



European Organisation for Astronomical Research in the Southern Hemisphere
Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral
Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

Data Management and Operations Division

Database Content Management Handbook

Doc. No.: VLT-MAN-ESO-19300-4538

Issue: 1

Date: 9 May 2008

Prepared A. Dobrzycki

Name	Date	Signature
------	------	-----------

Approved M. Romaniello

Name	Date	Signature
------	------	-----------

Released F. Comerón

Name	Date	Signature
------	------	-----------

This page was intentionally left (almost) blank

Change record

Issue	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1	9 May 2008	All	First release

Contents

Change record	3
1 Introduction	7
1.1 ESO data flow	7
1.2 Scope	8
1.3 Referenced documents	9
1.4 Acronyms/abbreviations	9
2 Hardware; computer access	12
2.1 Machines used by DBCM and their roles	12
2.2 Database servers	13
2.3 User accounts	13
2.3.1 Login accounts	13
2.3.2 Database accounts	13
3 ASE databases	15
3.1 The opc70 database	15
3.1.1 Inserting data	17
3.1.2 Maintenance	17
3.2 The user_portal database	17
3.2.1 Inserting data	18
3.2.2 Maintenance	18
3.3 The obrep2 and obrep2_hist databases	18
3.3.1 Inserting data	18
3.3.2 Maintenance	19
3.4 The observations database	19
3.4.1 The data_products table	19
3.4.2 Instrument Specific Tables (ISTs)	22
3.4.3 The isaac_stacks table	24
3.4.4 The dp_headers table	24
3.4.5 Header display	25
3.4.6 The olas_ins table	25
3.5 The qc1 database	26
3.5.1 Inserting data	26
3.5.2 Maintenance	27
3.6 The library database	27
3.6.1 Inserting data	27
3.6.2 Maintenance	27
3.7 The ambient database	27



- 3.7.1 Inserting data 27
- 3.7.2 Maintenance 27
- 3.8 The asto database 28
 - 3.8.1 Inserting data 28
 - 3.8.2 Maintenance 28
- 4 SAFIQ databases 29**
 - 4.1 Keyword repository 29
 - 4.1.1 Inserting data 29
 - 4.1.2 Maintenance 30
 - 4.2 Operation logs 30
 - 4.2.1 Inserting data 30
 - 4.2.2 Maintenance 30
- 5 DBCM Portal task tracking system 31**
- 6 DBCM on the World Wide Web 33**
- 7 DBCM software 34**
 - 7.1 Software development 34
 - 7.1.1 Software location 34
 - 7.1.2 Version control 35
 - 7.1.3 The development → integration → production S/W cycle 35
 - 7.2 Headers-On-The-Fly (HOTFly) 36
 - 7.3 File manipulation tools. 36
 - 7.4 Database update tool 36
 - 7.5 Scripts used for scheduled maintenance 38
 - 7.5.1 Non-standard access rights 38
 - 7.5.2 Hiding sensitive data 39
 - 7.5.3 Looking for new instruments 39
 - 7.5.4 Managing HARPS data 39
 - 7.5.5 Managing the library..bibPublic table 40
 - 7.5.6 Daily ADS report for the ESO Librarian 40
 - 7.5.7 Managing WFCAM access rights 41
 - 7.5.8 Monitoring data_products 41
 - 7.5.9 Comparison of data_products with ISTs 41
 - 7.5.10 Comparison of NGAS with data_products 42
 - 7.5.11 Filling /arc/headers directory 42
 - 7.5.12 Filling dp_urls and ambient parameters in ISTs 42
 - 7.5.13 Filling isaac_stacks table 42
 - 7.5.14 Filling keyword_repository with headers of raw files 43
 - 7.5.15 Filling keyword_repository with headers of processed files 43
 - 7.5.16 Garching ↔ Chile database replication 43
 - 7.5.17 Monitoring Garching → Chile replication 44
 - 7.5.18 Monitoring Chile → Garching replication 44
 - 7.5.19 Filling ops_logs with operation log records 44



8	What's special about...	45
8.1	... APEX?	45
8.2	... HARPS?	46
8.3	... GROND?	47
8.4	... WFCAM?	47
8.5	... Brazilian data from the La Silla 2.2-m telescope?	47
9	Common non-scheduled tasks	48
9.1	How to synchronise Chile and Garching databases?	48
9.1.1	Front-end databases	48
9.1.2	Back-end databases	50
9.2	What to do when new instrument is identified?	50
9.3	How to deal with a non-scientific instrument?	51
9.4	How does an instrument become "officially supported" by the ESO Archive?	51
9.5	How to access databases in Chile?	51
9.6	How to prepare a data package?	52
9.7	How to prepare HARPS GTO files with time information removed?	53
9.8	Maintenance of the qc1 database.	55
9.8.1	Request for a new table in qc1 database	55
9.8.2	Request for table modification in qc1 database	56
9.8.3	Maintenance of the qc1 web application	56
9.9	Merging user information in databases	57

Chapter 1

Introduction

ESO is the European Organisation for Astronomical Research in the Southern Hemisphere. ESO collects astronomical data at three major astronomical sites in Chile: the Very Large Telescope (VLT) on Cerro Paranal, several medium sized optical telescopes on La Silla, and a sub-millimetre APEX telescope.

1.1 ESO data flow

The ESO data flow can be summarised as follows:

Step 1 (also called Phase I): ESO issues *Calls for Proposals* (CfP) twice a year. Astronomers are encouraged to submit *proposals* – descriptions of scientific programmes they intend to perform with the use of an ESO telescope. All proposals are reviewed by the Observing Panel Committee (OPC). The database called *opc70* contain proposal information.

The VISiting Astronomer Department (VISAS) collects observation proposals from astronomers and organises the OPC. Two observation modes are offered: the Visitor Mode (VM), where the author of an accepted programme travels to the mountain to oversee execution of his/her programme or the Service Mode (SM), where the data are taken by the ESO staff and then sent to the programme's Principal Investigator (PI).

Step 2 (also called Phase II): For SM programmes only, the PI of each accepted proposal provides all information regarding the objects/targets they want to be observed. Observation Blocks (OB) are created for each target. The database containing information on the Observation Blocks, filled in Garching during Phase II, is called *obrep2*. The User Support Department (USD) collects observation details from the PIs using the Phase 2 Preparation tool (*p2pp*) for service mode programmes and prepare the observing queue for those programmes. Information from *obrep2* database is replicated to the observatories.

Step 3: Observations are performed at the telescopes as scheduled by VISAS (phase I). All information related to the actual observations is stored in the observations database. This information is first stored on the database servers on the mountains and then replicated to Garching.

Weather information and parameters describing the atmospheric conditions are stored in the ambient database.



Step 4: Observational data are transferred to Garching and archived in the NGAS system. The archive and ngas databases contain all information relevant to the location of the data.

Once the data arrive in Garching (typically a few days after the observations), Quality Control (QC) scientists process the data. The principal outcome are *master calibration frames*, *reduced science frames* and the so-called *ancillary frames* (various processing-related files wrapped into a FITS file). The calib database contains information on those files.

The Science Archive Operations team delivers the data to the PIs, using a variety of methods (e.g. DVDs, LAN, FTP).

The QC Team also generates the so-called QC parameters (information on the status of various instrument components). Those parameters are stored in the qc1 database.

The PI has the sole right to the scientific data for a *proprietary period* (under normal circumstances it is one year). After the proprietary period has expired, the data become available to the entire astronomical community. The users can request the data from the ESO Archive via the Archive user interfaces.

Step 5: Astronomers publish scientific papers based on the data obtained with ESO telescopes. Those papers are recorded by the ESO librarians. The library database contains information on scientific papers.

Databases from steps 1 and 2 are collectively known as the *Front End* databases. Databases from steps 3-5 are known as the *Back End* databases.

For detailed description of the ESO Data Flow, see [1].

The users access ESO services through accounts on the *ESO User Portal*. The database storing user information necessary for accessing the Portal is called `user_portal`.

1.2 Scope

This document describes the procedures utilised by the Database Content Management (DBCM) team. The DBCM team is part of the Data Flow Operations Department within the Data Management and Operations Division.

Team's duties can be summarised as follow:

- Serving as database owner (DBO) for all ESO operational databases.
- Maintenance of the databases, via one-time and regular updates, cron jobs, etc.
- Verification of the consistency of the content of the front-end databases in Garching, La Silla and Paranal.
- Verification of the contents of the back-end databases.
- Preparation of non-standard data packages for release (e.g. instrument commissioning, science verification).



