# Minutes of Herschel DP-SAG Meeting #9

| | |
|---|---|
| **Location** | RAL, CR13, R68 |
| **Date** | 17-18 August 2006 |
| **Reference** | SPIRE-RAL-MOM-002712 |
| **Issue** | 1.0 |
| **Date** | 8th September 2006 |
| **Prepared by** | S.Guest |

## *Welcome*

### Participants

| | |
|---|---|
| J.Bakker | HSCDT |
| S.Guest | SPIRE |
| R.Huygen | PACS |
| S.Ott | HSCDT (observer, first day only) |

It was stated and agreed that in principle:

- Representatives from each instrument and HSCDT should be present at DP-SAG meetings.
- Deputies should be sent if necessary.
- If this is not possible, decisions are made "ad referendum", e.g. they are not discussed again because the representative was not present, but only if serious flaws were found.

### Minutes of Previous Meeting

Everyone agreed that the level of detail in the minutes is fine.
It was agreed that the responsibility for producing minutes of meetings will be rotated. The host of meeting will take minutes. Minutes should be released within a week if possible. If not, the actions and next meeting date will be.

### Agenda

The organising host should prepare and send out the draft agenda 2 weeks in advance of the meeting.

The additional comments provided by RH and SG on the draft agenda were incorporated into the meeting agenda. It was pointed out that a number of actions from the previous meeting were missing – these were also added. As the agenda was crowded, the second day would be prioritised.

### Actions

**DP-SAG-08-06:** JB has produced a partial and unfinished prototype of serving telemetry packets as products. The performance of this prototype was a factor of 3 faster than

Versant (15,000 packets/sec). SG is worried about performance problems with trend analysis scenarios. RH will refactor PacketSequence to factor out the selection of packets into a special Selector class. The transport of telemetry packets might benefit from compression. BZIP should be faster than GZIP.

**DP-SAG-08-08:** There is now a prototype scheme for handling multiple Jython trees. This is dealt with as a separate agenda point.

**DP-SAG-08-12:** RH reported that Table Viewer is now available. However, JB couldn't find it, possibly not part of the build script or not tagged correctly? RH will check if it is in the build and JB will contact RZ if necessary.
**DP-SAG-08-13:** SG sent a proposal on the use of dates versus fine times. There was some discussion on formatting of data, e.g. TAI/Date and decimal/hex.

## *DP Processing & SPG*

The pipeline scripts used by each instrument team for the DP E2E tests were shown and the similarities and differences (which were significant) remarked upon.
**DP-SAG-09-02:** JB to brief SL on differences in instrument pipelines.

The HIFI pipeline was shown. It was noted that interactive stepping was not possible, and that the pipeline was implemented as task.

*Post-Meeting comments from PZ:*
The HIFI pipeline consists of three pipeline scripts:
1. HRS
2. WBS
3. super pipeline which, depending on APID, uses the WBS or HRS pipeline

The first two are stepwise and are not tasks, whereas the third one is a task and is not stepwise. The super task was needed as the e2e test requested a single pipeline to be deined or both backends. Note that the latter functionality (chaining pipelines) is needed within the ICC as well as in the future the WBS and HRS will be followed by the common (AOT) pipeline. The task was introduced to allow chaining, however conflicts the stepwise-requirement. Needs to be discussed. CC to Nicola? Or discuss first in SAG?

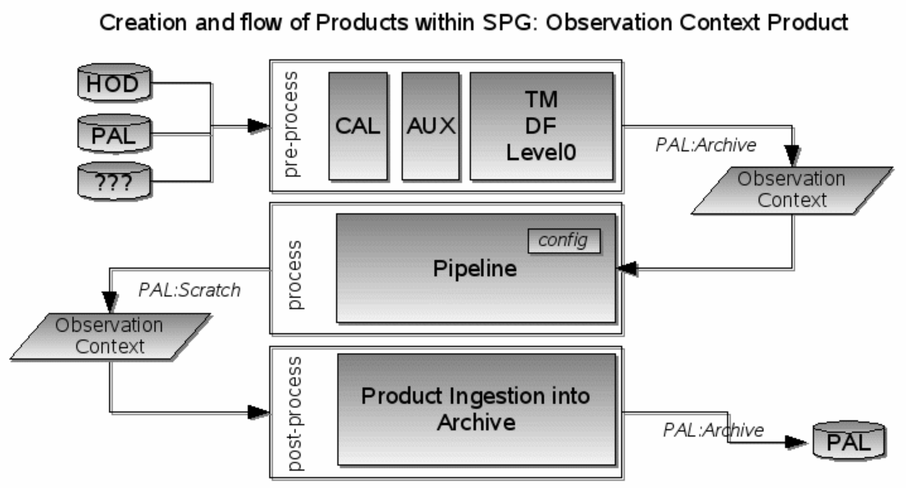
The PACS pipeline was shown. Currently not all of the processing elements in it are tasks.

