNod	HSPOT entry: true or false for chop-nod mode	CHOPNOD
Chopper Throw	the chopper throw length	CHPTHROW
chunkScanLegs	chunkScanLegs used when making tod (MADMap)	no keyword yet
compMode/Number/Version	internal: compression mode details	COMPMODE/COMPNUM/ COMPVERS
ConcatenatedOBSIDS	list of obsids concatenated, part of the same observation (e.g. off-source and on-source observation sequence)	OBSIDLST
contains[B2A|B2B|B3A]Data	T/F: at least one line or range has been observed in filter B and B2A/B2B/B3A	B2ADATA/B2BDATA/ B3ADATA
creationDate	date this particular product was created	DATE
creator	The name of the software that created the product	CREATOR
cusMode	internal: common uplink system mode	CUSMODE
dec/decNominal	Dec of satellite/Dec requested	DEC/DEC_NOM
decOff/decOffArcmins	HSPOT entry: Dec of the requested off position for the unchopped spectroscopy mode	DECOFF
decPosDegrees	actual Dec of the off position in degrees	DECOFFDG
density	wavelength sample density requested in HSPOT	WAVEDENS
description	brief description of this product	DESC
detRow/Col	number of detector rows and columns	DETROW/DETCOL
DIM1	the number of measures per Status parameter	STATUSDIM
Duration	of the observation in seconds	No keyword yet
dxid	internal: detector selection (table) identifier	DXID
endDate/Time	end date and time of the observation	DATE-END/TIME-END
equinox	equinox for coordinates	EQUINOX
faintLines	faint (default) or bright line mode	FAINTLIN
fileName	FITS file name associated with this product	FILENAME
Filter	filter for this slice (for sliced products in spectroscopy pipeline)	no keyword yet

fixedMinBinSize	taken from the task wavelengthGrid when an equidistant grid is requested: the equidistant grid's bins sizes were set to a user-requested size	WFIXBIN
flux[Ext|Pnt][Blu|Red]	flux estimate for the source (given in HSPOT)	FLUXEXT[B|R], FLUXPNT[B|R]
fluxUnit	HSPOT entry: unit of the flux for lines	FLXUNIT
formatVersion	internal: version of this product format	FORMATV
fracMinBinSize	taken from the task wavelengthGrid when an equidistant grid is requested: the equidistant grid's bins size are this fraction of the smallest bin size of the standard grid	WFRACBIN
GenericAotName	generic AOT name used when creating the AOR	AOTNAME
glitchTreatment	This parameter describes how the Second Level Deglitch Task treated the glitches	GLTCHALG
gratScan	internal: if true then you have unchopped (spectroscopy) mode	UNCHOPD
herschelVelocityApplied	was Herschel's velocity corrected for (in the pipeline)?	HERVELAP
hpfRad	high-pass filter radius	HPFRAD
infoSpecLowSampling	information flag: if True, more than 20% of spectral bins in this cube were created with fewer than 6 input datapoints (normally only those at the ends of the spectrum)	ISLWSAMP
infoPhot[Blue|Red]GlitchRate/ infoSpec[Blue|Red]GlitchRate	information flag: percentage of glitched datapoints for a blue photometry, red photometry, blue spectroscopy, red spectroscopy product	IBPSCGLT/IRPSCGLT/ IBSSCGLT/IRSSCGLT
infoPhot[Blue|Red]NotFlaggedBad/ infoSpec[Blue|Red]NotFlaggedBad	information flag: percentage of good data-points in this blue photometry, red photometry, blue spectroscopy, red spectroscopy product	IBPVALPX/IRPVALPX/ IBSVALPX/IRSVALPX
infoPhot[Blue|Red]SaturationRate/ infoSpec[Blue|Red]SaturationRate	information flag: saturation rate for this blue photometry, red photometry, blue spectroscopy, red spectroscopy product	IBPVALPX/IRPVALPX/ IBSVALPX/IRSVALPX
infoSpecLowSampling	True if more than 20% of spectral bins have less than 6 valid data-points from the *PacsCube* when creating the *PacsRebinnedCube* with the pipeline	ISLWSAMP

	task specWaveRebin (previously was "qflag" meta datum)	
instMode	instrument mode (e.g. PacsLineSpec)	INSTMODE
instrument	instrument used (here, PACS)	INSTRUME
INVALID	in the flag dataset of a cube: either a T/F indicating whether this cube was created including data with the INVALID flag, or a number indicating how many input data-points so affected were included	no keyword yet
isInLeak	does this slice contain spectroscopy data in the filter leak region (patially, fully, no), in the prime channel (that which the observed requested, i.e. not considering the free, "parallel" channel	ISINLEAK
isInterlaced	a mode of operation for the satellite's StarTracker	ISINTERL
isOffPosition	off or on position slice?	ISOFFPOS
isPrime	is this slice in the prime channel (that which the observer requested, not the free, "parallel" channel)?	ISPRIME
instrumentConfiguration	internal: instrument configuration (will be "FM"=flight model)	INSTCONF
jsGalac	galactic option used in JScanam	JSGALAC
lcontFlux	HSPOT entry: estimated continuum flux	LCONTLFX
level	pipeline processing Level of this product)	LEVEL
lineFlux/lineWidth/lineId/ lineDescription	HSPOT entries: the input line fluxes, width, id, and description	LINEFLX/LINEWID/LINEID/ LINEINFO
lines	HSPOT entry: the spectral lines chosen to observe	LINES
lineStep	raster line step in arcsec; meaningless for pointed observations	RASSTEPL
Line 1,2..	The wavelength range(s) requested for line scan AORs	LINE1/2...
lWave	HSPOT entry: line wavelength set somewhere in HSPOT	LINECENT
m	number of raster columns (a.k.a points) (taken from HSPOT); meaningless for pointed observations	no keyword yet
madPixSc	MADmap optional parameter: pixel scale	MADPIXSC
mapGratScanOffRep	HSPOT entry: for pointed or mapping observations: how of-	OFFREP

