



HERSCHEL

SPIRE

Ref: **SPIRE-RAL-NOT-002829**

Author: **Asier Abreu Aramburu**

Page: **1**

Issue: **Draft 0.1**

Date: **12th Feb 2007**

SPIRE Database Management and Configuration Control

Prepared by:	Asier Abreu Aramburu	Date 12/02/07
Checked by:		Date
Approval:		Date



Table of Contents

Table of Contents.....	3
1 Introduction	4
1.1 Intended Audience.....	4
1.2 Scope.....	4
1.3 Applicable Documents.....	4
1.4 Document Change Record.....	4
2 Assumptions	5
3 Database Management.....	5
3.1 Database Creation.....	5
3.1.1 System/Software Requirements.....	5
3.1.2 Creating a database	6
3.1.3 Creating a new configuration on an existing database	12
3.2 Database Deletion	13
3.2.1 Stopping database transactions.....	13
3.2.2 Removing the database	13
3.3 Database backup and restore process.....	13
3.3.1 Backing up an existing database.....	13
3.3.2 Restoring a database from a backup file.....	14
4 Configuration Control.....	15
4.1 Configuration Control Rationale	15
4.2 Configuration Control Approach	15
4.2.1 The CVS SPIRE repository structure	15
4.2.2 Versioning of definitions	16



1 Introduction

This document provides information on various topics related to database management, CUS usage and configuration control of the CUS and test control/ccs template definitions.

1.1 Intended Audience

The potential users of this document are those persons directly related with the SPIRE testing on its different scenarios (ILT, AVM, IST) which will be involved in the creation of the test setup.

The test setup is understood as the ‘system’ that allows the operation of the SPIRE instrument in real or semi-real time, the monitoring of its behaviour and the archiving of the telemetry produced by the instrument as defined in AD01.

1.2 Scope

This document is intended to be used on those test environments where a database is required for generation of the uplink sequences used to operate SPIRE, i.e, ILT, IST, HSC.

1.3 Reference Documents

Ref#	Title	Reference	Issue Nb	Date
RD01	Database Administration Procedures	HSC-HERSCHEL-DOC-0833	0.4	07/11/06
RD02	HCSS Installation Instructions	HSC-HERSCHEL-DOC-0427	1.18	12/10/06

1.4 Applicable Documents

Ref #	Title	Author	Reference	Date

1.5 Document Change Record

Date	Issue Nb.	Affected Pages	Changes
12/02/07	Draft 0.1		Document Created



2 Assumptions

Despite ALL SPIRE test environments are run under Linux systems, no assumptions are made here as to the expertise in Linux environment of the person reading this procedure, i.e. ,you don't need to be a Linux guru to understand it. Nevertheless it would help if you know something about Linux.

3 Database Management

3.1 Database Creation

This section specifies how to create a versant database using the available versant and HCSS functionality.

3.1.1 System/Software Requirements

Operating system:

Note: all given URL addresses will prompt for a user name and password. Please ask Steve Guest or Sunil Sidher for this information.

- The operating system requirements are mainly those imposed by the HCSS and Versant installation requirements, which can be found in RD01 and RD02
 - <http://www.rssd.esa.int/SD-general/Projects/Herschel/hscdt/releases/doc/Install.html>
 - <http://www.rssd.esa.int/SD-general/Projects/Herschel/hscdt/releases/doc/versant/dba-procedures/index.html> respectively
- On Linux systems the key parameters for database creation (and in general for all HCSS required operations) can be located in the *user.props* file located under `${USER_HOME}/.hcss` directory
- On Windows systems the key parameters for database creation (and in general for all HCSS required operations) can be located in the *user.props* file located under `%USER_PROFILE%/.hcss` directory

The following must be installed on the system were the database is to be created:

Third parties software:

- Versant ODBMS (v7.0.1.1 or higher)
- Java Run Time Environment (v1.5.0 or higher)

Note: this is the default current (13/02/07) installation on both the ILT setup at RAL and the IST setup machines.



