



Data Access & Data Handling in HIPE

Eva Verdugo
HSC-ESAC

HERSCHEL SPACE OBSERVATORY

Read Retrieved data from HSA in HIPE



- From the Navigator view: double click on the obs. xml file
- A variable is created and showed in the editor
- The command used is echoed in the console: ***loadObs***

The screenshot displays the HIPE 6.0.3 interface with the following components:

- Editor:** Shows the "ObservationContext for HIFI data of observation 1342194799". It includes a summary table and a tree view of data files.
- Variables:** Shows a single variable named "obs_1342194799".
- Navigator:** Shows a file tree structure with the file "1342194799" selected under the "everdugo29514683" directory.
- Console:** Shows the command `HIPE> obs_1342194799 = loadObs(path="/Users/everdugo/Data/new_format/everdugo29514683",obsid=1342194799)` and the prompt `HIPE>`.

ObservationContext for HIFI data of observation 1342194799			
Summary			
Object:	W51	Instrument:	HIFI
RA:	19h 23m 45.97s	DEC:	14° 30' 36.73"
Observation ID:	1342194799	Operational Day:	338
Observation Mode:	DBS fastChop		
Meta Data			
Data			
obs_1342194799	obs_1342194799		
History			
auxiliary			
calibration			
level0			
level0_5			
level1			
level2			
logObsContext			
quality			
trendAnalysis			

Read Retrieved data from HSA in HIPE



- Interactively: From a GUI
 - Double click on the observation in the Navigator view
 - Command line:
 - Task: **loadObs**
 - ✓ path to the data
 - ✓ obsid (only required if more than 1 obsid under the same path)
- obs = **loadObs**("/Users/everdugo/Data/everdugo29514683",[obsid])

HERSCHEL SPACE OBSERVATORY

It can be a whole obsContext or a partial observation (supported since HIPE 6.0)

A quick look of the Data



Interactively: ***Send to External Application*** from HUI

The screenshot displays the Herschel Science User Interface (HUI) with several windows and menus open. The main window shows a list of observations, with the observation ID 1342212304 selected. A 'Send to External Application' dialog is open, showing a list of data levels: All, Level2.5, Level2, Level1, Level0.5, Level0, and Calibration. The 'Tools' menu is open, showing options like 'Interoperability', 'Plug-ins', and 'SAMP Hub Status'. The 'Interoperability' sub-menu is also open, showing options like 'Connect to the VO', 'Disconnect from the VO', 'Send Product to...', 'SAMP Hub Status', and 'Help on Interoperability'. The 'Log' window shows the observation ID 1342212304. The 'Meta Data' window shows the following data:

name	value
type	OBS
creator	SPG v5.0.0
creationDate	2011-01-04T17:45:38Z
description	ObservationContext for SPIRE data of obser
instrument	SPIRE
modelName	FLIGHT
startDate	2011-01-04T08:57:41Z
endDate	2011-01-04T09:07:24Z

The 'Data' window shows the following data:

obsid_1342212304	obsid_1342212304
auxiliary	
browseImageProduct	
browseProduct	
calibration	
level0	
level0_5	
level1	
level2	
logObsContext	
quality	