	ten the off position is repeated (m=2 means off-on-on-off)	
mapper	Mapper used to generate this product	MAPPER
mapRasterAngle/ mapRasterRefFrame	HSPOT entry: requested map angle wrt the instrument or the sky, and the reference frame (wrt North or wrt the boresight)	RASANGLE/RASREF
mapScanAngle	scan angle for the map	SCANGLE
mapScanAngleRef	reference frame of the scan angle	SCANAREF
mapScanConstrFrom/To	constraint angle of the scan angle (taken from HSPOT)	SCANOFR
mapScanCrossScan	separation of scan legs (taken from HSPOT)	SCANCRSC
mapScanHomCoverage	whether or not the map was taken in homogeneous coverage mode	SCANHC
mapScanLegLength	length of the scan	SCANLEGL
mapScanNumLegs	number of scan legs	SCANLEGN
mapScanSpeed	scanning speed	SCANSP
mapScanSquare	whether or not the map is square	SCANSQ
maskDimension	the number of dimensions of the masks in a PacsCube	MASKDIM
max/min/meanWave	max, min and mean wavelength in the slice or observation	no keyword yet/no keyword yet/ MEANWAVE
meanBinSize	taken from the task wavelengthGrid when an equidistant grid is requested: the equidistant grid's bins size is the mean bin size in the standard grid	WMEANBIN
minRotation	minRotation used when making tod (MADMap)	no keyword yet
missionConfig/uration	internal: mission configuration identifier	MISSIONC
modelName	internal: Flight=PACS in space	MODELNAM
n	number of raster lines (taken from HSPOT); meaningless for pointed observations	no keyword yet
naidif	Solar system object NAIF identifier	NAIFID
nodCycleNum	nodding cycle number	NODCYDEN
nodPosition	nod position (A/B) that this product belongs to (mainly spectroscopy)	NODPOS
NOISYPIXELS	in the flag dataset of a cube: either a T/F indicating whether this cube was created including data with the NOISYPIXELS flag, or a number indicating	no keyword yet

	how many input data-points so affected were included	
Number of lines	number of spectral lines	NUMLINES
Number of Nod Cycles	number of nod cycles	NUMNODCY
number of rows/columns	number of rows/columns on the detector	DETROW/COL
NumFrames/GoodDetectors/SkyPixels/UniqueSkyPixels	number of frames/good detectors in a frame without BLINDPIXELS AND BADPIXELS/sky pixels/unique sky pixels	no keywords yet
Obpc	internal: Onboard Control Procedure	OBCP
object	target name	OBJECT
obsCount	internal	OBSCNT
observer	P.I. name	OBSERVER
ObservingMode	summary of all observing mode Meta data entries	CUS_MODE
obsid	observation identifier	OBSID
obsMode	HSPOT entry: observing mode, e.g. pointed or mapping	OBS_MODE
obsOverhead	observing overheads	OBSOVHD
obsState	Level processed to	OBSSTATE
obsType	internal	OBSTYPE
odnumber	operational day number	ODNUMBER
odStartTime	start of the operational day	ODSTART
optimizeOrientation	optimze orientation when making TOD (MADMap)	no keyword yet
onOffSource	on or off source: 0=not science, 1=on source, 2=off source	ONOFFSRC
order	grating order	ORDER
orderSel	HSPOT: selected blue band order (2=B2B, 3=B3A, 23=B2A)	ORDERSEL
origin	who/what provided these data	ORIGIN
oversample	the oversample parameter value in wavegrid used by pipeline task specWaveRebin	OVERSMPL
oversampleSpatialGrid	value of the oversample parameter used in creating the spatial grid for the drizzle task	OVERS_SG
PACS_OBS_MODE	bolometer readout mode	PHOTMODE
PACS_PHOT_MODE	photometer mode setting	no keyword yet
pacsSliceInfoUpdated	sliced information keywords were updated?	SLICEKEY
phot_[blue|red]_FailedSPUBuffer	internal: percentage of failed SPU (signal processing unit) (anomaly 70)	no keyword yet

pixFrac	spectroscopy: the value of the pixfrac parameter in the task drizzle (spectroscopy) or phot-Prject (photometry) that created this cube or map	PIXFRAC
pixSize	the spaxel size of this cube	PIXSIZE
pmDEC/RA	proper motion	PMDEC/PMRA
pointingMode	pointing mode keyword	POINTMOD
pointSourceCorrected	indicating whether this spectrum, created from a cube, has been point source corrected by one of the pipeline tasks that do this	PSFLXCOR
pointStep	raster step (point=column) in arcsec; meaningless for pointed observations	RASSTEPC
posAngle	Position angle of pointing	POSANGLE
processingMode	pipeline processing mode, SYSTEMATIC means "by the HSC"	PROCMOD
processingParams	see ConcatenatedOBSIDS	PROCPARM
productNotes	text describing the product. e.g. HPSFITB for spectroscopy blue fit ramps	PRODNOTE
proposal	name given to the programme when proposal was submitted	PROPOSAL
qflag_XXX	quality control flags, where "XXX" is replaced by various initials: see their Meta data entries for a description of each XXX; before SPG 13 all PACS products had qflag_ entries related to data issues, but from SPG 13 onwards these have been moved to "info" flags (see entry above for these); most of the flags remaining in SPG 13 products are related to instrument and satellite issues, but having a flag does not necessarily mean there is a problem	various: see the "qualitySummary" in the *ObservationContext* and documentation on the DpKnowIssue HSC web-page for the affected observations
ra/raNominal	RA of satellite/RA requested	RA/RA_NOM
raDeSys	RA, Dec coordinate reference frame	RADESYS
radialVelocity	spacecraft velocity along the line of sight to the target	VFRAME
rangeHigh1/2../rangeLow1/2/..	high or low wavelength end of the spectral range in this produce	RANGEHI1/2/.. RANGELO1/2..
rangeId/rangeSPOT	HSPOT: information about the wavelength range for range scan AORs	RANGEID/RANGDESC