SPIRE specific:

- SPIRE Build (#478 or higher) (or equivalent formal release). It is assumed that the SPIRE release contains the correspondent HCSS release. The SPIRE latest build can be retrieved @ http://www.rssd.esa.int/herschel_scripts/hscdt/ccInfo2Html.cgi?theProject=SPIRE&theDisplay=BUILD

3.1.2 Creating a database

The following steps specify how to create a new database on a test environment and the expected results of each of the actions performed. The commands should be typed in a terminal of the database server's. For a full description on db management refer to RD01.

3.1.2.1 Allocating the physical space:

Command line: `db_admin -i db_name@db_server`

Expected Result:

```
02-Feb-07 09:43:42.852 LogInitialiser: Initialising using local log configuration file
/home/sg55/hcss/userlogging.properties
02-Feb-07 09:43:42.942 Configuration: Build number is 1106
Initializing new database system...
[makedb] New database directory created: db_name@db_server
[profile.be] Updated profile.be for database: db_name@db_server
[createdb] New database created: db_name@db_server
[SchemaTool]
[indexing] Creating default indices.
02-Feb-07 09:43:50.200 DatabaseTools: Could not check for multi-user-mode, check is
disabled.
[DBI]
[initv]
Initializing database "db_name@db_server"
02-Feb-07 09:43:50.518 ReplicatingTransaction: Initializing normal Store:
db_name@db_server (main), shadowing disabled.
Initializing database using herschel.ccm.tools.BasicMissionInitializer
02-Feb-07 09:43:50.643 InstrumentModelRegistryImpl: Create new model registry for
instrument HIFI
02-Feb-07 09:43:50.738 InstrumentModelRegistryImpl: Create new model registry for
instrument PACS
02-Feb-07 09:43:50.739 InstrumentModelRegistryImpl: Create new model registry for
instrument SPIRE
02-Feb-07 09:43:50.906 ReplicatingTransaction: Initializing normal Store:
db_name@db_server (main), shadowing disabled.
Database system initialization finished, system id = 794.
Finished successfully.
```



3.1.2.2 Adding public access to the database:

Command line: *dbuser -add -P db_name@db_server*

Expected Result:

*VERSANT Utility DBUSER Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation*

3.1.2.3 Telemetry Packet Objects index sorting by time:

Command line:

dbtool -index -create -btree herschel.versant.ccm.TmSourcePacketImpl _time db_name

Note: there is no error, the @ part on the db is not necessary

Expected Result:

*VERSANT Utility DBTOOL Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation*

Creating btree index on herschel.versant.ccm.TmSourcePacketImpl._time...

3.1.2.4 Creating a new configuration for this database:

This step is required to associate all changes done to the contents of a database. The philosophy is that a new database will start with configuration config1 and if changes are then made to definitions on the database or calibration tables a new configuration will be required for those changes to have effect. See section 3.1.3 for more information on this.

Command line: *missetup -addconfig config1*

Expected Result:

*02-Feb-07 09:58:32.639 LogInitialiser: Initialising using local log configuration file
/home/sg55/.hcss/userlogging.properties
02-Feb-07 09:58:32.687 Configuration: Build number is 1106
02-Feb-07 09:58:33.776 ReplicatingTransaction: Initializing normal Store:
db_name@db_server (main), shadowing disabled.
02-Feb-07 09:58:33.852 MissionSetup:
Using database "db_name@db_server"
02-Feb-07 09:58:33.947 MissionSetup: Mission configuration "config1" created in
database "db_name@db_server"*



3.1.2.5 Ingesting the MIB into the database:

It is necessary to check the current settings for MIB ingestion before ingesting the MIB. The property which specifies where the MIB files to ingest are, is ***hcss.mib.datadir***. This can be found on the *user.props* file under $\${USER_HOME}/.hcss$. Make sure it is pointing to the directory you want.