A quick look of the Data



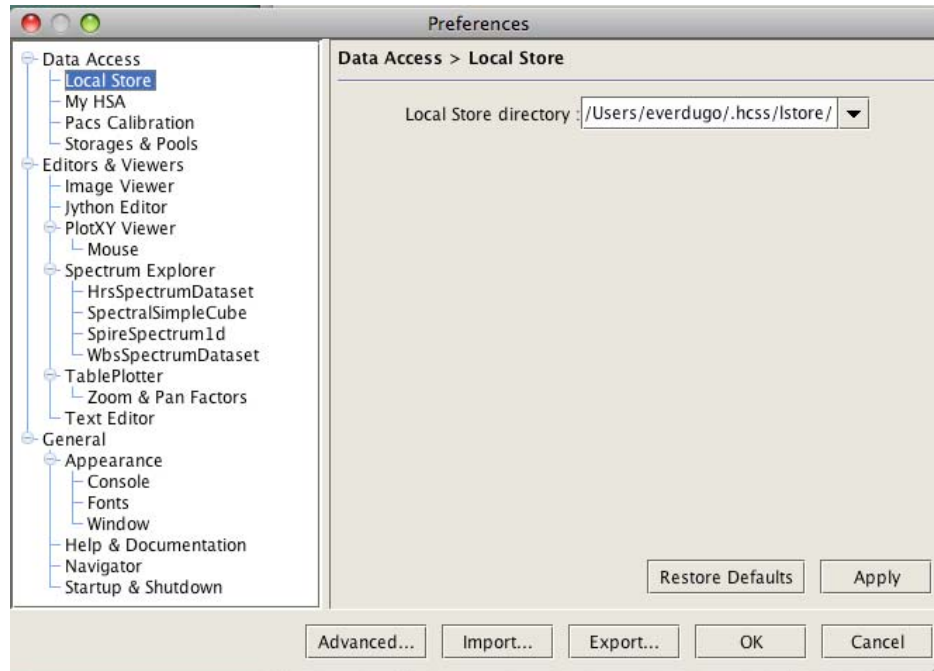
- Interactively: Send to External Application
 - Open HUI & HIPE (or HUI from HIPE)
 - HIPE => Tools => Connect to the VO
 - HIPE => Herschel Science Archive Perspective => Login
 - HUI => Query Results => **Send to External Application**
 - A variable obsid_XXXXXX is created in HIPE and displayed in the Editor
 - The data is **NOT** in your disk; **Only in HIPE memory**
- Command line:
 - Task: **getObservation**
 - ✓ obsid
 - ✓ useHSA = True
 - `obs = getObservation(134222305,useHsa=True)`
 - A variable obs is created in HIPE
 - The data is **NOT** in your disk; **Only in HIPE memory**

HERSCHEL SPACE OBSERVATORY

Advance usage: Pools



- Pools are “special” directories in your disk (local stores)
- By default they are created under `.hcss/lstore/`



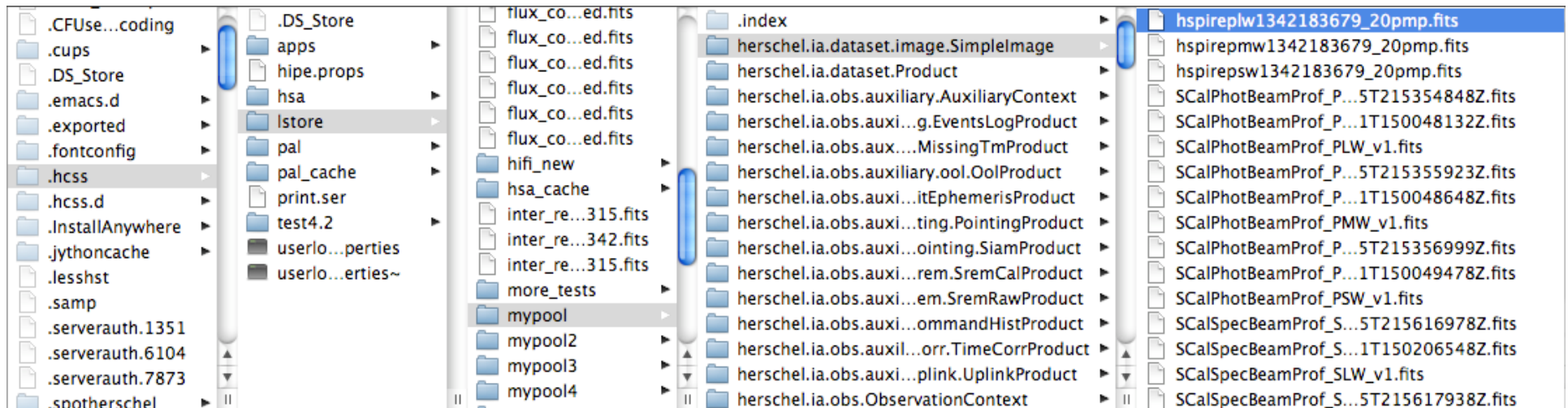
Use Edit → Preferences
to change default location