raOff	HSPOT value: requested RA off position	RAOFF
rasterId	raster line, column for this slice	RASID
RAWSATURATION	in the flag dataset of a cube: either a T/F indicating whether this cube was created including data with the RAWSATURATION flag, or a number indicating how many input data-points so affected were included	RSATWARN
redshiftType/Value	HSPOT value: the redshift given by the astronomer	REDSHFT/REDSHFTV
redWave	HSPOT: red wavelength range limit(s)	REDWAVE
refSelected	if true, then allow the use of raoff and decoff	REFSEL
relTimeOffset	offset between PACS internal CRDCcounter and on-board time	RELTMOFF
repetition	grating scan repetition index for this slice (UPSCAN/DOWNSCAN repeats)	REPEATS
repeatLine, repeatRange	how many times the line or range was repeated	LINEREP/RANGEREP
RemovedSetTime	number of removed Frames due to setTime command	RMTIMSET
SATURATIONWARNING	depending on which product it is related to, this is either T/F indicating whether saturated data were included when creating this cube, or a number indicating how many input data (when creating this product) were saturated	SATWARN
scale	pixel scale used when making TOD (MADMap)	no keyword yet
scope	the scope of the calibration products, can take on values of TEST, BULK, PRIVATE, or BASE	SCOPE
slewTime	time of start of slew to the target	SLEWTIME
sliceNum	slice number	SLICENUM
solarAspecAngle[Mean|Rms]	the mean/rms of the solar aspect angle (angle of instrument wrt the sun	SAAMEAN/SAARMS
source	HSPOT entry: type of observation, e.g. point(ed observation) or large (mapping observation) for spectroscopy	SOURCE
spaxelRow/Column/Ra/Dec	the assorted spaxel coordinates that this spectrum was taken from	SPAXROW/SPAXCOL/ SPAXRA/SPAXDEC

spec_[blue|red]_FailedSPUBuffer	internal: percentage of failed SPU (signal processing unit) (anomaly 70)	FAILSPUR/FAILSPUB
startDate/Time	start of observation	DATE-OBS/TIME-OBS
Starting time	start of observation in a friendly format	OBSSTART
subType	internal	SUBTYPE
telescope	obviously: Herschel!	TELESCOP
throw	chopper throw (spectroscopy)	CHPTHROW
TodFilename	filename of temporary TOD file	no keyword yet
type	product type, e.g. HPSFITB for spectroscopy blue fit ramps	TYPE
uniFiltS/StImg/DWgls/GWgls	optional parameters of the UNIMAP processing	UNIFILTS/UNISTIMG/ UNIDWGLS/UNIGWGLS
upsample	the upsample parameter value in wavegrid used by pipeline task specWaveRebin	UPSMPL
upsampleSpatialGrid	value of the upsample parameter used in creating the spatial grid for the drizzle task	UPSMP_SG
userNODCycles	HSPOT value: number of nod cycles requested	NODCYREP
UNVALID	depending on the product, the number of datapoints, regions, or spaxels falling inside the invalid wavelength region (as indicated by the UNVALID mask, set during the pipeline when data are divided by 0)	INVALID
velocityDefinition	which model was used to compute Herschel's radial velocity	VELDEF
verbose	internal	VERBOSE
wavelengthGridEquidistant	taken from the task wavelengthGrid and set to True when an equidistant grid was requested	WGRIDEQ
widthUnit	HSPOT value: line width unit	WID_UNIT

To query the meta data on the command line you can modify these examples:

```
# From an ObservationContext called obs
# - print all the meta data, so you know what are there
print obs.meta
# - print the meta data entry for "obsid"
print obs.meta["obsid"]
# - print only the value of the "obsid" (which is held as a long)
print obs.meta["obsid"].long
print obs.meta["obsid"].value
# - what was the requested RA and Dec
print obs.meta["raNominal"]
print obs.meta["raNominal"].double, obs.meta["decNominal"].double

# For a sliced product, access the meta data in this way:
slicedFrames = obs4.level1.fitted.red.product
print slicedFrame.meta
frames = slicedFrames.get(0) # the first Frames in slicedFrames
```

```
print frames.meta
```

Chapter 5. The Standalone Browse Products

5.1. Introduction

The science-quality products that the pipelines produce are found in Level 2 and 2.5. PACS have also created products that can be downloaded as a tarball of FITS files from the HSA directly, to be read into HIPE or into any other software: these are called the *Standalone Browse Products.* For photometry these are simply the most highly recommended maps for different modes. For spectroscopy they are slightly modified version of one of the set of Level 2/2.5 cubes, also depending on the observing mode, and an additional set of tables.

The term "standalone browse" means that the products have a format that allows them to be loaded into other software fairly easily and are a good example of the data the observation has produced. But they are not necessarily the products that are the final recommendation for science use. While, for photometry the standalone products usually *are* the final science products, for spectroscopy this is not the case. In order to make the cubes easier to read into other software it was necessary to modify the wavelength grid to make it equidistant, and this also slightly modifies the data: hence they are not *exactly* the same as the standard Level 2/2.5 counterparts.

