Herschel Data Processing

Local Store Requirements

Authors: Li Bo, Steve Guest, Maohai Huang

Doc. Ref: SPIRE-RAL-DOC-002576
Date: 9th March 2006
Issue: 1.0

Document Status Sheet

1. Document Title: Herschel Data Processing – Local Store Requirements			
2. Issue	3. Revision	4. Date	5. Reason for change
1	0		Initial issue

Document Change Record

1. Document Title: Herschel User Requirements Document	
2. Document Reference Number:	
3. Document Issue/Revision Number:	
4. Section	5. Reason for change

Table of contents

1. Introduction

This document describes identified requirements of the “Local Store” concept of Herschel Data Processing. This addresses the high-level user requirement that the user should be able to run IA and SPG in a local environment without a database or network connection. The term "local store" was coined by the "Tiger Team" report into common IA to refer to license-free persistent storage on the user’s own computer.

2. Applicable and reference documents

2.1. Applicable Documents:

URD	HCSS User Requirements Document – FIRST/FSC/DOC/0115
DP-UC	HCSS Data Processing Use Case Definitions – HERSCHEL-HSC-DOC-0480

2.2. Reference Documents:

SDP	Specifying Data Products in the Herschel Interactive Analysis System – SPIRE-RAL-DOC-00194
SPG	HERSCHEL DP SPG viewpoint of the DP-SAG, HERSCHEL-HSC-DOC-0658, to be distributed
TTR	Herschel IA Common Framework Architecture Technical Report (“Tiger Team Report”), HERSCHEL-HSC-DOC-0241

3. Terms, definitions and abbreviated terms

API	Application Programming Interface
IA	Interactive Analysis
DP	Data Processing
HCSS	Herschel Common Science System
FITS	Flexible Image Transport System
SPG	Standard Product Generation

"A local store" (non-capitalized) refers to a collection of locally stored data
"Local Store" means the software package

4. Software overview

Local Store addresses the high-level user requirement that the user should be able to run IA and SPG in a local environment without a database or network connection, see **[URD]** requirement HCSS-UR-3.1-1480.

Note that the applicable Use Cases are not specific to Local Store, but generic to SPG, IA etc. These Use Cases are specified in **[DP-UC]**.

4.1. Actors

The Local Store concerns the following main actors or groups of actors:

Interactive and non-interactive User

Astronomer
Calibration Scientist
Instrument Engineer
Developers of DP modules

All actors may wish to execute DP scripts or command-line commands without being connected to the main HCSS database.

4.2. Modes of Use

The following modes of processing are considered:

Explicit use

The actor stores objects locally in local store and runs DP without having to access the main database by choosing the stored objects from local store and loading them into the memory.

Inexplicit use

This mode refers to the use of a local store as a data cache. Here the user uses DP to access objects from the Herschel main database while the local store package stores the objects retrieved from the main database locally. Then the user could run DP unconnected to the main database by having the local store supply the DP tasks with locally stored (cached) objects.

4.3. Environments

The following different environments are considered:

Connected environments

DP is connected to the main database via network while running.

Disconnected environment

DP is not connected to the main database via network while running.

The following scenarios are covered as expected typical local store usage. These scenarios should help to scope the requirements.

1. Calibration analysis. In this scenario there are a number of calibration products, possibly with different versions, and a small number of observations used to generate the calibration products.
2. The local store is used to store all of an observer's private Herschel data.
3. The local store is used to store related observations for a particular purpose, for example a large scale survey.

These scenarios indicate that:

- The local store does not need to meet certain requirements of commercial database systems such as high throughput of transactions.
- It does not need to handle many terabytes (or more) of data. The maximum size supported is expected to be in the region of a few hundred gigabytes. If the implementation works with larger quantities, all well and good, but this is not a requirement.

5. Requirements

5.1. Functional Requirements

HDP-LSTORE-R-1. No network connections or commercial databases should be needed when running a local store.
This is the main requirement that defines a local store. The commercial database referred to here is Versant in particular, though it equally applies to the relational database used at ESAC.

HDP-LSTORE-R-2. Products can be saved in a local store.
The minimum requirement for local store is to be able to handle products. Additionally, data frames need to be supported, and – at least for PACS – telemetry. *However*, it is possible to deal with these cases by reformatting them into products. In other words, no special treatment is required from the local store.

It is possible that a small number of other types (e.g. signature XML definitions) might also require some support in future.