- They constitute a local database which implies pools should not be
 - Moved
 - Renamed
 - Partially deleted
 - Manipulated manually

Data Structure in Pools



- Ordered by “classes” under which the FITS files are stored
- HIPE does the tree-like structure from this

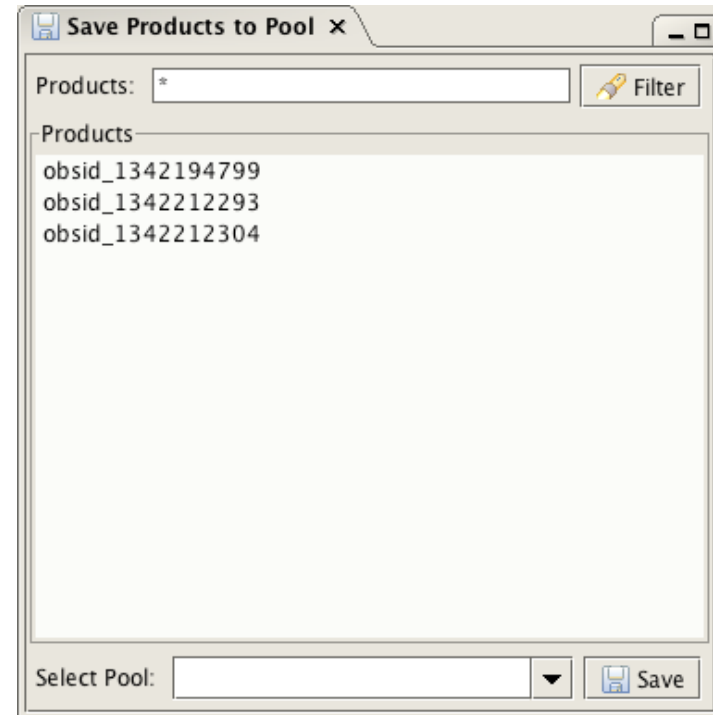
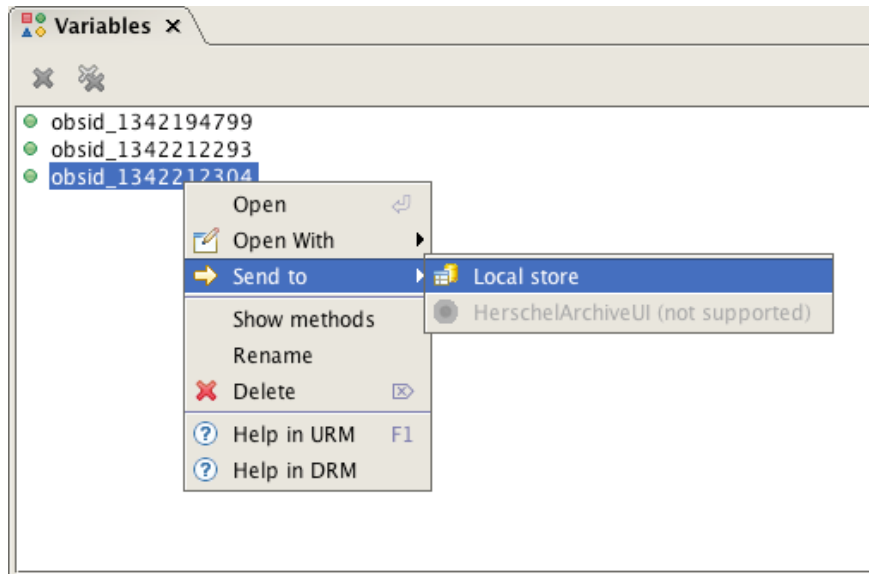


HERSCHEL

How to populate Pools



- Interactively: Save Products to Pool View
 - Displays all product-type variables of your HIPE session
 - Save into an existing pool or create a new one
- Right click on the variables