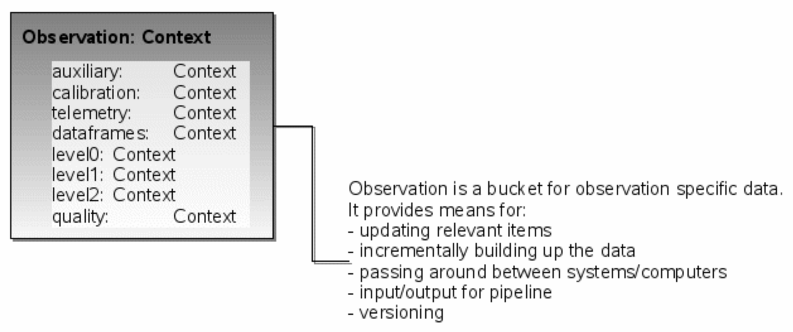
The SPIRE pipeline was shown. The SPIRE pipeline is already using product contexts to group related products. The POF2 and POF5 pipelines contain loops over building blocks. The downside of this is a loss of stepping. (In theory there is a workaround with JIDE i.e. set the BBID manually and then start stepping past the loop; however some JIDE problems such as a problem with continuation lines prevent this). The SOF2 pipeline explicitly resets task definitions to "None" to counteract a memory leak with tasks.
**DP-SAG-09-01:** SG to send issues with task framework to NdC.

**DP-SAG-09-03:** JB to distribute the DP E2E Test Reports to DP-SAG.

Though the pipelines were quite different they all followed a similar pattern, revealing some needs in SPG that are currently lacking.

JB then presented some ideas some ideas concerning the use of an “observation context” for use in SPG, see figures below. This takes into account the fact that not all data is necessarily available to the pipeline at the same time. The observation context containing all necessary data can be built up one item at a time. It needs to be remembered that data to be processed might not be in an observation, i.e. is not limited to an observation. SG already has a BuildingBlockContext class in the proto area, which has some similarities.





Observation is a bucket for observation specific data.
It provides means for:
- updating relevant items
- incrementally building up the data
- passing around between systems/computers
- input/output for pipeline
- versioning

A preprocessing step in SPG would be run for the whole OD. This step could save its data in a (scratch area) local store. It gathers/generates OOL, telemetry loss, TA data that might be needed during the execution of the pipeline. This scenario seems to be ok for

SPG, and should also work in an interactive environment. Interactive use by the ICCs might use a *different* preprocessing step, where data can be directly extracted from the DB, and Level-0 prods don't necessarily have to be written to the local disk. This preprocessing step makes the system more robust and could be used for e.g. indexing available data, processing calibration flashes at start of OD etc. It can be expected to have the object database available.

SO remarked that the Data Products group should be briefed about contexts for grouping products. It is possible that some data e.g. data that contains sensitive information, is not included in final SPG output. Could this be configurable? Can ObservationContext support it? It was felt that it probably could.

There followed some discussion of what is passed to tasks, whether everything (ObsContext), or just what is relevant to the particular task. It was concluded that it should *not be* the entire observation context, which should only be available to the pipeline and not to tasks. In particular, no hidden global variables would be available to a task. In other words, a task should be fully defined by its inputs.

Default parameters as inputs to tasks, such as calibration items, could be set by the preprocessing or a separate "inputs" stage. Tasks would then always be supplied with meaningful defaults without having to search global variables for them themselves. This task set-up then needs to know about the time, instrument model etc. RH expects tasks to have available all instrument model configuration info. Configuration parameters can be accessed either by being setup at start of pipeline, or configured for a specific task. SO asked if there could be common keywords in pipeline scripts such as debug?

It is possible that one pipeline branch might need a completely different set of parameters to another. This is a downside to using a single pipeline script entry point for each instrument (e.g. pacs.py). An alternative is a script for each observing mode (i.e. current SPIRE approach). An observer would want to select an observation and be automatically presented with a sensible default set of pipeline configuration parameters, with an option to change them if desired.

SPIRE needs (e.g. for resets and offsets data) to be able to make available for an observation non-scientific data from other observations. Where would this data be stored? This objective could be achieved by the preprocessing step querying the object database and updating the cal layer via the PAL. The data is then treated as part of the calibration dataset.