We note that the software should not impose any arbitrary constraints as to what type of object the local store could store unless there are massive gains to satisfy resource requirements.

For more information on products, see **[SDP]** and a subsequent product definition document (when available).

HDP-LSTORE-R-3. Products can be retrieved from the local store.
See the remarks on the previous point. Saving and retrieving could be like this example:

```
> save ("dir1/p1", p1) # write p1 to local store dir1
> save ("dir1/p2", p2) # write p2 to local store dir1
> save ("dir2/p3", p3) # write p3 to local store dir2
> p1 = load ("dir1/p1") # explicit reload
> p1 = load ("dir1", myQuery) # load p1 from local store dir1 subject to query
> p1 = load (".", myQuery) # same as above, but search both dir1 and dir2
```

The last example introduces a concept of a current directory. It is an open issue whether this feature is meaningful and/or desired.

HDP-LSTORE-R-4. Saved objects can be deleted from the local store.
More generally, the user should be able to maintain the contents of the local store.

HDP-LSTORE-R-5. DP scripts should run locally or from a database *without modification*.
This is an SPG requirement. An SPG script should not contain calls like `getLocalStore`. The requirement implies that the local store should be integrated with the "product access layer" which will be defined by DP-SAG later. The SPG requirements document should be referenced here when available.

HDP-LSTORE-R-6. When explicitly retrieving objects from the local store, one or more

meta-data constraints could be specified to choose which object to retrieve.

This identifies the ability to query based on the metadata contained in the products.

HDP-LSTORE-R-7. It should be possible to search data as well as metadata.

This is a data mining requirement.

HDP-LSTORE-R-8. It should be possible to query the store using logical combinations of “and”, “or”, “not”, “=”, “>”, “<”, “>=”, “<=”.

The requirements on querying the store are the ones that are identified as providing added value to the user beyond what is possible with the IO package. See also product browser requirements in SCR-1696.

HDP-LSTORE-R-9. The query of local store should accept a string using a 'query language' in a Jython environment. This language should be consistent with Jython syntax.

Syntactically it might look something like this:

```
# query the products whose creator is 'tiger' and instrument is 'SPIRE'
> results = store.query ('creator=="tiger" and instrument=="SPIRE"')
```

HDP-LSTORE-R-10. Multi-stage queries should be possible.

This means that it should be possible to further refine the results of a query with additional steps. This is useful when an initial query results in a large number of matches. Here is an example:

```
> results = store.query (‘instrument==”SPIRE”’)
> print results.size
100000
> results = store.query (results, 'creator=="tiger"')
> print results.size
5
```

HDP-LSTORE-R-11. The query mechanism should have an API for programming use.

The API should be suitable for both Java and Jython use. Its use might contain features in addition to the user-focused style of the previous point. For example:

```
# make a class to define query condition
from proto.lstore import Predicate as Pre
class ProductPredicate1(Pre):
    def match(self, o):
        if o.creator=="Tiger" :
            return 1;
        else :
            return 0;
# query the local store with the defined predicate class
ss=lstore.query(ProductPredicate1());
```

HDP-LSTORE-R-12. It should be possible for the user to use and maintain multiple local stores.

In other words, the user is not restricted to a single store.

HDP-LSTORE-R-13. Multiple versions of products can be present in a local store.

A local store should be capable of simultaneously storing multiple versions of a product. It should be possible to:

- Retrieve the latest version of a product.
- Retrieve an explicit version of a product.
- Retrieve multiple versions of the same product as the result of a query (*however* the "normal" query behaviour should return only the latest version, i.e. multiple versions in the same result set should occur only on explicit request).

The version could be included as a metadata item, or a part of a file name, or both.

HDP-LSTORE-R-14. It should be possible to store an arbitrary quantity of data in the store.

By "arbitrary" we mean that the implementation should not impose any artificial limit on the data quantity. We note again the earlier remark that the local store is not expected to duplicate the functionality of large commercial databases. Users might want to have quite a lot of data on their disk, many gigabytes in some cases. We expect that a user should be allowed to fill up the hard disk on their laptop, if they so desire, with the contents of a local store. As an example which might violate this requirement, if all the data were stored in a single file, the store might run into trouble when it gets to a certain size, e.g. file system limits, often around 2GB. A user might also want to write some saved data to a memory stick.

HDP-LSTORE-R-15. The user should be able to store the data in local store in a familiar format, e.g. FITS.