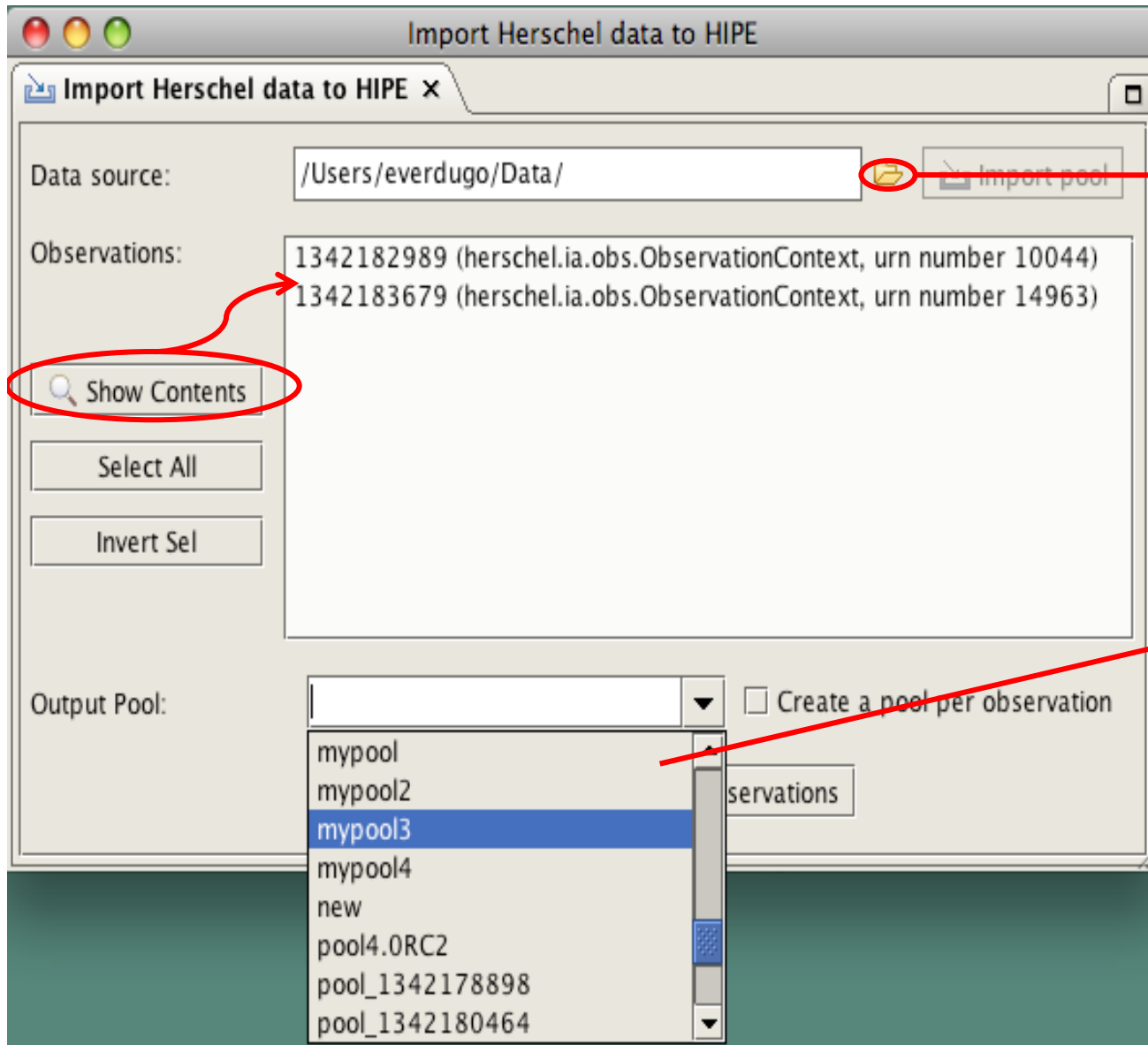


- Command Line: ***saveObservation***
saveObservation(obs)
saveObservation(obs, poolName="od144")