1.3 Referenced documents

- [1] ESO. *VLT On-line Data Flow Requirements Specification*, VLT-SPE-ESO-19000-749/1.11, June 1996.
- [2] R. J. Hanisch, A. Farris, E. W. Greisen, W. D. Pence, B. M. Schlesinger, P. J. Teuben, R. W. Thompson, and A. Warnock. Definition of the Flexible Image Transport System (FITS). *Astronomy & Astrophysics*, 376:359–380, September 2001. For the latest version, see http://fits.gsfc.nasa.gov/fits_documentation.html.
- [3] ESO. *Data Interface Control Document*, GEN-SPE-ESO-19400-0794/4, April 2008.
- [4] ESO. *Physical Data Model Report: opc70*, September 2007. Document available at <http://arcdev.eso.org/dbdoc/opc70.doc>; will be incorporated into the new release of [6].
- [5] ESO. *Physical Data Model Report: User Portal*, October 2007. Document available at <http://arcdev.eso.org/dbdoc/UserPortal.doc>; will be incorporated into the new release of [6].
- [6] ESO. *Data Flow System, Database Design Document*, VLT-SPE-ESO-19000-1614/3, May 2003. Release of the new version of this document is expected in 2008.
- [7] ESO. *Physical Data Model Report: obrep2*, November 2007. Document available at <http://arcdev.eso.org/dbdoc/obrep2.doc>; will be incorporated into the new release of [6].
- [8] ESO. *qc1 database*, August 2007. <http://arcdev.eso.org/dbdoc/qc1.html>; will be incorporated into the new release of [6].
- [9] A. Dobrzycki, D. Brandt, D. Giot, J. Lockhart, J. Rodriguez, N. Rossat, and M. H. Vuong. Observation metadata handling system at the European Southern Observatory. In *Observatory Operations: Strategies, Processes, and Systems*. Edited by Silva, David R.; Doxsey, Rodger E. *Proceedings of the SPIE, Volume 6270*, pp. 627027 (2006), volume 6270 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, July 2006.
- [10] A. Dobrzycki, D. Brandt, D. Giot, J. Lockhart, J. Rodriguez, N. Rossat, and M. H. Vuong. Observations Metadata Database at the European Southern Observatory. In R. A. Shaw, F. Hill, and D. J. Bell, editors, *Astronomical Data Analysis Software and Systems XVI*, volume 376 of *Astronomical Society of the Pacific Conference Series*, pages 385–388, October 2007.
- [11] ESO. *ops_logs Database*, April 2008. Document available upon request; will be incorporated into the new release of [6].
- [12] ESO. *HOTFly – Headers on the Fly; Detailed Design*, March 2005. Document available upon request.

1.4 Acronyms/abbreviations

ADP: Advanced Data Products team of the Virtual Observatory Systems Department.

APEX: Atacama Pathfinder EXperiment telescope, a 12-m single dish antenna at Llano Chajnantor.

ASCII: American Standard Code for Information Interchange.

ASTOPP: database server on Paranal.



BSCW: Basic Support for Cooperative Work, a collaborative workspace software package for collaboration over the Web.

CfP: ESO Call for Proposals.

CSV: Comma-Separated Values, a file type for storing tabular data. In order to avoid confusion with CVS (see below), in this document this acronym is always used in lowercase. Note: the name is somewhat misleading, since the csv files can use any character (not just comma) as the field separator.

CVS: Concurrent Versions System.

DBCM: DataBase Content Management Team of the Data Flow and Operations Department.

DBO: DataBase Owner.

DDL: Data Definition Language, a computer language for defining data structures. In this document DDL specifically refers to SQL scripts used for create database objects (tables, views, triggers, etc.).

DDT: Director's Discretionary Time; special category of proposals submitted outside of regular Call for Proposals. Up to 5% of the available ESO general observing time may be used for DDT programmes.

DFI: Data Flow Infrastructure Department of the ESO Software Development Division.

DFO: Data Flow and Operations Department of the ESO Data Management and Operations Division.

DMO: ESO Data Management and Operations Division

EDAT: ESO Database Administration Team of the Operations Technical Support Department.

FITS: Flexible Image Transport System; standard data format widely used in the astronomical community. The FITS standard is defined in [2]. Detailed specifications for files generated at ESO are listed in [3].

GROND: Gamma-Ray Burst Optical/Near-Infrared Detector; an instrument owned by the Max Planck Institute for Extraterrestrial Physics (MPE), operated at the 2.2-m telescope on La Silla.

GTO: Guaranteed Time Observations; observations during time allocated to the external consortia who build ESO instruments.

IST: Instrument Specific Table.

LSP: La Silla Paranal Observatory

NGAS: ESO Next Generation Archive System.

OB: Observation Block; a sequence of commands to be executed by the telescope/instrument.

OLASLS: database server on La Silla.

OPC: ESO Observing Programme Committee.

OTS: Operations Technical Support Department of the ESO Data Management and Operations Division.

QC: Quality Control Team of the Data Flow and Operations Department.

RRM: Rapid Response Mode observations.



SAO: Science Archive Operations Team of of the Data Flow and Operations Department. Note: it should not be confused with the Smithsonian Astrophysical Observatory or the Soviet/Russian Special Astrophysical Observatory.

SCCS: Source Code Control System.

SDD: ESO Software Development Division.

SM: Service Mode.

SQL: Structured Query Language, a standard interactive and programming language for querying and modifying data and managing databases.

TOO: Target of Opportunity observations.

URL: Uniform Resource Locator.

USD: User Support Department of the ESO Data Management and Operations Division.

VISAS: ESO Visiting Astronomers Department.

VM: Visitor Mode.

VOS: Virtual Observatory Systems Department of the ESO Data Management and Operations Division

WFCAM: Wide Field Infrared Camera at the UK Infrared Telescope (UKIRT) on Mauna Kea, Hawai'i.

Chapter 2

Hardware; computer access

2.1 Machines used by DBCM and their roles

The following machines are used by DBCM. Unless noted otherwise, they run Solaris 2.8.

acproc: the principal operational/processing machine, used for servicing user archive requests. It runs Solaris 2.9. It is commonly referred to as the “production” machine.

acweb: the archive web server.

acrep: the host machine for the database replication server. Used by DBCM scripts to access database servers in Chile.

acd: the host machine for several datasets (including `/arc/headers`, see Sec. 3.4.5 on p. 25) as well as the access point to the SAFIQ server; used by DBCM to run related scripts.

acint: the “integration” machine; this machine is set up to mimic the environment on the acproc machine. This machine should be used for testing the interaction of newly developed or newly updated tools with the operational system, prior to their upload to the operational machine.

acwebint: the integration machine for the web tools; it is set up to mimic the environment on the acweb machine. All web-related tools should be integration-tested on this machine prior to being uploaded to acweb.

dbcmdev1: the DBCM “own” development machine. Tool development and unit testing should be performed on this machine (with the exception of Solaris-specific software). It runs Linux 2.6.9.

dmdarc1: the development and unit testing machine for Solaris-specific tools.

acbu3: machine used for data transfer between Chile and Garching.

See Sec. 7.1.3 on p. 35 for the description of the procedure for tool development, transfer between machines and installation.



2.2 Database servers

The principal operational database server in Garching is a Sybase Adaptive Server Enterprise (ASE) v. 12.5.3. It is called **ESOECE**. The development and integration ASE servers are, respectively, **DEVSRV** and **GARINT**.

The principal data warehouse server in Garching is a Sybase IQ v. 12.7. It is called **SAFIQ**. The test/development IQ server is called **warehouse** (subject to change soon!).

The database servers on Paranal and La Silla are called **ASTOPP** and **OLASLS**, respectively. Both are Sybase ASE v. 12.5.3.

2.3 User accounts

Note: passwords for shared operational login and database accounts should be obtained from the DBCM team leader. The passwords should not be disclosed to other users, even if those users have a legitimate interest in using those accounts; such users should be directed to the DBCM team leader.

2.3.1 Login accounts

For development work on the `dbcmdev1` and `dmdarc1` machines, DBCM personnel own user accounts should be used. Care need to be taken that personal settings of the accounts (such as file access permissions set by the user, e.g. with the `uname` command) are not propagated to files transferred to operational machines.

The following accounts are used to login into operational machines:

- `archeso` – to access `acproc` and `acint` for database related activities,
- `dha2dbcm` – to access `acbu3` machine (daily transfer of operational logs),
- `dbcm` – to access the `safiq` machine, to maintain the IQ databases,
- `web` – to access `acwebint` and `acweb`.

2.3.2 Database accounts

The following database accounts are used:

- `<database>_dbo` – to access `<database>` as DBO,
- `archeso` – to maintain the back-end databases; this account can also be used as a generic access for all databases/objects which allow public access,



- `dbcm` – to access the `keywords_repository` database in the IQ server,
- `esolib` – to maintain the library database,
- `ops_log` – to maintain the `ops_logs` database in the IQ server.