As well as being able to download the standalone products from the HSA directly, you can find them in any *ObservationContext* produced by SPG 13 and higher. When looking at an observation in HIPE with the Observation viewer (double-click on the observation as listed in the *Variables* panel, the viewer will open in the *Editor* panel), you can find the standalone browse products in the :"directory" listing in the Data section of the viewer,

- *Photometry*: in the entry called **browseProduct**. These are in fact links to the maps as held in the final Level of the *ObservationContext*.
- *Spectroscopy*: in the entry called **browseProduct** and also in the **final Level** (2.5 is present [unchopped range scan] or 2 otherwise). These are the cube *context*s with an "EQ" in the name (e.g. HPS3DEQIB for the "equidistant interpolated blue" cubes), and the tables called HPSTBR[R|B].

5.2. Photometry

There are two standalone browse products in the *ObservationContext* of a photometer observation, and one supplementary products:

- **browseImageProduct**: this is the image that can be found at the upper right corner of the *ObservationContext* and also seen in the HSA search result pages.
- **browseProduct**: this is a pair of JScanam maps (red and blue) that are identical to the **level2.5** products; can also be downloaded direct from the HSA as FITS files.
- **pacsObsSummary**: This is not strictly part of the browse product of the photometer but it is a useful piece of the *ObservationContext* so we list it here. It tells you all the important details of the observation in a human-readable form.

5.3. Spectroscopy

Here we describe the standalone browse products created by SPG 13 and higher. You can also create these products when you pipeline process an observation yourself—this is explained in a PACS Useful script in the Scripts menu of HIPE *Scripts#PACS useful scripts#Spectroscopy: Re-create the standalone browse products*.

5.3.1. Equidistant cubes

The pipeline-produced cubes from PACS do *not* have a wavelength grid that is equidistant, i.e. the bins-sizes of the spectral grid are not equal at every wavelength, but rather they scale with resolution, which in turn scales with wavelength. This was done so that even for spectra which cover the entire SED of PACS (50—220µm) a Nyquist (or better) spectral sampling is achieved at *all* wavelengths. This means that the spectral grid is not defined via the third axis of the World Coordinate System (WCS), i.e. we do not have values for the reference wavelength and dispersion in axis 3. Instead, the wavelength grid is contained as a dataset in the cubes. Unfortunately, this can make it cumbersome to load PACS cubes into software other than HIPE and for their spectral grid to be immediately recognised. To combat this, we have created cubes with an equidistant wavelength grid. These can be found in the *ObservationContext*s processed by SPG 13, and they are also one of the a standalone browse products that are now provided via the HSA. These equidistant cubes are created from the final pipeline Level 2/2.5 cubes:

- For Nyquist and oversampled mapping line scan observations: from the drizzled cubes (HPS3DEQD[R|B])
- For Nyquist and oversampled mapping range scan observations: from the projected cubes (HPS3DEQP[R|B])
- For all other observations (pointed and undersampled mapping, line and range scan): from the interpolated cubes (HPS3DEQI[R|B])

(See the PDRG chp. 1 in *PACS Data Reduction Guide: Spectroscopy* for an explanation of the types of mapping modes.) The equidistant cubes are created with the task "specRegridWavelength". The task takes an input wavelength grid that has evenly spaced bins (i.e., "equidistant"), and re-samples all the datasets (image, error, ...) on this new grid. The new cube then has an equidistant spectral grid, and the spatial *and* spectral information is held in the WCS component of the cube:

```
# For a Level 2 interpolated cube (HPS3DI[R|B])
HIPE> print cube.wcs
World Coordinate System
-----------------------
naxis  : 3
naxis3 : 142
naxis1 : 15
naxis2 : 15
crpix1 : 1.0
crpix2 : 1.0
crval1 : 221.29708839108744
crval2 : -20.879270056738726
cdelt1 : -0.001305555555556
cdelt2 : 0.001305555555556
ctype1 : RA---TAN
ctype2 : DEC--TAN
cunit1 : degree
cunit2 : degree
ctype3 : Wavelength
cunit3 : micrometer

HIPE> print cube.getWcs().getCdelt1()*3600 # spaxel size in arcsec
 -4.7
HIPE> print cube.getWcs().getCdelt3()      # wavelength grid bin size
 herschel.ia.dataset.image.wcs.WcsException: The cdelt3 keyword is not available

# For the Level 2 equidistant interpolated cube
HIPE> print eqcube.wcs
World Coordinate System
-----------------------
naxis  : 3
naxis1 : 15
naxis2 : 15
crpix1 : 1.0
crpix2 : 1.0
```

```
crval1 : 221.29708839108744
crval2 : -20.879270056738726
cdelt1 : -0.001305555555556
cdelt2 : 0.001305555555556
ctype1 : RA---TAN
ctype2 : DEC--TAN
cunit1 : degree
cunit2 : degree
naxis3 : 406
crpix3 : 1.0
crval3 : 79.4005558677062
cdelt3 : 0.003359694455696
cunit3 : micrometer
ctype3 : Wavelength

HIPE> print eqcube.getWcs().getCdelt1()*3600
 -4.7
HIPE> print eqcube.getWcs().getCdelt3()
 0.0033596944557
```

External software such as ds9 will be able to not only recognise these cubes as 3D products—this should, in fact, be the case for *all* the cubes PACS produces—but will also display the spectra immediately with the correct wavelength information. The non-equidistant cubes *can* also be read into other software, but because the wavelength grid is not evenly spaced and is instead held in an ImageIndex extension in the FITS files, it may requires extra steps to display the correct spectral information.

When creating the equidistant wavelength grid, attention should be paid to the bin sizes. Bearing in mind that the datasets (e.g. the spectral fluxes) are interpolated from the old grid to the new grid, this will change the spectra. To minimise the amount of change, i.e. to make the new spectra follow the same shape as the old spectra, a higher number of smaller bins is better than a lower number of larger bins. But if you create too many new bins, the cube can rapidly become extremely large and take up much space on disk and memory in HIPE.