Saving data retrieved from HSA into pools



Import Herschel data to HIPE view



Select a directory with HSA data

Select an output pool or create a new one

HERSCHEL SPACE OBSERVATORY

Access Data in pools: Interactively



Product Browser Perspective

The screenshot displays the HIPE 6.0.3 interface with the Product Browser perspective. The main window is titled "HIPE 6.0.3 - obsid_1342183470". The interface includes a menu bar (File, Edit, Run, Pipeline, Window, Tools, Help), a toolbar, and several panels:

- Product Browser:** Shows a search interface with tabs for "Observation", "Metadata", and "Free Metadata". A search query is entered, and a list of results is displayed. The selected result is "herschel.ia.obs.ObservationContext".
- Table:** A table with columns: Observation Id (obsid), Operational Day (odNumber), Observation start date (startDate), AOT ID (aot), Instrument (instrument), and Observation. The first row is highlighted in green.
- Console:** Shows the execution of a query:

```
ProductStorage(PalModels.getPool('demo-spire-spec-0x5000182E')).select(herschel.ia.pal.query.MetaQuery(herschel.ia.obs.ObservationContext, "p", "1"))
```

 and the output:

```
HIPE> # Added variable: QUERY_RESULT_2
HIPE> # Added variable: selected
HIPE> obsid_1342183470 = QUERY_RESULT_2[0].product
HIPE>
```
- Variables:** A list of variables including obsid_1342183470, obsid_1342194799, obsid_1342212293, obsid_1342212304, and QUERY_RESULT.
- Editor:** A panel titled "obsid_1342183470" showing a tree view of the product structure, including "browseImageProduct", "browseProduct", "calibration", "level0", "level0_5", "level1", "logObsContext", and "quality".

The status bar at the bottom indicates "Jython Interpreter 100%" and "144 of 2044 MB" of memory used.

Access Data in pools: Command line



- Access pools in scripts is done with the task:
getObservation(obsid, poolName)
- In this workshop we have provided you with pools that will be read by the pipeline scripts with ***getObservation***

HERSCHEL SPACE OBSERVATORY

Powerful queries using pools



HERSCHEL SPACE OBSERVATORY

- Interactively: with the Product Browser
 - On pre-defined metadata
 - On free metadata

- Scripts: **Possible queries down to data level in pools**

The following demo script searches for all simpleImage products in a pool and retrieve only those images in which at least one pixel has a flux value larger than 1 Jy. It does this in a couple of seconds for 32 simpleImages.



```
# Demo script to make PAL searches down to data level
# Author: Paul Balm (HSC)
```

```
# preparatory work:
# prepare a pool called dpug_search with all scanmaps in OD 160
# search to data level in dpug_search pool
```

```
# Define a function that returns true for the images we're looking for
def testImage(i):
```

```
    # image type: Blue detector
    if i.type != "HPPPMAPBS":
        return False
    # AND: at least one pixel over 1 Jy
    if max(i.image) < 1.0:
        return False
    return True
```

```
# Define the query using this function and run it
query = FullQuery(SimpleImage,'i','testImage(i)')
refs = dpug_store.select(query)
```