Other database accounts, which are not directly used by DBCM, but which are otherwise notable, are:

- `operator` – used by SAO to perform updates (e.g. release dates) in the observations database,
- `starcats` – traditional “general use” account for direct querying of databases; this user should not have “update” or “delete” permissions on any database object, and “select” permissions should be assigned to it only on objects which do not contain confidential data,
- `www` – account used by queries generated by the archive/database web interfaces.

Chapter 3

ASE databases

3.1 The `opc70` database

`opc70` is the name of the database filled in Garching during Phase I, i.e. proposal submission. This database contains all information related to the submitted programmes.

Detailed description of the database is contained in [4]. Only most important and/or most commonly accessed information is mentioned below.

The principal tables in the database are: `sched_rep`, `obs_runs`, `obs_programmes` and `latex_source`.

The most important information stored in those tables is:

Proposal period	e.g. 80
Mode	Service or Visiting
Programme ID	e.g. 080.A-1234 (see below)
Run ID	e.g. 080.A-1234(B) (see below)
Allocated time	e.g. 1800 sec.
Telescope	e.g. UT3
Instrument	e.g. VIMOS
PI/CoI	
PI's email address	
Title of proposal	
Abstract of proposal	

ESO issues Call for Proposals (CfP) every six months, with the deadlines around 1 April and 1 October. Each CfP is associated with an “observing period.” There are two observing periods per calendar year, and they run from 1 April to 1 October and from 1 October to 1 April. A CfP is always issued for the period beginning half a year after the deadline.

All proposals obtained as a result of the CfP are reviewed by the ESO Observing Programme Committee (OPC). The OPC selects the successful candidates (Principal Investigators and Co-Investigators, commonly referred to as “PIs” and “Co-Is”) and allocates observing time to the programmes.



The proposals above the limit of resources (there is a finite amount of time available per instrument and telescope) have `opc70..sched_rep.published` set to 1.

The structure of the programme ID designation is as follows:

TPP.S-nnnn

The leading digit 'T' indicates the programme type:

- 0 – normal programme, intended to be completed within one six-month period.
- 1 – large programme (LP), intended to last longer than one period.
- 2 – Director's Discretionary Time (DDT) programme, allocated outside of the regular Call for Proposals. It is typically used to observe objects/phenomena of high interest which were not known during the regular CfP.
- 3 – short programme. A special type of normal programme, where total observation time is under 10 hours.
- 4 – calibration programme. Programmes aiming at improving calibration of the ESO instruments.

The two digits 'PP' show the observing period during which the observations are intended to be performed.

The capital letter 'S' following the dot describes the proposal category.

- A cosmology
- B galaxies and galactic nuclei
- C interstellar medium, star formation and planetary systems
- D stellar evolution

Four digits 'nnnn' following the dash show proposal running number within the period. It is padded with zeros to always take four digits.

So, for example, observing programme 080.A-0123 indicates normal programme from observing period 80, which deals with cosmology, and which was the 123th proposal submitted during the Call for Proposals for Period 80.

Every observing programme has one or more *runs*, which are denoted with a capital letter inside parentheses at the end of the programme ID, i.e. the run ID is:

TPP.S-nnnn(R).

So, run ID of 080.A-0123(B) denotes the second run of the programme 080.A-0123 from the above example.

Important: please note that programme ID and run ID are frequently confused. For example, the `prog_id` column in `observations..data_products` is in fact the run ID.

Note that the following types of programmes are also often mentioned:



Guaranteed Time Observations (GTO) – time allocated to the consortium which build the instrument.

Rapid Response Mode (RRM) – usually used for observation of highly variable objects, when quick response to some phenomenon is essential. One such case are the Gamma-Ray Bursts (GRBs), which fade very quickly, and thus need to be observed as soon as possible after the discovery.

Target of Opportunity (ToO) – used for observations of unpredictable objects, e.g. novae and supernovae.

but they are not in principle associated with any specific programme type.

3.1.1 Inserting data

The tables in the `opc70` database are filled using automated procedures by users submitting proposals and by the VISAS Department using tools under their control.

3.1.2 Maintenance

There are no scheduled maintenance tasks for `opc70`.

In case of Garching–Chile replication problems, contents of this databases may have to be synchronised. The procedure is described in Sec. 9.1.1 on p. 48.

A common non-scheduled task is caused by the fact that an individual user may have more than one user ID in the system. The task is known as *merging users*; the procedure is described in Sec. 9.9 on p. 57.

Other content management (updates, changes, deletions) is only done per specific requests, using ad-hoc SQL queries. All requests for modification need to be evaluated individually, and need to be approved by VISAS; this approval may be considered to be implied if the request for database content modification comes from this group.

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

3.2 The `user_portal` database

The `user_portal` database is used to store information necessary for ESO users to access ESO services: User Portal account information, contact information, etc. Parts of this database are replicated to the Chilean sites. DBCM is the `user_portal` database owner (DBO).

The database design is described in [5].



3.2.1 Inserting data

The data are inserted into the database by the users, using ESO User Portal web interface.

3.2.2 Maintenance

There are no scheduled maintenance tasks for `user_portal`.

In case of Garching–Chile replication problems, contents of this databases may have to be synchronised. The procedure is described in Sec. 9.1.1 on p. 48.

Merging users (Sec. s:usermerging on p. 57) affects the `user_portal` database.

Other content management (updates, changes, deletions) is only done per specific requests, using ad-hoc SQL queries. All requests for modification need to be evaluated individually, and need to be approved by USD; this approval may be considered to be implied if the request for database content modification comes from this group.

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

3.3 The obrep2 and obrep2_hist databases

The `obrep2` database contains information on Observation Blocks (OBs), i.e. command sequences used to execute observations. The contents of this database are replicated to the Chilean observatories.

The Service Mode OBs are created in Garching database during Phase II of the ESO Dataflow. USD is in charge of the contents of this database, but DBCM is the `obrep2` database owner (DBO).

To save space and increase access efficiency, old Service Mode OBs are periodically moved to the `obrep2_hist` database in Garching, which has identical structure as `obrep2`. The criteria for classifying an OB as “old” are provided by USD.

The Visitor Mode and technical OBs are generated on the mountains. They are replicated directly to the `obrep2_hist` database in Garching and periodically cleaned using criteria defined by the LSP Science Operations team.

The design of the `obrep2` databases is described in [6, 7].

3.3.1 Inserting data

The tables in the `obrep2` database are filled by users generating OBs with the use of the `p2pp` tool.



3.3.2 Maintenance

There are no scheduled maintenance tasks for obrep2. In case of Garching–Chile replication problems, contents of this databases may have to be synchronised. The procedure is described in Sec. 9.1.1 on p. 48.

Merging users (Sec. s:usermerging on p. 57) affects the obrep2 database.

Other content management tasks (updates, changes, deletions) are done per specific requests, using ad-hoc SQL queries. The frequent requesters include VISAS and USD. All requests for obrep2 modification need to be evaluated individually and need to be approved by USD.

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

3.4 The observations database

This is the principal back-end database, containing information on metadata of all raw frames observed in the Chilean observatories, some additional astronomical data (WFCAM), as well as some test data generated off-line in Garching.

Detailed description of the database is contained in [6]. Only most important and/or most commonly accessed information is mentioned below.

The main tables in the observations database are `data_products`, the so-called Instrument-Specific Tables (ISTs): `uves`, `fors1`, `fors2`, `isaac_all`, etc. The other important tables are `dp_headers`, `isaac_stacks`, `dp_urls`, `packages` and `pkg_cont`.

The principal, unique identifier of all raw frames is `dp_id`. The structure of this identifier is:

```
<instrument>.<observation time>
```

where `<instrument>` is the common name of the instrument, as recorded in the `olas_ins` table (see Sec. 3.4.6 on p. 25) and `<observation time>` is the time tag, in the `CCYY-MM-DDThh:mm:ss.sss` format, accurate to 1 ms, of the beginning of the exposure (as stored in the `DATE-OBS` keyword in the frame). For example:

```
UVES.2007-04-01T12:34:56.789
```

3.4.1 The `data_products` table

This is the principal database table in the observations database. It contains the most important, generic observation information such as: target identification, celestial position, instrument, type of observation (e.g. science or calibration), observation mode (e.g. imaging or spectroscopy), exposure time, ESO programme identification, etc.

This table also contains information on the frame public release date – to protect the data during proprietary periods – and any access restrictions that may apply to the file.