Tests have shown that for line scans and ranges of a few microns the match between the old and the new spectra can be achieved with a fractional bin size of 0.35 of the normal grid (this value is explained in the next section), even for spectra crowded with lines. For long ranges and SEDs, the standard wavelength grid varies with dispersion, which varies with wavelength. Hence, the best fractional bin size will not be the same at different wavelengths. In the SPG pipeline we have chosen to use a fractional bin size also of 0.35 as this provides a good compromise and produces well-matched spectra for most cases. **But if you want to do science on these equidistant cubes, you should check this value also works well for your data**: *spectra that are more crowded and with blended lines, or SEDs where the important lines lie in the spectral regions with the greatest difference between the standard and the equidistant bin sizes, generally require smaller bin sizes.*

5.3.1.1. Create your own equidistant cubes

Here we show how to create equidistant cubes and compare them to the original cubes. We will call the wavelength grid from the first the "equidistant grid" and that from the second the "scaled grid".

Starting from an observation (we give an example obsid you can test this on), extract the Level 2 *PacsCube*s (HPS3DB), *PacsRebinnedCube*s (HPS3DRB), and then the cube you want to make equidistant, being the projected, drizzled, or interpolated cube (HPS3D[P|D|I]B; noting that for this example observation there are no interpolated cubes):

```
obsid=1342187206
obs=getObservation(obsid,useHsa=1)

slicedCubes = obs.level2.blue.cube.product
slicedRebinnedCubes = obs.level2.blue.rcube.product

slicedProjectedCubes = obs.level2.blue.pcube.product
# Or, to get the drizzled or interpolated cube context
slicedDrizzledCubes = obs.refs["level2"].product.refs["HPS3DDB"].product
slicedInterpolatedCubes = obs.refs["level2"].product.refs["HPS3DIB"].product
```

To create the equidistant cubes it is necessary to create the wavelength grid using the same task that the original (non-equidistant) wavelength grid of the cubes was created with, some parameters being exactly the same and others being different. This is necessary so that the spectra of the equidistant cube are as much as possible the same as those of the input cube. The parameters for the task wavelengthGrid that need to be the same when creating the equidistant grid are: `upsample`, `oversample`, and `standardGrid`. The former two we get directly from "slicedRebinnedCubes" as shown below, the latter parameter is anyway always True when used in the pipeline (it is the default value in the task). To learn what these parameters mean, we refer you to the *URM* entry for "wavelengthGrid" and the PDRG chp. 6 in *PACS Data Reduction Guide: Spectroscopy*.

```
# Query on the meta data to find the values of oversample
# and upsample used in creation of their the wavelength grid
slice = 0  # any slice will do
oversample = slicedRebinnedCubes.refs[slice].product.meta["oversample"].double
upsample   = slicedRebinnedCubes.refs[slice].product.meta["upsample"].double

# Set the size of the new bins
frac = 0.5 # let's say you want bins of 50% of the smallest in the input cubes

# Get the calibration tree
calTree = getCalTree()
```

Create the wavelength grid, and then resample the projected (or drizzled, or interpolated) cube on that grid, e.g.

```
waveGrid = wavelengthGrid(slicedCubes, oversample=oversample, upsample=upsample,\
 calTree = calTree, regularGrid = True, fracMinBinSize = frac)
slicedProjectedCubes_eq = specRegridWavelength(slicedProjectedCubes,waveGrid)
```

The important parameters in the task wavelengthGrid to be used when creating the equidistant grid are the following:

- **fracMinBinSize**: this is the fraction by which the smallest bin size of the scaled grid is multiplied to set the bin size of the equidistant grid.
- **fixedBinSizes**: to specify fixed bin sizes, enter these with this parameter; since there may be more than one cube in the input sliced cube product, you must specify a line id for each slice using a dictionary, e.g.

  ```
  fixedBinSizes={2: 0.02, 3: 0.03}
  ```

  sets a bin size of 0.02 micron for lineId=2 and 0.03 micron for lineId=3. The lineId values can be obtained by running the task: slicedSummary(sliced cube product).
- **oversample, upsample**: as stated above, their values should be the same as that with which the wavelength grid of the projected (drizzled, interpolated) cubes were created with.
- **regularGrid**: should be set to True to have a grid with equidistant bins. If `regularGrid` is True but neither `fracMinBinSize` nor `fixedBinSizes` is specified, then the bin size of the equidistant grid is the minimum bin size of the scaled grid (i.e. `fracMinBixSize` = 1).
- *standardGrid*: set to True to use a scaled grid with standardised starting and ending wavelength points; this parameter is not important for creating a regular grid but the name is similar—please do not confuse it with "regularGrid"

Do not be tempted to use the equidistant grid in the pipeline when creating the rebinned cubes, rather than the scaled grid that is the default. Compared to re-sampling cubes at the end of the pipeline, the result is noticeably inferior for all except the narrowest wavelength range observations.

Note that it is not a good idea to apply the task specRegridWavelength to projected cubes created for single pointing observation: these cubes have very many very small spaxels (0.5"), and HIPE can run out of "Java heap space".

5.3.1.2. Comparison of equidistant with standard cube spectra

Here we show a comparison between a spectrum taken from a projected cube and the spectra from its equidistant counterpart cubes for different values of `fracMinPix`. In the black curve is the spectrum of the central spaxel from the projected cube with a standard wavelength grid (oversample=2, upsample=4), and in coloured dots are the datapoints from the equidistant version of that same cube with a wavelength grid created with fracMinBinSize of 0.5, 0.35, and 0.25. The top spectrum is an entire range and that below a zoom in on the line. It is clear that all of the equidistant spectra match the shape of the standard spectrum very well.