Command line: *ingestmib -settings | grep 'mib'*

Expected Result:

```
02-Feb-07 10:04:33.179 LogInitialiser: Initialising using local log configuration file
/home/sg55/.hcss/userlogging.properties
02-Feb-07 10:04:33.224 Configuration: Build number is 1106
+hcss.mib.dumprawmib.stdout=false (Package default property, source:
herschel/mib/defns/mib.xml)
+hcss.mib.preparemib.outdir=. (Package default property, source:
herschel/mib/defns/mib.xml)
+hcss.binstruct.mib.source=ascii (Package default property, source:
herschel/binstruct/defns/binstruct.xml)
+hcss.mib.logfile=mibchecker.log (Application default property, source:
IngestMib.defaults)
+hcss.mib.readallcmds=true (Package default property, source:
herschel/mib/defns/mib.xml)
+hcss.binstruct.mib=/spires/hcssbld/spire/lib/hcss/data/binstruct/mib (Package default
property, source: herschel/binstruct/defns/binstruct.xml)
+hcss.mib.database=db_name@db_server (Application default property, source:
IngestMib.defaults)
+hcss.binstruct.mib.pal.database=database@server (Package default property, source:
herschel/binstruct/defns/binstruct.xml)
+hcss.binstruct.mib.pal.poolname=mibs (Package default property, source:
herschel/binstruct/defns/binstruct.xml)
+hcss.mib.preparemib.indir=. (Package default property, source:
herschel/mib/defns/mib.xml)
+hcss.mib.tabledefs=/spires/hcssbld/spire/lib/hcss/data/mib/defns/table-defns/
(Application default property, source: IngestMib.defaults)
+hcss.mib.datadir=/home/sg55/PFM5/MIB/ (User global property, source:
/home/sg55/.hcss/user.props)
+hcss.binstruct.mib.pal.tm_version_map=default (Package default property, source:
herschel/binstruct/defns/binstruct.xml)
+var.mib.datadir=/home/sg55/PFM5/MIB/ (User global property, source:
/home/sg55/.hcss/user.props)
+hcss.mib.instrument=SPIRE (Application default property, source:
IngestMib.defaults)
+hcss.mib.cus_file=cus.script (Application default property, source:
IngestMib.defaults)
+hcss.mib.tc_command_list=/home/sg55/PFM5/MIB//auxil/tcmds (Package default
property, source: herschel/mib/defns/mib.xml)
+hcss.mib.erroronly=false (Application default property, source: IngestMib.defaults)
```




+*hcss.mib.tc_command_durns=/spires/hcssbld/spire/lib/hcss/data/mib/example-mibs/example-1/auxil/tc-durns* (Application default property, source: *IngestMib.defaults*)

Command line: *ingestmib MIB_LABEL* (where *MIB_LABEL* is a user specified label)

Expected Result: (only the last lines of the output are copied here as it is too long)

If there is an error:

There was one or more serious problems in the semantics of the imported MIB files. Check the file mibchecker.log for details.

Exception in thread "main" herschel.mib.utils.ValidationProblemsFoundMibException: One or more serious semantical validation problems encountered.

at herschel.mib.api.MibFactory.createDefinitions(MibFactory.java:411)

at herschel.mib.api.MibFactory.createDefinitions(MibFactory.java:93)

at herschel.mib.tools.IngestMib.exec(IngestMib.java:65)

at herschel.mib.tools.IngestMib.main(IngestMib.java:100)

This will indicate an error on the MIB files themselves and therefore nothing you can't do unless you have some insight onto MIB file correction. If this is not the case Sunil Sidher should be able to help you. A mibchecker.log file will have been created on the directory were the ingestmib tool has been run.

If everything went well:

MibFactory: There was one or more problems in the imported MIB files. These are not judged to be serious. Check the file mibchecker.log for details.

02-Feb-07 12:16:24.187 MibFactory: Semantical checks done.

02-Feb-07 12:16:28.120 IngestMib: MibDefinitions now added to database with label SPIRE_MIB_2.2.G1

02-Feb-07 12:16:28.127 IngestMib: Ingestion Completed Successfully.

Note: despite the reference to "...one or more problems..." this does not stop the MIB from being ingested and can be used normally.