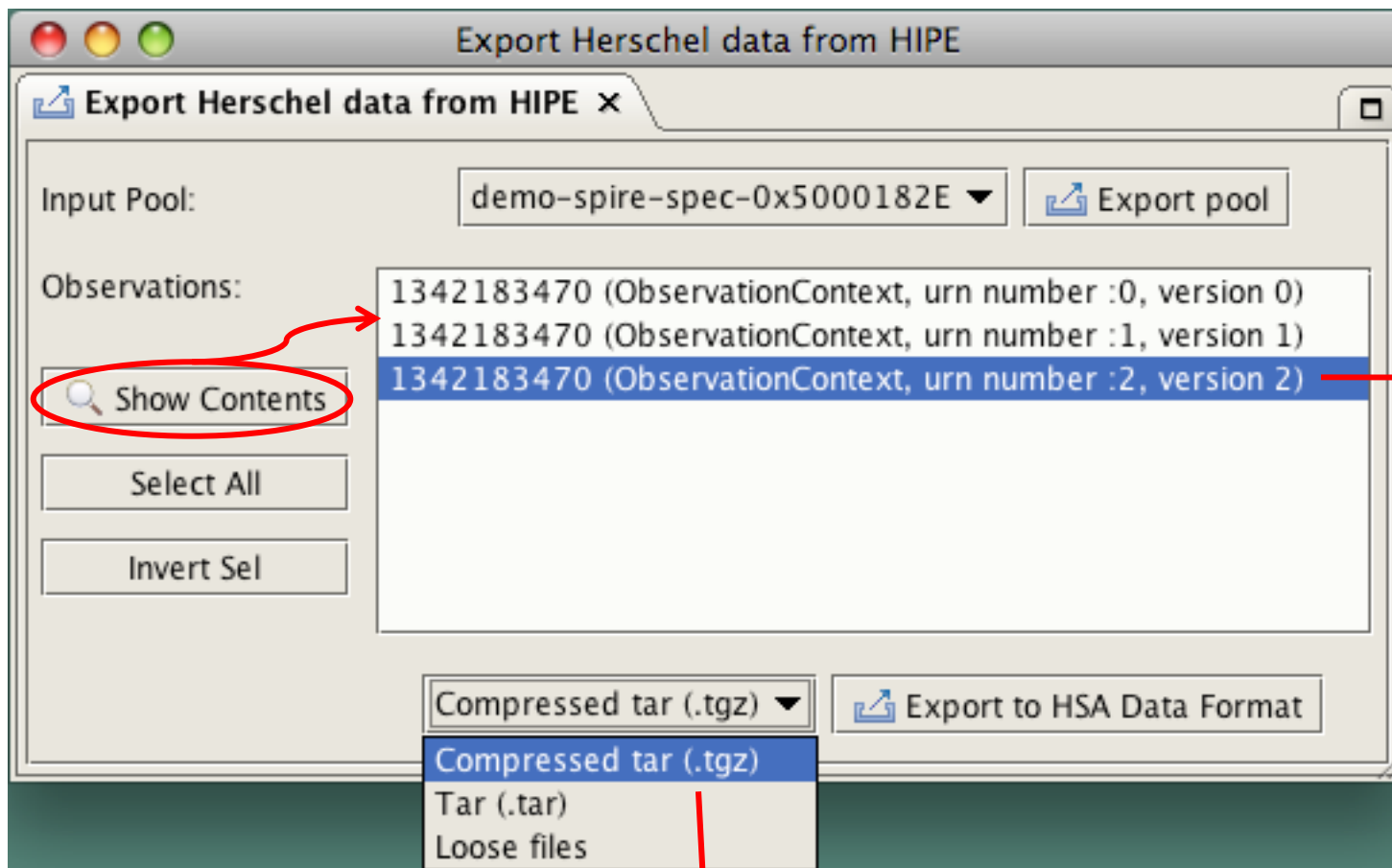
```
# Print the obsids of the found images (it should retrieve the 6 obs of the asteroid 4 Vesta)
# which are the only ones with fluxes above 1 jy.
```

```
i = 0
for ref in refs:
    m = ref.meta
    print str(i),': Obsid',m['obsid'].value
    i += 1
```

Export Herschel Data in pools



- Data saved in pools can be also exported to the tree-like structure equal to the one provided by the HSA:



Choose version

Choose output

HERSCHEL SPACE OBSERVATORY

Summary: Lifecycle of an observation



1. Retrieve the observation(s) from the HUI: Whole or partial
2. Read it into HIPE: `obs= loadObs("path")`
3. Process the observation with your favourite pipeline script
4. Save:
 - a) Individual FITS files: *simpleFitsWriter*
 - b) Replace new files in the same obsContext and save into a pool:
`obs.level2=xxxx`
saveObservation(obs,"pool")
 - c) Save new version of the obsContext into a pool
saveObservation(newobs,"pool")
5. Export obs to tree-like structure: *Export Herschel Data from HIPE*