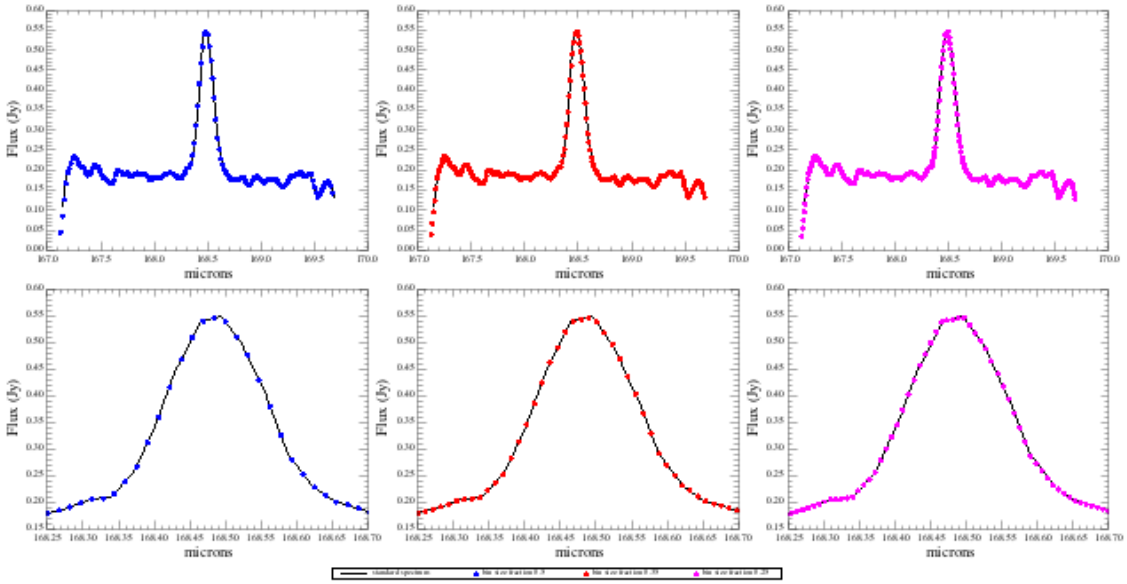


Figure 5.1. Comparison of the spectrum with a scaled grid and the same data interpolated onto different equidistant grids (fracMinBinSize = 0.5: blue; 0.35: red; 0.25: magenta): a very short range scan

Next we show the same plots but for a full SED, using the standard cube (created with oversample=2, upsample=2) and the equidistant cubes with a range of `fracMinBinSize` values. The plots show parts of the blue, central, and red ranges of the blue camera and red camera data. The largest bin size for the scaled grid for this observation is in the very blue for the blue camera (B2B) and also for the red camera (R1).

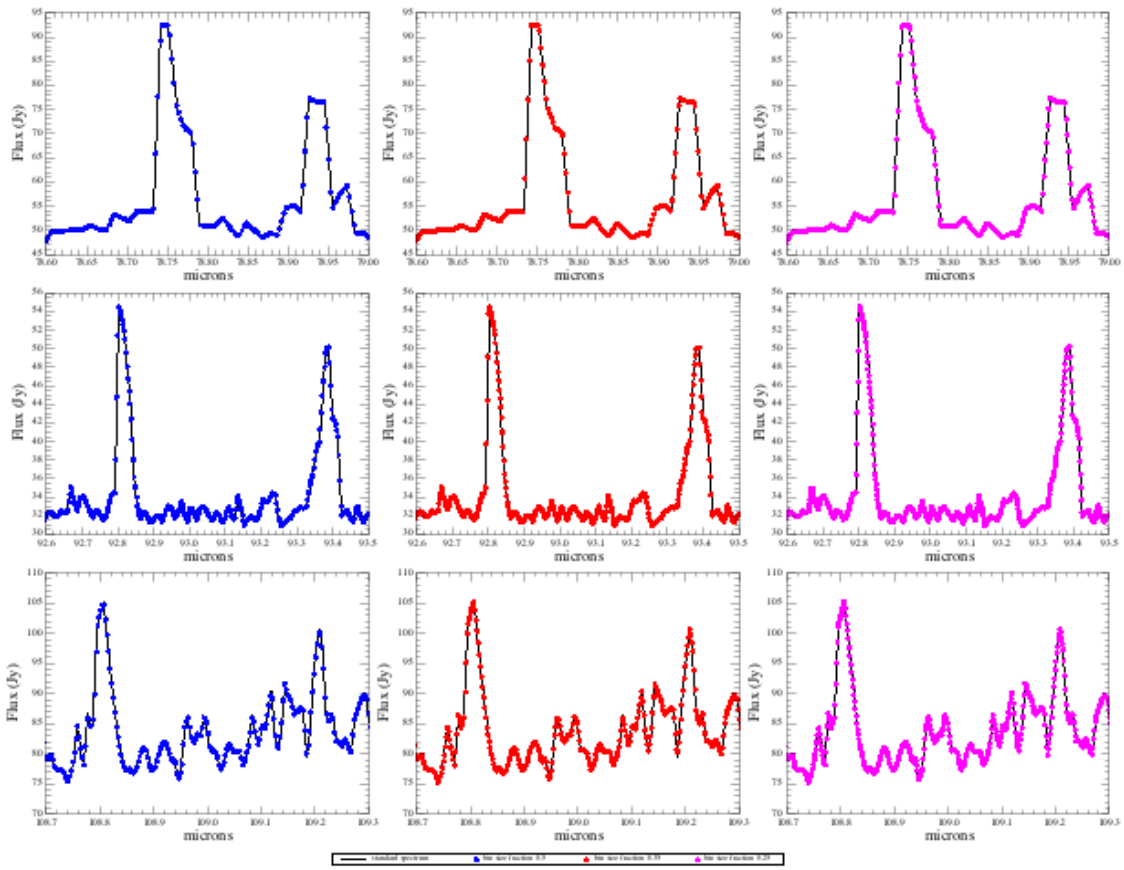


Figure 5.2. Comparison of the spectrum with a scaled grid and the same data interpolated onto different equidistant grids (fracMinBinSize = 0.5: blue; 0.35: red; 0.25: magenta): blue part of an SED