This refers to the persistent format in which the products are stored, i.e. the underlying data storage mechanism. It is unclear whether this is a genuine requirement. There are even some concerns about the implications of doing so (but see the following requirement on integrity). However, the typical user might well prefer it, or indeed, *require* it. The DP-UG might wish to discuss this. The format could even be configurable.

HDP-LSTORE-R-16. It must be possible to maintain the integrity of a local store.

Note that this requirement is less stringent than a statement like "the integrity of a local store must be maintained at all times". For example, if the file system is used as implementation then using a separate index table is acceptable if there is utility to synchronize them. This means that if the integrity of the store is corrupted, e.g. by deleting a file without removing the corresponding entry in a lookup table, then a tool must be available to easily regenerate and correct the tables.

HDP-LSTORE-R-17. It is unacceptable for an end user to have to perform anything more than minimal database administration.

This requirement should be clarified to be more specific, but its purpose is to state that the user should not have to assume the role of a database administrator. For example, it is acceptable to have to change a property to use a local store instead of a database, but not to have to manually perform schema evolutions. This requirement needs to be taken into account when choosing the underlying data storage mechanism.

5.2. Performance Requirements

HDP-LSTORE-R-18. The performance of a query on the local store should be "good enough".

Performance requirements are to be understood to apply to a “typical” modern laptop, e.g. Sony Vaio with 1.6Ghz Centrino chip and 1GB of RAM (i.e. SG’s).

A metadata query over 1000 products should take less than 5 seconds. It is desirable to be under 1 second.

A full (data mining) query of 1000 products each containing 1MB of an Int1d array (250,000 elements) should take less than 2 (TBC) minutes (i.e. 2 hours to search a 60GB disk).

5.3. Design requirements and implementation constraints

HDP-LSTORE-R-19. The Local Store is a module distributed as part of DP.

Its implementation should satisfy DP package requirements, such as regarding use of third party packages, or the ability to freely distribute it as part of the build.

HDP-LSTORE-R-20. A Local Store deliverable consists of documentation, executable code, and test-harness.

This requirement is a general one for software components within the HCSS.

HDP-LSTORE-R-21. The local store implementation should be compatible with and usable by the “product access layer”.

The product access layer documentation should be referenced here when available.

HDP-LSTORE-R-22. It should be reasonably easy/not too hard to change the format of data in local store i.e. it should be pluggable.

This is a software requirement on the design, anticipating changes in desired data formats.

HDP-LSTORE-R-23. It should be possible to optimize the query implementation.

This is a general requirement for queries in the Herschel system.

HDP-LSTORE-R-24. It should be possible to plug in a “fuzzy logic” query.

This requirement anticipates the possibility of needing to find a “best” match for calibration data if the set of available data is incomplete. This facility is known as “find best”. The actual implementation of such an algorithm is outside of the scope of the local store WP.

HDP-LSTORE-R-25. Objects returned by local store queries should contain the size of the result set.

This requirement states that simple iterators are not adequate. We also note that collections might provide a richer functionality than iterators, as proposed in SCR-0856. Strictly, the results of a query should be a *set* i.e. a collection where each entry is unique.

6. Implications and open issues

6.1. Local store query framework

The calibration product browser requirements are also relevant if that tool is to work with a local store.
Querying the local store should use the same query framework as querying the main database.

6.2. ObjectStore

Originally, an implementation of the herschel.store.api.ObjectStore interface was considered. If the requirements are restricted to products, then this does not obviously provide much in the way of benefit.

6.3. Query front end

Is a custom parser for the query command needed? Can the Jython interpreter be used directly? Could other tools be used for this? Will the “data access layers” provide generic query handling support? Note that **HDP-LSTORE-R9** mandates Jython syntax.

6.4. Current directory

Is the concept of a current directory meaningful and/or desired? It implies a clear mapping between the local store and the file system. This might be intuitive and therefore beneficial to the user, but could have the drawback of limiting the implementation possibilities.

7. Related Topics

7.1. Calibration Data Access

Any framework for calibration data access should be compatible with, and work with, a local store.

7.2. Data Access Layer

A more generalized framework for a data access layer, as may be required for SPG, should be compatible with, and work with, a local store (and vice versa).

7.3. Product Browser

An implementation of a product browser (sometimes referred to as “calibration product browser”) should work with a local store. See also SCR-1696.

8. Traceability