Inserting data

This table is filled in three ways:

1. Following data acquisition on La Silla and Paranal, the FrameIngest tool extracts relevant information from the raw frame headers and inserts it into copies of the table on the respective mountain database servers. All insert operations are replicated to Garching.
2. WFCAM data are transferred from England using archiving procedure. As part of the procedure, FrameIngest installed locally (i.e. in Garching) extracts information from the FITS headers and puts it in `data_products`.
3. APEX files are delivered from Chajnantor on USB disks in semi-regular intervals. The contents of those disks are transferred to NGAS by the SAO team using special procedure, which includes running a Garching copy of FrameIngest.

Controlling access to data

Visibility of a frame to outside users and availability of frame for download is managed by two entries in the `data_products` table.

The `public_f` flag is used by archive queries to establish whether a file is to be made visible to the query as well as to put restrictions on access to the file.

For security reason, all data are by default hidden (`public_f=NULL`) and the `data_products` insert trigger in the Garching copy of the database updates the access information following rules specified in the `olas_ins` table.

This flag can have the following values:

NULL – the file is “hidden” in the database. In this case, archive queries will not display the file.

'R' – access to the data is restricted to users from ESO member states,

'W' – explicitly allows worldwide access to the file, i.e. the file can be downloaded by any registered archive user

'T' – default access.

Per current ESO policy, the default access is equivalent to worldwide access.

The `rel_date` column lists the date after which the relevant file can be requested and downloaded by external users. Under normal circumstances, the values of `rel_date` is equal to the exposure start plus one year, if the file belongs to the SCIENCE or ACQUISITION category, or just to the exposure start for all other categories, such as calibration, test, and technical data.

For SCIENCE/ACQUISITION frames obtained in Service Mode, `rel_date` is later modified to show the release date of the data to the PI plus one year.

Special access rules apply to HARPS, WFCAM, and APEX data. For details, see Chapter 8 on p. 45.



The data_products insert trigger

The tri_data_products_ins trigger was created on ESOECF data_products for insert operations.

Whenever a new observation is performed, a new file (identified by unique dp_id) is generated. The file's metadata are then inserted into ASTOPP/OLASLS observations..data_products table. This information is then replicated to Garching and inserted into observations..data_products on ESOECF. The trigger is executed at this stage.

IMPORTANT: this trigger is *not* used on the Chilean sites.

If the new entry is a science/acquisition frame (as indicated by the dp_cat column), then the release date (rel_date) is set to the observation time (exp_start) plus one year. For other files, rel_date is set to exp_start.

Then, the trigger will make files visible or not to external queries (by setting public_f flag) according to dp_cat and the value of the trigger_flag in the olas_ins table.

The trigger also checks if the file has been produced by a previously unknown instrument. If this is the case, a new entry is inserted into the olas_ins table (see Sec. 3.4.6 on p. 25).

Maintenance

Maintenance of the data_products table is most commonly done by manual modification of entries following requests from various groups.

All requests need to be evaluated on a case-by-case basis.

The most common manipulations on this table are:

HIDE Requests to hide files usually come from the Garching QC and SAO teams.

The correct action is to set public_f to NULL and note the requester and date in dp_class, e.g.:

```
update data_products
set public_f=NULL,dp_class='darthvader,20080401'
where ...
```

UPDATE Request to update files come from various sources: QC, Paranal/La Silla, USD, SAO. Own investigations may also lead to updates.

The correct action is to use the updateTool (Sec. 7.4 on p. 36), which – in addition to entering the update in data_products and ISTs – will also introduce correct update in the keywords_repository table in SAFIQ.

RELEASE DATE Requests for change of the release date need to be considered on a case-by-case basis. Changes can be implemented only after explicit approval of the DBCM team leader, DFO Department Head or DMO Division Head, or at the specific request of VISAS Head or the Director General.

The correct action is to modify rel_date to the approved date and note the requestor in dp_class, e.g.:



```
update data_products
set rel_date='2007-12-31',dp_class='darthvader,20080401'
where ...
```

DATA ACCESS It is conceivable that DBCM receives a request for modifying access rights to certain data sets (e.g. to ESO Member States only).

Such changes require explicit approval of the DBCM team leader, DFO Department Head or DMO Division Head, or VISAS Head or the Director General.

The correct action is to set `public_f` to the appropriate flag and note the requestor and date in `dp_class`:

```
update data_products
set public_f=R,dp_class='darthvader,20080401'
where ...
```

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

3.4.2 Instrument Specific Tables (ISTs)

NOTE: this section describes the "old" way of generating and filling the IST: creation of a table in both Garching and Paranal, and filling it with FrameIngest on the mountain and replicating to Garching. It is anticipated that new ISTs will be created based on information from keyword_repository. At the time of the release of the present document, this new method is still under development.

Whenever a new instrument is put online, data specific to this instrument (filters, grisms, etc.) should be stored in an Instrument Specific Table (commonly called IST). The design of ISTs has to be done in collaboration with the relevant QC scientist.

With the exception of the APEX IST (see below) the tables are filled directly in Paranal by the FrameIngest tool. The contents of the ISTs are then replicated from Paranal to Garching. Thus, one has to create the ISTs both in Paranal (ASTOPP server) and Garching (ESOECF server) databases. The replication is set up by the EDAT team.

Creating new ISTs

The following steps have to be followed when a new IST is created:

- The QC scientist has to provide a list of relevant keywords specific to the instrument.
- An SQL script for creation of the table needs to be developed. ESO keyword dictionary database (<http://archive.eso.org/Tools/DidRep/DidRepWebQuery>) needs to be consulted for definition of the format of entries.
IMPORTANT: the table must allow insert permission for user `frameingest`, update permission for `archeso` and `frameingest`, and select permission for users: `archeso`, `frameingest`, `operator` and `www`.
- The script must be tested on DEVSRV.
- The script must be provided to SDD/SEG for testing.



- SDD will take care of updating FrameIngest configuration in Garching and on Paranal and will coordinate generation of the tables in both sites.
- Installation of a new IST requires additions and/or modifications in the `obs_tables` and `obs_columns` tables in the observations database. On Paranal, those tables are modified by SDD. In Garching, the tables must be modified by DBCM. The necessary step is to edit:

```
/saf/arcs/w/ecf/config/develop/db/archive/sql/gar_frameingest_config.sql
```

adding information about the new IST, and then to execute the script.

- Replication needs to be then set up by the EDAT team.
- Once an IST is implemented and put on-line in the observations database, it needs to be back-filled with historical data.

Special case: APEX IST

APEX data are ingested in Garching. Therefore, the APEX IST exists only in the Garching database.

Because one APEX file can contain data from more than one “instrument” (the “FEBEs”, see Sec. 8.1 on p. 45), a special way of inserting the data is used (in order to be able to utilise FrameIngest, which allows for only one entry in the IST per file):

- FrameIngest tool inserts the data into `apex_ingest` table. The table contains room for twenty FEBEs per file (which, according to APEX Project Scientist, is more than enough).
- An insert trigger `apex_ingest_to_apex_storage` splits information for every FEBE and puts it into `apex_storage` table. The trigger also assigns an integer index to each APEX `dp_id` and stores it in the `apex_file_index` table.
- Information from `apex_storage` and `apex_file_index` is combined in a view called `apex`, which serves as the final APEX IST.

NOTE: unlike in other ISTs, `dp_id` in the `apex` IST is *not* unique.

The contents of the `apex_ingest` table (but not the table itself) can be deleted at any time when there is no APEX loading activity.

Inserting data

With the exception of the APEX IST (see above), the ISTs are filled in a way which is virtually identical to the way it is done for the `data_products` table.



IST insert triggers

The insert triggers on some of the ISTs (*amber*, *crires*, *sinfoni*) are used to convert values of certain FITS keyword to formats compatible with SQL — e.g. to convert into decimal degrees the pseudo-float entries for celestial coordinates, where, for example, declination of 12 deg, 34 arcmin, 56.789 arcsec is stored in FITS header as a floating point value of 123456.789.

In case of *vimos*, the trigger recognises the VIMOS quadrant used and then converts the quadrant-specific keywords into generic database entries, e.g. it copies the value of the `INS.FILT3.NAME` keyword (which has quadrant no. 3 hardwired in the keyword name) into `ins_filt_name` (which has no quadrant reference in the name). This enables queries for all VIMOS files using some specific filter independently of the quadrant. Without it, one would have to use a complicated query using “or” statements for all possible quadrants.

Maintenance

The regular maintenance of the contents of the ISTs should be done only with the `updateTool` (see Sec. 7.4 on p. 36).

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

3.4.3 The `isaac_stacks` table

A single observation with ISAAC produces a group of files. The `isaac_stacks` table provides reference to groupings (“stacks”) of individual files, allowing retrieval of all files which are logically combined.