3.1.2.6 Open the cusgui and load this MIB:

Command line:

1. *cusgui&*
2. **When the gui appears hit tab labelled Mib then hit the load new mib from the drop down menu.**
3. **From the pop up window select the MIB labelled *MIB_LABEL***

Result:

Ingested MIB commands appear on the MIB commands subwindow of the cusgui (on the bottom right)



3.1.2.7 Ingesting required calibration tables onto the database:

Command line: `./UploadCaltables.csh`

Note: the “./” (dot-slash) is important

Result: (only contains the output of the first table loaded for reference)

```
Using database db_name
VERSANT Utility DBTOOL Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation
```

Volume 0:

```
Sysname "sysvol" Size: 8387584K
Pathname "/spired/versant/db/pfm5_test/system"
Percentage of volume space free in sysvol : 99%
Free space in vol sysvol : 8361344KB
```

```
Percentage of free space in DB : 99%
Total available free space in DB : 8361344KB
```

```
02-Feb-07 12:26:23.921 LogInitialiser: Initialising using local log configuration file
/home/sg55/.hcss/userlogging.properties
02-Feb-07 12:26:23.978 Configuration: Build number is 1106
02-Feb-07 12:26:24.962 ReplicatingTransaction: Initializing normal Store:
db_name@db_server (main), shadowing disabled.
02-Feb-07 12:26:26.057 Cus: Calibration table "Flash.txt" successfully uploaded from
"/home/sg55/PFM5/CUS/temp/Caltables/Flash.txt"...
...
```

one “successfully uploaded from ...” message for each calibration table

3.1.2.8 Ingesting the CUS definitions:



It is assumed that a single file containing all latest CUS definitions (building blocks, procedures and modes) is available at CVS for download.

There are two ways to ingest the CUS definitions, through the cus gui or from command line

Using cusgui:

1. *cusgui*
2. Press the tab labelled Registry then Import and on the pop-up window selected the file that contains all the definitions that are to be uploaded.
3. Then committed the changes pressing the Registry, then Commit window and in the pop-up window insert a comment.

Command line: *./IngestCUSDefinitions.csh*

This shell script is provided with the checked out CVS repository and it executes a cus command to import the same file as specified on the previous step.

3.1.2.9 Setting the initial obsid for this database:

Command line: *java SetObsId db_name@db_server start_obsid*

Note: the start_obsid should be given as the hexadecimal value of the start obsid without the '0x' indicator, i.e., if the initial obsid to set is 0x30012001 the command should look like: *java SetObsId db_name@db_server 30012001*

Expected Result:

```
02-Feb-07 12:40:26.714 LogInitialiser: Initialising using local log configuration file
/home/sg55/.hcss/userlogging.properties
02-Feb-07 12:40:26.763 Configuration: Build number is 1106
02-Feb-07 12:40:27.522 ReplicatingTransaction: Initializing normal Store:
db_name@db_server (main), shadowing disabled.
02-Feb-07 12:40:27.637 IdRegistryCommitThread: Creating new IdRegistry entry
idreg_Observation
```

3.1.2.10 Associating ALL of the previous (MIB + CUS definitions + eventually test control scripts) by creating and instrument model on this configuration:

Command line: *cus -createinsmodel config1*

Expected Result:

```
02-Feb-07 12:54:31.063 LogInitialiser: Initialising using local log configuration file
/home/sg55/.hcss/userlogging.properties
02-Feb-07 12:54:31.115 Configuration: Build number is 1106
02-Feb-07 12:54:31.968 ReplicatingTransaction: Initializing normal Store:
db_name@db_server (main), shadowing disabled.
02-Feb-07 12:54:32.096 MissionImpl: Using configuration "config1"
```



3.1.2.11 Checking the currently available instrument models in the database:

Command line: *missetup -listmodels*

Expected Result:

```
02-Feb-07 09:59:12.860 LogInitialiser: Initialising using local log configuration file
/home/sg55/.hcss/userlogging.properties
02-Feb-07 09:59:12.907 Configuration: Build number is 1106
02-Feb-07 09:59:13.883 ReplicatingTransaction: Initializing normal Store:
db_name@db_server (main), shadowing disabled.
02-Feb-07 09:59:13.957 MissionSetup:
Using database "db_name@db_server"
02-Feb-07 09:59:13.968 MissionSetup:
Instrument models :-
SPIRE PFM5
PACS WHOCARES
HIFI WHOCARES
```