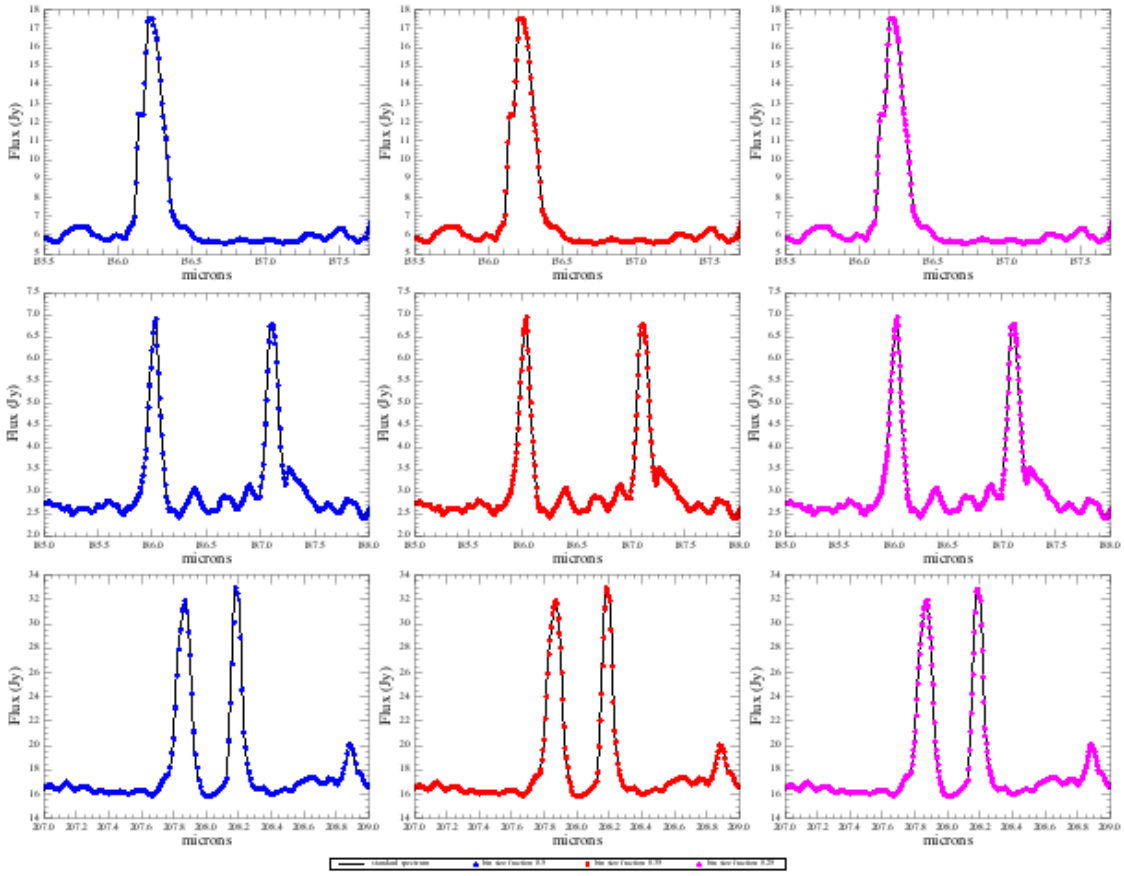


Figure 5.3. Comparison of the spectrum with a scaled grid and the same data interpolated onto different equidistant grids(fracMinBinSize = 0.5: blue; 0.35: red; 0.25: magenta): red part of an SED

It is clear from these plots that even for this SED, the equidistant grid follows the standard grid data very well.

A note of warning, however, about having many, small bins: this can make the data appear to have a resolution greater than it actually has. It must be remembered that these equidistant-grid data are an *interpolation* of the original data and have a sampling that implies a greater resolution than the data actually have. It is for this reason that the Meta datum "fracMinBinSize" is included with the cubes, so you can work out by how much this is.

Finally, we include here plots of the bin size of the scaled wavelength grids for the different filters, computed with the parameter upsample=1 and oversample=2 in the task "wavelengthGrid"—the bin sizes depend on these two parameters (see the PDRG in *PACS Data Reduction Guide: Spectroscopy*) but the same shape as plotted here will always be present.