This table is filled by a cronjob (see Sec. 7.5.13 on p. 42).

3.4.4 The `dp_headers` table

This table is used to facilitate fast transfer of raw frame headers from Chile to Garching. The file headers are extracted and stored in the database as compressed binary data. The table is filled by `FrameIngest` on the remote sites and replicated to Garching.

The table is used by the cron `LaunchEX_LOAD` script (see Sec. 7.5.14 on p. 43) as input for loading header information into the `keywords_repository` table.

NOTE: the `dp_headers` in the observations database is in fact a view of the same table in a separate database, `headers`. The reason is the fact that this table grows rather quickly, and is used only intermittently, and if it were in the observations database, it would slow down recovery from a potential database crash.

No updates of any kind are done on this table.

To save space, the entries which are older than 3-6 months should be periodically deleted.



3.4.5 Header display

This paragraph describes the present setup. It is planned to move to header display based on the contents of the `keywords_repository` table.

The information necessary for header display is stored in two places:

1. The `/arc/headers/` directory contains ASCII files with header dumps – which are the source files for header display. The files for any particular night are stored in a subdirectory corresponding to that night (e.g. files from the night of 1-2 April 2007 are stored in `/arc/headers/2007-04-01/`. The files are called `<dp_id>.hdr`, e.g. `ISAAC.2007-04-01T22:33:44.555.hdr`.

IMPORTANT: the header display tool can use gzipped input header dumps. Since the `.hdr` files consist only of ASCII characters and contain numerous blank spaces, they compress extremely well, which results in massive space savings. Therefore the contents of the `/arc/headers/` subdirectories should be periodically gzipped.

2. The `dp_urls` table contains the relation between the location of the `.hdr` file and the URL information shown in the header display link in the archive query result.

The `dp_urls` table also contains information necessary for allowing data previews.

Inserting data

Both the table and the `/arc/headers/` directory are filled by a cronjob (see Sec. 7.5.12 on p. 42).

Adding preview information in `dp_urls` is done manually, per requests from the VOS Department.

Maintenance

There is no scheduled maintenance of the `dp_urls` table.

The maintenance of the `/arc/headers/` directory consists of periodical gzip-ping of the contents of the subdirectory. There is no specific rule; a rule-of-thumb would be that data older than 6 months can be compressed, but younger data can safely be compressed if, for example, SOS informs DBCM about disk space issues.

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

3.4.6 The `olas_ins` table

The information stored in this table is the basis for the visibility rules for all ESO raw frames.

This table contains general information about instruments and `olas_id` values. It also contains four important flags: `valid_flag`, `trigger_flag`, `in_ist_flag` and `hotf_flag`, which are used as input values by several scripts/triggers.



The `olas_id` entry is the instrument identifier. This is the prefix for all archived ESO raw files.

The `ins_name` entry is the common instrument name. Please note that for historical reasons the same instrument may have several values of `olas_id`. At present, care is taken to use the same `olas_id` for the entire life of the instrument.

The `tel_seq` entry is a coded order of arrival of the instrument in the observatory, for example:

`NULL` – test instrument, not mounted or dismounted, no longer in use, etc.

`'PL-VLT,6'` – sixth visible/UV instrument on VLT in Paranal.

`'PL-VLTI,I1'` – first infrared instrument on VLTI in Paranal.

`'LS,I2'` – second infrared instrument in La Silla, etc.

When new `olas_id` is entered into `olas_ins`, it has `ins_name` and `tel_seq` automatically set to `'?'`. Actual values need to be updated manually.

By default, all `xxx_flag` entries are set to `'N'`.

Inserting data

New row(s) can be inserted into `olas_ins` by the `data_products` insert trigger (see p. 21).

Maintenance

Maintenance of this table is done manually, using ad-hoc SQL queries. The principal tasks are updating information on new instruments (see Sec. 9.2 on p. 50 and Sec. 9.3 on p. 51).

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

3.5 The qc1 database

This database contains several tables showing Quality Control parameters, as derived by the Garching QC Team.

The database is described in [8].

3.5.1 Inserting data

The tables are filled by the QC Team, using dedicated tools.



3.5.2 Maintenance

There is no scheduled maintenance of this database. Occasionally, DBCM performs DBO tasks following requests from the QC group (see Sec. 9.8 on p. 55).

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

3.6 The library database

This database contains information on scientific papers based on data obtained with the use of ESO telescopes or ESO Archive.

3.6.1 Inserting data

The tables are filled by a cronjob (see Sec. 7.5.5 on p. 40).

3.6.2 Maintenance

There is no scheduled maintenance of this database. Occasionally, DBCM performs DBO tasks following requests from the Library staff; this is done manually, using ad-hoc SQL queries.

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

3.7 The ambient database

This database contains information about ambient conditions at Chilean sites. The database design is described in [6].

3.7.1 Inserting data

The database is filled via replication from the Chilean sites.

3.7.2 Maintenance

There is no scheduled maintenance of this database.



3.8 The asto database

This database contains only one table, `mdfiles`, which stores site-specific information for files ingested into the observations database. The database exists both in the Chilean sites and in Garching. However, since the information is site-specific, this table is *not* replicated.

The Chilean sites actively maintain the contents of this database.

The Garching version of this table must exist because it is required by `FrameIngest`. However, there is no use for the contents of the local version of the table, and – to save space – it should be kept empty.

3.8.1 Inserting data

The database is filled by the `FrameIngest` tool. In Garching the inserted entries are immediately deleted.

3.8.2 Maintenance

The Garching version of `asto.mdfiles` should remain empty. This is achieved with the use of the insert trigger, `tri_mdfiles_insert`, which deletes new entries as soon as they are inserted.

There is no scheduled maintenance of this database.

Chapter 4

SAFIQ databases

4.1 Keyword repository

The following points summarise the basic ideas behind the keyword repository tables in SAFIQ:

- A table named `keywords_repository` contains keyword names, values, formats, etc. for all FITS headers stored in ESO Archive. That includes all raw files from La Silla, Paranal and Chajnantor, as well as all headers from WFCAM. It also contains all headers of science and calibration files generated by QC-Garching. In the future, it will also contain headers of advanced data products generated in house by the VOS department or provided by users.
- The `keywords_repository` table contains the up-to-date information.
- The historical information (i.e. updates, additions, modifications, deletes) is stored in a separate table, `keywords_history`. The structure of this table is similar to the `keywords_repository` table, but it also contains three additional columns – `update_timestamp`, `update_requestor` and `update_comment` – which record information about the updates.

The database is described in [9, 10].

4.1.1 Inserting data

The header information from the raw data from Paranal and La Silla is inserted into `keywords_repository` by a cron job, which reads the data from the `observations.dp_headers` table (see Sec. 7.5.14 on p. 43).

The header information from the processed data is inserted into `keywords_repository` by a cron job, which obtains headers directly from NGAS as soon as the corresponding file is ingested (see Sec. 7.5.15 on p. 43).



4.1.2 Maintenance

The regular maintenance of `keywords_repository` should be done only with `updateTool` (see Sec. 7.4 on p. 36).

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

4.2 Operation logs

The Paranal operation logs (sensor readouts, instrument/detector settings, etc.) are stored in a large (billions of rows) table in SAFIQ, called `ops_logs`. There are two auxiliary tables, `log_sources` and `log_files`.

The database is described in [11].

4.2.1 Inserting data

This table is filled by a cronjob (see Sec. 7.5.19 on p. 44).

4.2.2 Maintenance

Since the principal idea behind this database is to record the history of Paranal operations, there is no scheduled maintenance. Principal DBCM activity related is to monitor database completeness, by verifying that daily loads worked, troubleshooting in case of problems, and filling the historical gaps.

Any maintenance of this database needs to be done using ad-hoc SQL queries.

All maintenance needs to be recorded in the DBCM Portal (Chapter 5 on p. 31).

Chapter 5

DBCM Portal task tracking system

The Action Remedy Reporting System at ESO (<http://support.eso.org/>) provides support for the “DBCM Portal.” It is intended to serve as the action/request tracking system in the transitional period before a DMO-wide system is implemented.

Action requests (commonly referred to as “tickets”) can be initiated in two ways:

1. By sending email to dbcm-help@eso.org, with subject line beginning with DBCMREQUEST. The latter is used as an email spam filter (messages without that word are rejected by the email server). In this case the subject of the message is logged as the **Title/Summary** of the request, and the body of the message becomes the **Description**. An email to dbcm@eso.org is generated automatically; this email contains URL to the ticket.
2. If the request was obtained in any other way (plain text email, phone, direct communication, etc.), the ticket needs to be registered by logging into the Remedy system, either directly to the DBCM Portal:

<http://support.eso.org/arsys63/forms/remgar.hq.eso.org/DBCM/>

or through the main page, selecting DBCM from the options.