3.1.3 Creating a new configuration on an existing database

The following steps specify how to create a new configuration on an already existing database

1. If ccshandler (IST) or testcontrolserver (ILT) applications are running, stop them, by pressing Ctrl-C on the correspondent terminal. If not go to step 2.
2. In order to know which configuration number to create, hit the up arrow in order to get the latest missetup command with the correspondent config **or** edit the *user.props* file located under $\{\text{USER_HOME}\}/.hcss$ and look for *hcss.ccm.mission.config* property. Then on a terminal type: *missetup -addconfig confign* (where n the correspondent incremental number). Wait for completion, then go to step 3.
3. On a terminal type: *cus -createinsmodel confign*
4. On the *user.props* file located under $\{\text{USER_HOME}\}/.hcss$ update the *hcss.ccm.mission.config* property to the newly create configuration *confign*
5. Restart ccshandler (IST) or testcontrolserver (ILT) applications.



3.2 Database Deletion

The following steps describe how to delete an existing database.

Note: This action should only be performed if you REALLY know what you are doing.

3.2.1 Stopping database transactions

Command line: *stopdb -s db_name@db_server*

Expected Result:

*VERSANT Utility STOPDB Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation*

3.2.2 Removing the database

Command line: *removedb -rmdir db_name@db_server*

Note: you will be prompted whether you want to go ahead

Expected Result:

*Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation

Are you sure you want to remove the database "db_name@db_server" (y/n) ? : y*

3.3 Database backup and restore process

The following steps describe how to backup an existing database and restore it afterward

3.3.1 Backing up an existing database

1. **Command line:** *stopdb -s db_name@db_server*

Expected Result:

*VERSANT Utility STOPDB Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation*

2. **Command line:** *dbinfo -l dbname@db_server*

Expected Result:

*VERSANT Utility DBINFO Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation*



Database is in DBA-only and single connection mode ...

3. **Command line:** *vbackup -device devicename -backup db_name@db_server*

Expected Result:

*VERSANT Utility VBACKUP Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation*

Backing up database `db_name@db_server' to device `devicename':

```
0%          50%          100%  
| | | | |  
.....
```

Backup has completed successfully.

Note: this last message may take some time depending on the size of the database you are trying to backup

3.3.2 Restoring a database from a backup file

1. Create directory to hold the database volumes

Command line: *makedb restoredb_name@db_server*

Expected Result:

*VERSANT Utility MAKEDB Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation*

2. Restoring the database

Command line: *vbackup -device 'devicename' -restore restoredb_name@db_server*

Expected Result:

*VERSANT Utility VBACKUP Version 7.0.1.1
Copyright (c) 1989-2005 VERSANT Corporation*

Restoring database `backedup_db@chichester' from device `temp.bck':

```
0%          50%          100%  
| | | | |  
.....
```

Restore has completed successfully.

*Would you like to do another level of restore
on database `backedup_db@chichester'? [default = no]
Cleaning up...*

4 Configuration Control

4.1 Configuration Control Rationale

The rationale behind creating and maintaining a configuration control of the software used during the testing of the SPIRE instrument on any environment was/is to be able to know at some point in the future, which version of which script was used to produce a particular set of SPIRE telemetry data.

4.2 Configuration Control Approach

4.2.1 The CVS SPIRE repository structure

With this in mind, the SPIRE software used for different campaigns has been maintained on the ESA CVS repository:

http://www.rssd.esa.int/herschel_scripts/cvsweb.cgi/?cvsroot=Herschel_CVS under the following directory: **develop/main/herschel/spire/instrument/**

Note: Access can be obtained to this repository by asking either Steve Guest or Sunil Sidher. The user and password cannot be revealed on this document for security reasons.