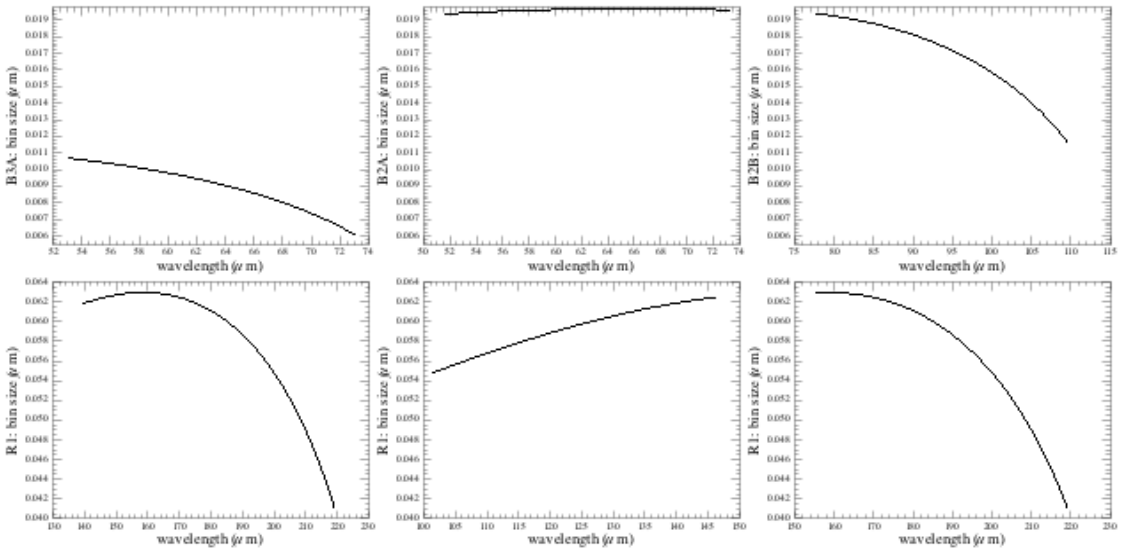


Figure 5.4. Dispersion as a function of wavelength for the various spectroscopy filters for a wavelengthGrid created with upsample=1 and oversample=2.

Note: to create a similar plot to the one above, this is the key phrase:

```
wave = cube.getWave()
PlotXY(wave, wave[:-1]-wave[1:])
```

5.3.2. Rebinned cube tables

The rebinned cubes are the recommended science-end product for single pointed observations, and especially for those of point sources. These cubes, however, have neither an equidistant spatial nor equidistant spectral grid. The sky footprint of these cubes is that of the instrument, i.e. spaxels of 9.4" size, in a slightly irregular 5x5 grid (see the PDRG chp. 8 in *PACS Data Reduction Guide: Spectroscopy* for more detail). Because these rebinned cubes can be less than straightforward to read into other software—which usually expect cubes to be equidistant along all axes—we have provided the same data in a table format. For each wavelength range in the observation, and for each camera separately, all the spectra of all spaxels for all raster positions are included in a single table. The collection of tables are held in a context of class *SlicedPacsSpecTable*, called **HPSTB[R|B]**, which can be found at Level 2 (or 2.5).

To see these tables in HIPE, open the Observation viewer on your observation (double-click on the observation in the Variables panel), and go to the Data panel. Click on +level2 (or +level2.5 if there is one) and then on +HPSTBR, and then click on the +# of any one of the tables, e.g. +0 (which will also have text similar to "0 L1 N1 R(0 0)"—this text is explained in Chapter 2), and then finally on the "Spectra" to see the data in the table:

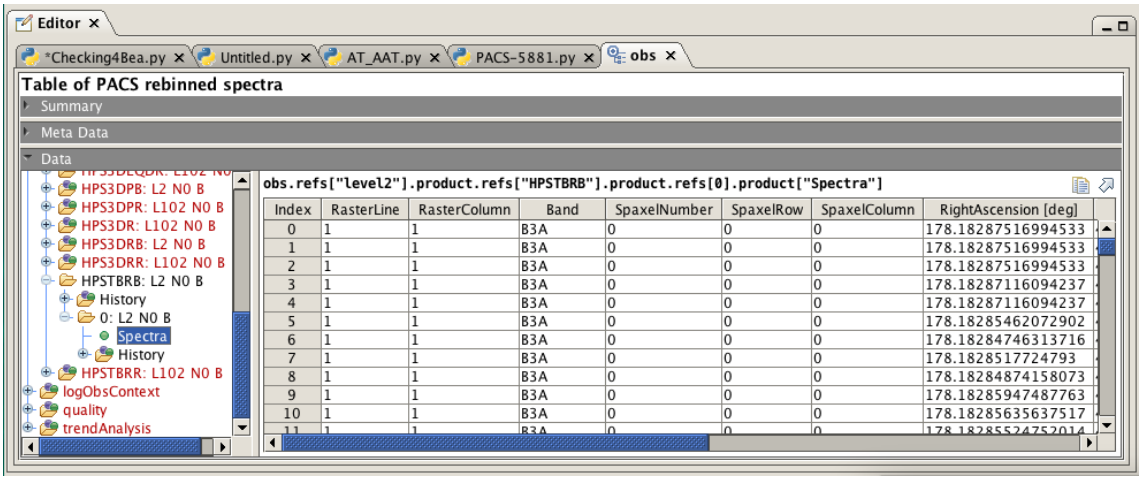


Figure 5.5. The PACS rebinned cube table

There is one table per wavelength range in the observation. For observations with several raster positions, the data from all raster positions are contained within one table. The length of the table is dictated by how many spectra there are, since all the data for each wavelength point of each spectrum fills its own row. The columns are:

- Raster Line and Raster Column: the position within the raster sequence
- Band: the filter band (the same for all data)
- Spaxel Number, Row, and Column: spaxel index (0—25) and row (0—4) and column (0—4) coordinates
- RA, Dec
- Wavelength, Flux, Flux Error

So, for a cube with spectra of 10 datapoints, in a raster with 4 pointings, the order of rows is:

- 0—9: ra, dec, wave, flux, error for raster line 1, column 1, for spaxel row 0, column 0, index 0
- 10—19: ra, dec, wave, flux, error for raster line 1, column 1, for spaxel row 0, column 1, index 1
- ...
- 240—250: ra, dec, wave, flux, error for raster line 1, column 1, for spaxel row 4, column 4, index 24
- 250—260: ra, dec, wave, flux, error for raster line 1, column 2, for spaxel row 0, column 0, index 0
- ...

These tables can be saved to FITS or as an ASCII file to disk. In the FITS file, the header of extension 1 contains the names and units of the columns; in the ASCII file these information are given at the beginning of the file.

The useful Meta data is attached to the *PacsSpecTable* product (+0 etc in the Observation viewer).