Note: at every available opportunity, non-DBCM users should be informed/reminded about the possibility of generating tickets automatically via email; this will hopefully lead to reducing the number of tickets generated manually.

All DBCM tickets should be promptly assessed.

As a rule, the status of the ticket should be first changed to “WIP” (Work-In-Progress) before the requested action is initiated, followed by change of status to “Done” (or “Rejected”) after the completion of the task. However, for tickets which are expected to be completed immediately, it is acceptable to skip the work-in-progress state.

Every ticket which is put in “Done” or “Rejected” state must contain the description of the action taken in the **Worklog** area.



It is recommended, but not required, to set the **Category**, **Responsible** and **Priority** entries to appropriate values.

For tickets set to the “Work-In-Progress” state, it is recommended that the **Anticipated completion time** entry be set to the best estimate for the amount of time needed for the completion of the task.

For historical reasons, the tickets in the DBCM Portal contain a **Level** attribute. This attribute is not used and can be left at the default value.

Completion of a task – i.e. changing its status to “Done” or “Rejected” – is automatically communicated via email to the submitter and to dbcm@eso.org.

Chapter 6

DBCM on the World Wide Web

DBCM does not maintain any operational web pages or interfaces. The presence of DBCM on the web is limited to:

DBCM web site: a basic team presentation page.

The source files reside in the `/home/http/archive/docs/DBCM` directory. Any updates to this page need to be performed on the `dmdarc1` machine, and transferred to `acwebint` and `acweb` using the standard procedure described in Sec. 7.1.3 on p. 35,

Important: unless it is specifically intended otherwise, the HTML source file should only contain relative links to pages in the `http://archive.eso.org` domain and its subdomains (e.g. a link intended to point to `http://archive.eso.org/somepage/` should be entered as `...`); in that way, the development and operational versions of the page will point to the appropriate versions of the linked sites.

The web site for the DBCM web page is:

`http://archive.eso.org/DBCM/`

with `archive` replaced with `arcdev` and `acwebint` for the development and integration versions, respectively.

DBCM Twiki page: this page is intended for recording progress of DBCM projects and storage of documentation. Its URL is:

`http://arcdev.hq.eso.org/atw/bin/view/Main/DBCMGroup`

DBCM BSCW Workspace: contains various DBCM-related documents. The URL is:

`https://groupware.eso.org/bscw/bscw.cgi/0/279402`

However, this workspace is no longer actively used; all new documentation should be put into the Twiki page (see above).

Chapter 7

DBCM software

7.1 Software development

In principle, software development at DBCM should be limited to scripts/tools used for database monitoring and manipulations. All operationally-critical software must be developed and tested by the SDD/DFI Department.

For historical/manpower reasons, DBCM continues to maintain certain operational tools (e.g. H0TFly, file manipulation tools used by the Request Handler). It is anticipated that this role will diminish in the foreseeable future, and will eventually disappear.

7.1.1 Software location

The software used by DBCM can be found in the following directories and subdirectories:

1. The software used for scheduled maintenance and for operational tasks resides in the subdirectories of the /opsw tree:
 - /opsw/rh and subdirectories: scripts used by the Request Handler.
 - /opsw/dbcm and subdirectories (including the cron subdirectory): scripts used for regular and semi-regular database monitoring/maintenance tasks.

2. The scripts used for ad-hoc database manipulations reside in

`/home/dbcm/`

and subdirectories.

It is anticipated that all scripts from this area will be eventually put under proper version control and moved into the /opsw tree.

3. The DDL SQL scripts reside in:

`dmdarc1:/saf/arcsw/ecf/config/develop/db/archive/sql/`



7.1.2 Version control

All scripts in the /opsw directory tree should be placed under version control. Currently, both CVS repository and SCCS are used; the goal is to migrate to CVS.

Whenever applicable, the name of the CVS module containing the tool is mentioned.

7.1.3 The development → integration → production S/W cycle

Note: the expression commonly used to describe this procedure is *to put a tool in production*.

The following procedure for development and installation of tools should be utilised:

- Unless there are specific reasons for avoiding dbcmdev1 (examples of those reasons are necessity of compiling tools under Solaris or unavailability of libraries/modules), development and unit testing of tools should be done on this machine.

Any maintenance of any tool should include an attempt to port the tool to dbcmdev1.

- Development and unit testing of tools that need to be compiled in the Solaris environment (in particular, all Request Handler related software falls into that category) should be done on dmdarc1.
- Prior to transferring the tool to the integration machine, make sure that the tool has correct permissions, i.e. it is group- and world-readable and executable (i.e. its permission flags are -rwxr-xr-x).
- After successful unit testing, the tool should be saved with a version control software. CVS should be used if possible.
- For integration testing, the tool is to be transferred to the respective integration machine, i.e. acwebint for web-related tools and acint for all other tools. This is done using rsync:

```
rsync -av --rsync-path=/opsw/util/bin/rsync \\
      <full path>/tool.py                \\
      <user>@<machine>:<destination path>
```

For example:

```
rsync -av --rsync-path=/opsw/util/bin/rsync \\
      /opsw/dbcm/harps/scramblegto/lgetharpsgtoids.sh \\
      archeso@acint:/acdist/opsw/dbcm/harps/scramblegto/
```

NOTE! When transferring tools to the /opsw directory tree, it is necessary to use /acdist/opsw on the destination machine, even though the /acdist part can be omitted when directly accessing the tool while on the destination machine.

NOTE! Do not forget to explicitly mention user@machine combination in the destination.

- After integration testing, the tool is transferred to the respective operational machine using rsync as in the previous item. For example:

```
ssh archeso@acint
rsync -av --rsync-path=/opsw/util/bin/rsync \\
      /opsw/dbcm/harps/scramblegto/lgetharpsgtoids.sh \\
      archeso@acproc:/acdist/opsw/dbcm/harps/scramblegto/
```



7.2 Headers-On-The-Fly (HOTFly)

Archive retrieval requests provide for the possibility of applying on-the-fly file header corrections. This is performed by the HOTFly utility. This programme compares the contents of selected columns in the `observations..data_products` database table with the contents of respective keywords in the FITS file header and – if necessary – updates the header.

The tool is also used for correcting headers prior to header display.

Description of the tool can be found in [12] and documents referenced therein.

HOTFly development is frozen. Maintenance is limited to fixes of critical bugs.

It is anticipated that in the near future the functionality of this tool will be taken over by a new tool developed by SDD/DFI.

7.3 File manipulation tools.

For historical reasons, DBCM continues to maintain several Python scripts related to handling of archive queries and requests:

- `/opsw/rh/bin/start_hotf.py`: launching HOTFly.
- `/opsw/rh/bin/start_gunzip.py`: uncompressing files, called by `start_hotf.py` before applying header corrections.
- `/opsw/rh/bin/start_gzip.py`: compressing files after HOTFly run and before delivery to the user, called by `start_hotf.py`.
- `/opsw/util/python2.1.3/lib/python2.1/site-packages/imprv/src/ImprvShowHdr.py`: script for applying keyword corrections to the ASCII file prior to header display.

The unusual location of the file is caused by Python version-specific issues.

Maintenance of the above scripts is limited to fixes of critical bugs.

7.4 Database update tool

The update tool is used to update the contents of `keywords_repository` on SAFIQ as well as `data_products` and `ISTs` on ESOECF. If any updates are done in `data_products`, the relevant keywords are also updated in the repository. The tool can also be used to update the repository alone for keywords not present in `data_products`.

All updates in the repository first create a copy of the entry in `keywords_history`, and then the entry is updated.



The update tool should be run on the acd machine. It resides in `/home/archeso/dbcm/Tools/` and is named `updateTool`.

The script is run from the command line. The basic use, once in the folder:

```
./updateTool
```

The following options can be used:

- To specify entries, there are 3 different ways:

- A single file with the `-f` option, for example:

```
./updateTool -f UVES.2007-06-27T15:24:24.241
```

- A where clause (without the `where` keyword) using the `-w` option, eg.:

```
./updateTool -w "prog_id='079.D-5032(A)'"
```

(Do not specify the keyword's name for the repository here!). NOTE: because of limitations on the use of spaces, the "where" clause should always be put at the end of the command line.

- A File with a list of `dp_id`'s, using the `-F` option:

```
./updateTool -F dp_ids_for_update.txt
```

The `-F` option expects one `dp_id` per line, and the `dp_id` to be the first item of the line. If this last point is not fulfilled, then one can use:

- * `-s` to specify a separator for the file (default: none).
- * `-n` to specify the field number for `dp_id`'s, starting at 0 (default: 0).