The tree structure under that directory was created so that a folder would exist for each test campaign, as it can be seen on the snapshot taken to the WinCVS folders:

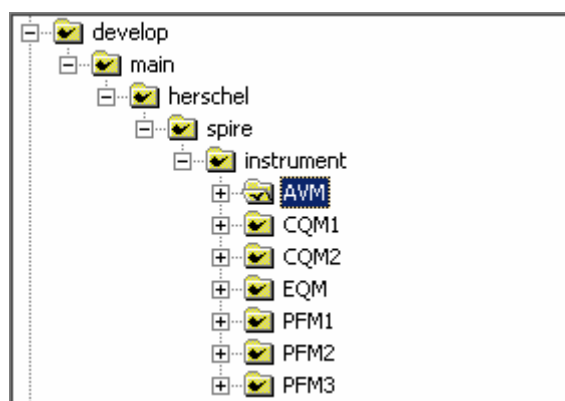


Figure 1. CVS SPIRE repository tree structure

For the ILT level campaigns each of these folders contains two subfolders: CUS and TestControl in which the CUS definitions and the TestControl scripts are located respectively.

For the IST (AVM) level campaigns each of these folders contains two subfolders: CUS and CCS in which the CUS definitions and the CCS templates are located respectively.



4.2.2 Versioning of definitions

Note: this is not a cvs manual, only a guideline on how the versioning has to be done and therefore not every single step for using CVS will be explicated here. Useful information regarding CVS and WinCVS can be found on the links appearing below:

http://sourceforge.net/docman/display_doc.php?docid=766&group_id=1

And

<http://www.thathost.com/wincvs-howto/>

Note: script refers to both CUS definitions and testcontrol /ccs templates

4.2.2.1 *Checking in scripts onto CVS*

See: <http://www.thathost.com/wincvs-howto/#startingnew>

Prior to the start of a new campaign the definitions that have been verified to be correct using a dummy database on an alternative test setup have to be configured on CVS.

If the definitions did not exist they will have to be imported, if they existed they are checked in onto the correspondent folder in the CVS repository.

This can be done easily by using WinCVS by selecting all new scripts then adding them and then committing the changes by inserting a comment when prompted.

This process may require the creation of a new folder on the CVS repository, a new folder is just as every other file on CVS so the same process of 'creation' is applicable for it.

This step will automatically assign initial versions to those definitions.

Once they are checked in, a tag must be assigned to these versions so that each definition has a unique version number + TAG.

4.2.2.2 *Creating a local repository*

See: <http://www.thathost.com/wincvs-howto/#gettingnew>

These tagged versions then have to be checked out onto a 'local repository' on the test database server or a networked machine that has access to the database server. This local repository will have the same structure once checked out than the CVS repository and all versioned + tagged definitions.

See: <http://www.thathost.com/wincvs-howto/#tag>

Then, following the steps described on section 3 above we create a test database.

4.2.2.3 *Configuration control after the test is started*

Because there is the possibility that things (scripts) can change when they start being used for one reason or another, some approach must be taken to the configuration control of these changes.

The HCSS provides a good enough mechanism for this.



Every definition or calibration table that is ingested onto a database gets automatically assigned a version (1). Therefore, if our script had an original cvs version of let's say 1.2 on the repository, when it gets ingested onto the db, its full version becomes 1.2.1.

From then on any changes made to this script must be **ONLY** tracked at the database through the incremental versions automatically assigned by the db whenever a change is done to this script and then committed.

The reason that this versioning should only be tracked this way is that experience says that trying to maintain a versioning system using new versions checked in and out every time a change is made can become highly unstable (i.e. a nightmare) on a system that uses hundreds of different definitions.

When the tests are finished all latest versions of scripts residing on the database will be checked in on the CVS repository and this will assign them a new version (1.3 following the previous example). A tag should also be assigned to these new versions.

Since the introduction of the mission configuration concept a set of scripts + calibration tables ingested at one point on a database must be assigned a correspondent configuration. When creating a new configuration this effectively creates a 'snapshot' of the database at that particular moment that freezes those versions and therefore any 'new' changes performed to scripts since the config was created require a new configuration.

In this scenario the concept the automatic versioning of the database still holds as different versions of script will correspond to different configurations and this does not affect the tracking of versions by the database.