Pipeline scripts will be run by a "pipeline manager", either a real person, or an application, or both. Inputs and an observation context are supplied to the script. Scripts do not poll for available data. There needs to be a means of saving intermediate data, but possibly not to the same pools where the standard products are stored. Alternatively, all products can be saved to the same temporary pool, and a post-processing step could clone the standard products into the "official" area.

The Quality Control Product was then discussed. This can include quality control measures/metrics such as glitch rates and RMS values. The tasks are responsible for calculating these metrics, not the pipeline. They can be stored as metadata items in the output products of the tasks.

Should the QCP be passed to tasks? Or should the pipeline (or extra task) be responsible for collating the information into the QCP. If the analysis of quality metrics is an additional post-processing step, it avoids having to rerun the entire pipeline if a quality threshold changes.

Responsibilities for pre-processing stage:

- AUX is the responsibility of ESA.
- LEVEL 0 generation is the responsibility of the ICCs.

Responsibilities for post-processing:

- Responsibility of ESA to clone the output standard products into the official product pool.
- Responsibility for filling the QCP must be shared.

What are the requirements placed by trend analysis on the SPG? Will this break the current assumptions/restrictions that we have put into the system? For some TA parameters you know in advance that you want to monitor them, for others not, but that might change during the mission. What happens then? Will there be a major reprocessing or will the preprocessing step be adapted so these particular parameters are saved as metadata at the next major re-processing or at the next pre-processing of any observation?

## *DP Framework*

### Logging

This issue was discussed as it was felt that a lack of agreed conventions on use of logging levels was resulting in serious messages being missed owing to too much noise. It was agreed that:

- INFO should be the standard level for a user.
- FINE should be the standard level normally used by a developer. Messages at this level should enable a developer to see what is going on without being swamped by too many messages.

This information should go in the Developer’s Manual.

The logging mechanism can be used to modify the levels on offending classes, but how to do this is opaque to most users. It was remarked that the Java 5 graphics libraries present a problem here. These settings can reduce the noise from these:

```
javax.swing.level = INFO
java.awt.level = INFO
sun.awt.im.InputContext.level = INFO
```

It was also felt that we need better filtering of logging messages, for example, only show the first few messages of the same kind with a comment like “this message is repeated XX times”. Such filters could be part of the herschel.share.log package.

**DP-SAG-09-04:** SG to inform GT on additional information in developer’s manual.
**DP-SAG-09-05:** SG to talk to HS about filtering.

## Jython Imports

JB gave a demo of Jython import for parallel trees, see develop/proto/dp/jython-import. Three lines need to be put at the front of each “__init__.py” file (see examples). Jake could be adapted to auto-generate these files. Individual package owners would be informed what to do to conform, should any action be necessary.

**DP-SAG-09-08:** JB to raise the issue on environmental variables again, discuss with JB/CP/RZ.
**DP-SAG-09-09:** JB to introduce the Jython import mechanism into the HCSS.

## Masking

It was discussed whether there was commonality in the different implementations for the different instruments and in such things as ImageDataset and SpectrumDataset. For example, ImageDataset uses Bool2d, while both PACS and SPIRE use an integer bit mask in pipeline products. The existing PACS implementation was considered too PACS-specific for consideration as common code; while the SPIRE one was only in rough prototype form. These two approaches could well be compatible however, as they both involve bit masks.

The SPIRE approach defines a MASTER bit, which is used to override other bit settings, e.g. if for a particular value the GLITCH bit is set by the glitch detection mechanism, this value will be ignored by the software and not processed. A user might decide that the glitch is not really a glitch and he or she can unset the MASTER bit, indicating that the software should process that value, but leaving the glitch information in the mask. Data processing software would normally only test if the MASTER bit is set.

The SPIRE prototype is implemented as a Java *enum* type, which is very convenient, but disallows the possibility for users to define their own mask bits. It could also be implemented using the old “typesafe enum” pattern, which is wordier but does not have the same restriction.

The discussion then turned to whether mask support could and should be defined in the numeric package. We would need a selection mechanism along the lines of:

```
    s.where(GLITCH|BAD|NOISY)
    tds.select(s)
```

The above is possibly already implemented with the ALL_PRESENT and ANY_PRESENT functions. Functionality would also be needed to set and unset bits. The methods on Int1d objects in the SG class would be best put into AbstractElementalArrayProcedure (TBC).

It was concluded that SCR-0952 is *not* necessary to implement a mask scheme. However, the group felt that it is still worthwhile to implement this SCR.

**DP-SAG-09-10:** SG to investigate some more how masks could be handled in numeric. The topic will then be revisited at the next meeting.