For example, in a comma-separated table, with `dp_id`'s in the second column, the command would be:

```
./updateTool -F dp_id_for_update.txt -s, -n 1
```

Also, one of the 3 options must be specified and only one will be used.

- Specifying the update

The first part of the update is to tell the `updateTool` which column to specify. There are two options for this:

- Specifying a column in `data_products` with the `-c` option. The tool will automatically find the associated column in the repository for the update, if necessary. For example:

```
./updateTool [dp_ids] -c prog_id
```

- Specifying a specific keyword in the repository with the `-k` option. This should *not* be used if the keywords is also present in `data_products` since it would not update it. For example:

```
./updateTool [dp_ids] -k "HIERARCH ESO RANDOM KEYWORD"
```

At least one of the above two options must be filled.

The following options are mandatory:

- The new value for the update, with the `-v` option.



- The update rquestor, with the `-r` option.

For example:

```
./updateTool -c prog_id -v '081.C-1234(A)' -r lskywalker \<\  
-w "ob_id in (200166725,200166724,200166723)"
```

7.5 Scripts used for scheduled maintenance

This section lists all database maintenance scripts run automatically by the cron daemon on operational machines.

To see the actual times/dates when any given script is run, login to the appropriate machine as the appropriate user (e.g. as user `archeso` to machine `acproc`), and type:

```
crontab -l
```

See `cron` and `crontab` man pages for information on how to interpret the output.

To edit cronjobs, use:

```
crontab -e
```

which will open the `crontab` file using the default editor.

7.5.1 Non-standard access rights

The script:

```
acproc:/opsw/dbcm/cron/Prog_id/update_prog.py
```

modifies `public_f` and `rel_date` for programmes for which access rights are different than the standard one-year proprietary period and worldwide access.

This script does not deal with HARPS GTO and WFCAM data; they are taken care of by specific scripts (see below)

The script is run daily.

The input is a hand-edited list of programmes with their proprietary periods and access flag. It is planned for near future to replace this file with an automated mechanisms based on the information contained in the `opc70` database.

Note: this mechanism is used to hide the “Brazilian” data (see Sec. 8.5 on p. 47) obtained with the 2.2-m La Silla telescope. The list of Brazilian run IDs needs to be obtained from USD.



7.5.2 Hiding sensitive data

The general rule at ESO is that metadata (i.e. headers of FITS file) of observations are available immediately after observation. For certain programmes this is a problem; for example, the observer may want to keep the coordinates of the observed source to be kept secret.

ESO Director General may allow this additional protection to certain programmes. Access to those programmes is controlled by the:

```
acproc:/opsw/dbcm/cron/Prog_id/Hide_proprietary_data.py
```

script.

The script searches `data_products` for data from those programmes and (1) sets `public_f` to NULL if the current date is before the release date, or (2) sets `public_f` to 'T' if the release date has already passed.

The input is a hand-edited list of programmes. Requests for additions to this list can only be made by the Office of the ESO Director General.

7.5.3 Looking for new instruments

The script:

```
acproc:/opsw/dbcm/cron/Olas_id/new_olas_ids
```

searches `data_products` for instruments missing in `observations..olas_ins` table (see Sec. 3.4.6 on p. 25) and notifies DBCM via email if such an instrument was found.

The script produces no output if there are no unknown instruments.

The script is run daily.

7.5.4 Managing HARPS data

The script:

```
acproc:/opsw/dbcm/cron/Harps/update-HARPS.csh
```

manages access rights to HARPS data, which are specifically excluded from being handled automatically with the `data_products` insert trigger (see Sec. 3.4.1 on p. 21).

The script identifies HARPS data which do not belong to HARPS GTO programme and applies to them the regular access rights (i.e. `public_f='T'` and `rel_date=exp_start+1 year`).



For the HARPS data identified as belonging to any of the GTO observing runs, it sets `rel_date` to `exp_start+6` years (the five additional years were negotiated between ESO and HARPS GTO consortium).

As a safety check, the script also verifies that no HARPS GTO data have `public_f` set to non-null value. This is to avoid situations in which manual database manipulation may have inadvertently left some data open.

The script is run daily.

7.5.5 Managing the `library..bibPublic` table

The script:

```
acproc:/opsw/dbcm/cron/Library/library_fill_bibPublic.script
```

recreates the `bibPublic` table. It is necessary for the `ads.script` (see below).

The script is run daily.

7.5.6 Daily ADS report for the ESO Librarian

The script:

```
acproc:/opsw/dbcm/cron/Library/ads.script
```

generates a report for the ESO Librarian. It consists of a join select statement on `bibPapers` (to retrieve the `BibCodes`) and on `bibPublic` (to retrieve the list of programme IDs associated with a particular `BibCode`). The result of the select statement is logged in `ads.log`. A few additional `awk` command are applied on the log file at the end of '`ads.script`', for display/format purposes, since ADS requires an ASCII file formatted in the following way:

```
Bibcode 1  Prog ID 1
Bibcode 1  Prog ID 2
Bibcode 2  None
Bibcode 3  Prog ID 1
Bibcode 4  Prog ID 1
Bibcode 4  Prog ID 2
Bibcode 4  Prog ID 3
...        ...
```

instead of, for example:

```
Bibcode 1  Prog ID 1  Prog ID 2
...        ...        ...
```



The output is directly piped to mailx command, which sends it to the librarian, who then put it on the web for ADS to download.

7.5.7 Managing WFCAM access rights

The script:

```
acproc:/opsw/dbcm/cron/Wfcam/update-WFCAM.csh
```

takes care of special release rules for the WFCAM data in the ESO Archive.

It updates `public_f` to 'R' (restricted to ESO Member States) and `rel_date=exp_start` in `data_products` for all new WFCAM data (i.e. with current date smaller than `exp_start+18` months). For those data it sets `qual_flag='UPDATED'` in order to not scan the whole WFCAM dataset every time the script is run.

Then for all WFCAM data for which `exp_start+18` months is greater than current date it sets `public_f` to 'T' (i.e. default, worldwide access).

The script is run daily.

7.5.8 Monitoring data_products

The script:

```
acproc:/opsw/dbcm/cron/Check_data_products/Check_data_products.py
```

monitors the `data_products` table and identifies entries with dummy values and sends a report via email to `dbcm@eso.org`.

The script currently checks for incorrect programme ID values. This script can be expanded if monitoring of other parameters becomes necessary.

The script is run daily.

7.5.9 Comparison of data_products with ISTs

The script:

```
acproc:/opsw/dbcm/replication/src/ist_data_products_check.py
```

compares the content of the tables in the observations database, to assure completeness in both the ISTs and `data_products`.

The script is run daily.



7.5.10 Comparison of NGAS with data_products

The script:

```
acproc:/opsw/dbcm/cron/Compare_dataproducts_ngas/compare_dp_ngas.sh
```

compares the contents of `ngas..ngas_files` with `observations..data_products` for previous month and sends a report to `dbcm@eso.org` if there are discrepancies between the databases.

The script is run monthly.

7.5.11 Filling /arc/headers directory

The script:

```
acd:/home/archeso/tools/h2db/dataclient/Launch_Fill_Headers
```

is used to extract the FITS file headers from the `observations..dp_headers` table and storing the resulting ASCII files in the `/arc/headers/...` directory. Those files are used for header display, activated by clicking on the link on the query result web page.

The script is run daily.

7.5.12 Filling dp_urls and ambient parameters in ISTs

The script:

```
acd:/home/archeso/tools/h2db/Dp_urls_Ambient_Param
```

serves two purposes. First, it fills the `observations..dp_urls` table with the URLs for header files, to be used for header display (see above).

Second, it calculates and fills the ambient parameter information in all ISTs: seeing (average and rms), moon illumination and `night_flag` (0=night, 1=twilight, 2=daytime).

This script is run three times per month.

7.5.13 Filling isaac_stacks table

The script:

```
acd:/home/archeso/tools/h2db/h5_inse3_isaac
```



is used to fill the `isaac_stacks` table, including the ambient parameters.

The script is run three times per month.

7.5.14 Filling `keyword_repository` with headers of raw files

The script:

```
acd:/home/archeso/dbcm/Extract/LaunchEX_LOAD
```

is used to extract header information for raw files from `observations..dp.headers` table and store it – via load file – in the `keywords_repository` table in the SAFIQ server.

The script is run daily.