## *Product Access Layer*

PACS are looking into 'simple' way for users to define their own Jython (?) classes extending from Product. This should enable a calibration scientist to define new calibration files themselves. SG remarked that the product “type” attribute can be used for the purpose of creating new calibration product types without subclassing.

The PAL should include a utility that can be used to ensure that the contract and behaviour of each pool is the same (testing against the interface). This utility can then be used in test harnesses as well. Similarly, such a tool should be developed for Products and Datasets (see for example SCR-2255).

### HCSS action 160606/7

The justifications for not making datasets and products equivalent were contained in the minutes of the “products & algorithms” meeting of 4-5 February 2004 (livelink http://xrl.us/rcrj, Appendix M). There remain some arguments to change and improve things, but they are not compelling or justifiable considering the costs and knock-on effects (backward compatibility etc). It is therefore recommended to keep the current split. It may be useful to provide more explanation of this in the documentation.

Users would be helped if the Product class contained a setDefault method, as this would allow a more convenient wrapping of a dataset inside a product.

There was some discussion on when to use a CompositeDataset and when to use a ListContext. It really depends on the number of products/datasets. The context would contain a history; on the other hand the context might result in a lot of products to work with when one would be more convenient. The question then came up as to how history was handled recursively inside products, when this had previously been identified as one of the problems with merging product and dataset.
**DP-SAG-09-06:** JB to discuss with NdC how history is handled with product contexts.

Product nodes were an earlier implementation of composite products. Now PAL contexts should be used instead i.e. the product nodes are obsolete.
**DP-SAG-09-07:** JB to create first draft of response to HCSS action 160606/7.

### PAL/HAB connection

JB summarised the PAL/HAB meeting.

- The PAL can be used as a front-end in an application that needs access to the HAB.

- It can be used to clone output products from the pipeline and ingest them into the HAB. The cloning of products, especially contexts, should be checked with respect to references to other pools etc.
- It can be used to interface between the object database and provide e.g. telemetry products.
- The public PAL interface to the HAB will disable the "save" feature.

**DP-SAG-09-11:** JB to distribute location of the minutes of PAL/HAB meeting in Livelink.

## Local Store

SG summarised that local store is a product pool with a few extra features, such as:
- control about what files are to be called and where they go
- can do an import without changing the directory structure
- resistant to schema changes
- attributes and meta data are saved in separate tables outside of a given directory structure
- additional maintenance methods such as remove()

JB wondered if the remove method should be propagated into the PAL, and questioned the use of public versus private classes in the package.
**DP-SAG-09-12:** SG to check with LB if all local store classes need to be public.

## *AOB*

SG presented his list of issues arising from the preparations for, and execution of, the DP E2E test. This resulted in a number of additional actions, see summary below.

## *Summary of Actions*

All actions have a due date of next meeting, except where otherwise stated.

**DP-SAG-09-01:** SG to send issues with task framework to NdC.
**DP-SAG-09-02:** JB to brief SL on differences in instrument pipelines.
**DP-SAG-09-03:** JB to distribute the DP E2E Test Reports to DP-SAG.
**DP-SAG-09-04:** SG to inform GT on additional information on logging in developer's manual.
**DP-SAG-09-05:** SG to talk to HS about log message filtering.
**DP-SAG-09-06:** JB to discuss with NdC how history is handled with product contexts.
**DP-SAG-09-07:** JB to create first draft of response to HCSS action 160606/7.
**DP-SAG-09-08:** JB to raise the issue on environmental variables again, discuss with JB/CP/RZ.
**DP-SAG-09-09:** JB to introduce the Jython import mechanism into the HCSS.
**DP-SAG-09-10:** SG to investigate some more how masks could be handled in numeric *(due date 8th September).*
**DP-SAG-09-11:** JB to distribute location of the minutes of PAL/HAB meeting in Livelink.

**DP-SAG-09-12:** SG to check with LB if all local store classes need to be public.
**DP-SAG-09-13:** SG to raise SPR on PAL rules failure messages.
**DP-SAG-09-14:** SG to raise SxR on predicate operations losing toString info.
**DP-SAG-09-15:** JB to look into loading a product storage from a configuration via a factory method.
**DP-SAG-09-16:** SG to test DB Pool again and raise any SPRS.
**DP-SAG-09-17:** SG to check whether the dates in exported FITS files are UTC or current locale.
**DP-SAG-09-18:** JB to follow up JConsole issues with NdC e.g. multiline comment problem (SPR-2164), problem with lines with continuation inside for loop, comments required on empty lines.
**DP-SAG-09-19:** RH to scan the E2E test reports for the relevant issues at an architectural level and bring them to the attention of the DP-SAG.

## *Date of Next Meeting*

24-25 October at ESAC (co-located with DP-UG)