7.5.15 Filling `keyword_repository` with headers of processed files

The script:

```
acd:/home/archeso/dbcm/Extract/LaunchEX_LOAD_MC_Science
```

is used to extract header information for processed (science and master calibration) files from NGAS, and store it – via load file – in the `keywords_repository` table in the SAFIQ server.

The script is run daily.

7.5.16 Garching \longleftrightarrow Chile database replication

The script:

```
acrep:/home/dbcm/scripts/checkOBsStatus.csh
```

verifies the consistency of the observation blocks status column in the `obrep2..run_states` table in Garching and Chile.

The script is run daily.

The script:

```
acrep:/opsw/dbcm/replication/src/check_ob_events_and_register.py
```

verifies the consistency of the contents of `ob_events` and `ob_events_register` tables in Garching and Chile.

The script is run daily.



7.5.17 Monitoring Garching → Chile replication

The script:

```
acrep:/home/dbcm/scripts/checkFE_UP.py
```

checks the consistency of replicated tables in obrep2, obrep2_hist and opc70 and user_portal databases, by looking for rows missing in the Chilean databases.

The script is run daily.

7.5.18 Monitoring Chile → Garching replication

The scripts:

```
acrep:/home/dbcm/scripts/checkObs.py
```

```
acrep:/opsw/dbcm/replication/src/check_dp_headers.py
```

```
acrep:/home/dbcm/scripts/ist_rep_check.py
```

monitor whether the contents of, respectively, data_products, dp_headers and ISTs in the observations database are correctly replicated from Chile to Garching.

The scripts are run daily.

7.5.19 Filling ops_logs with operation log records

Because of large volume, the contents of Paranal operation logs are not replicated. Instead, ASCII dumps of the daily increments in the Paranal ops_logs database tables – ops_logs, log_sources and log_files – are ftp-ed into the acbu3 machine (account: dha2dbcm).

The script:

```
acrep:/home/dbcm/scripts/opslogs_lsources_lfiles_into_safiq
```

searches for new files, converts them to SAFIQ load files and loads them into appropriate tables.

The script is run daily.

Chapter 8

What's special about...

8.1 ... APEX?

APEX is a joint project of ESO, Onsala Space Observatory in Sweden and Max Planck Institute for Radio Astronomy. Chile also gets some time on the telescope. Non-ESO data will be stored in the ESO Archive, but it needs to be remembered that ESO does not own those data.

APEX is different from what other ESO facilities in several ways:

1. Unlike VLT/VLTI, APEX allows running several “instruments” at the same time and storing the resulting data in the same file. To complicate things even more, those instruments have two components, referred to as “Front End (FE)” and “Back End (BE)”, which can in principle enter into different combinations with one another. The results of those combinations is closest to what is traditionally called an “instrument”; in APEX lingo they are referred to as FEBEs. Again, APEX can run many FEBEs at the same time and store them in the same file, although in vast majority of cases it is expected to have just one FEBE per file.

There are, however, two main categories of instruments: heterodynes and bolometers; APEX may also produce some test data. Thus, from DBCM’s point of view, APEX will produce three flavours of data, with `clas_id` either APEXHET, APEXBOL or APEXTEST.

2. Because of the possibility of having more than one FEBE in a file, loading of the APEX-related IST needed to be arranged in a way different from the traditional method. See Sec. 3.4.2 on p. 23 for detailed information.
3. Access rights to APEX data are non-standard. Therefore, the `data_products insert trigger` ignores APEX and data access information is set using external procedure. The following outlines the present arrangements:

ESO and Onsala data: One year from the release of the data to the observer for science data, immediate availability for calibration and test data, with worldwide access (`public_f='T'`). The access information is set by SAO as part of the release procedure; only files which are actually released are made visible.

Max Planck and Chile data: Data remain not visible to the outside world; `public_f=NULL`.



8.2 ... HARPS?

There is a special arrangement between ESO and HARPS consortium regarding access to HARPS GTO data. The data themselves will be available after normal one year proprietary period, but during additional five years the precise time information in the file headers will be hidden, in order not to allow the recovery of the precise radial velocity information.

This is achieved by generating special HARPS GTO files, which show midnight as the observation time, and from which all keywords containing clues enabling reconstruction of precise observation time are removed.

The following is a list of keywords which need to be removed or modified:

Keyword	Action
MJD-OBS	Round to nearest midnight (see note below)
CHECKSUM	Recalculate (see note below)
DATASUM	Recalculate (see note below)
ARCFILE	Replace with new value
DATE	Delete
DATE-OBS	Delete
UTC	Delete
LST	Delete
UT	Delete
ST	Delete
ORIGFILE	Delete
HIERARCH ESO DET EXP NO	Delete
HIERARCH ESO DET FRAM ID	Delete
HIERARCH ESO OBS START	Delete
HIERARCH ESO OBS TPLNO	Delete
HIERARCH ESO TEL AIRM START	Delete
HIERARCH ESO TEL AIRM END	Delete
HIERARCH ESO TEL ALT	Delete
HIERARCH ESO TEL AZ	Delete
HIERARCH ESO TEL AMBI FWHM END	Delete
HIERARCH ESO TEL AMBI FWHM START	Delete
HIERARCH ESO TEL AMBI PRES END	Delete
HIERARCH ESO TEL AMBI PRES START	Delete
HIERARCH ESO TEL AMBI RHUM	Delete
HIERARCH ESO TEL AMBI TEMP	Delete
HIERARCH ESO TEL AMBI WINDDIR	Delete
HIERARCH ESO TEL AMBI WINDSP	Delete
HIERARCH ESO TEL MOON RA	Delete
HIERARCH ESO TEL MOON DEC	Delete
HIERARCH ESO INS ADC1 END	Delete
HIERARCH ESO INS ADC2 END	Delete
HIERARCH ESO OBS ID	Delete
HIERARCH ESO OBS NAME	Delete
HIERARCH ESO TEL PARANG START	Delete
HIERARCH ESO TEL PARANG END	Delete
HIERARCH ESO TPL START	Delete

Note: care needs to be taken to not only modify the value of the keyword, but also the keyword comment field. In particular, the comment to the MJD-OBS keyword often provides the keyword value – i.e. the time of the beginning of the observation – in the ‘YYYY-MM-DDThh:mm:ss.sss’ format. Also, the comment fields in the CHECKSUM/DATASUM keywords often contain the timestamp showing when the checksums were calculated, which is usually very close to the observation time.



The scrambled files are made available to the users via special ESO Archive query form.

The procedure describing generating those files, uploading them to the Archive and uploading their metadata to the database can be found in Sec. 9.7 on p. 53.

8.3 ... GROND?

Data from the GROND instrument on the La Silla 2.2-meter telescope are wholly owned by Max Planck Institute. They are archived at ESO solely for safekeeping purposes. They are not to be made visible or downloadable, i.e. they all should have `public_f=NULL`.

8.4 ... WFCAM?

As part of an agreement between ESO and the United Kingdom, the astronomers from ESO member states have access to WFCAM/UKIRT data for 18 months. After that time, the data are available worldwide.

8.5 ... Brazilian data from the La Silla 2.2-m telescope?

As part of a Brazil-ESO agreement, some time on the La Silla 2.2-m telescope (WFI and FEROS) is reserved for exclusive use by astronomers from Brazilian institutions. Just like in the case of GROND (see Sec. 8.3 on p. 47) those data are archived at ESO solely for safekeeping purposes. They are not to be made visible or downloadable, i.e. they all should have `public_f=NULL`.

Chapter 9

Common non-scheduled tasks

This part of the documents summarises, for quick reference, actions performed by DBCM team members following most common external requests. The rationale for those tasks is explained in detail in the above sections of the document.

9.1 How to synchronise Chile and Garching databases?

9.1.1 Front-end databases

The script `checkFE_UP.py` (see Sec. 7.5.16 on p. 43) sends out an email warning with subject “Data Mismatch in...”. If this happens, the following steps need to be taken:

1. Contact EDAT.

First check with EDAT whether it is not a problem with a non-empty replication queue. If this is the case, one needs to wait for the queue to finish and rerun the `checkFE_UP.py` script. If no warning is received after the script run, the problem can be considered solved.

2. Contact persons in charge of `obrep2` and `opc70`.

Check with USD whether they are aware of any problems on the Chilean sites. If this is the case, the further action will depend on the nature of the particular problem.

3. Synchronisation

After the agreement from EDAT and/or USD, one can proceed with the synchronisations between Chile and Garching. Note that the following is an example of how to synchronise contents of `obrep2` or `opc70` databases; in case of problems with the `obrep2_hist` database, the procedure is identical, but the data flow is opposite: i.e. the `bcp` files need to be generated in Chile and transferred to Garching.

Chilean database with the contents of the database in Garching In case of



Creation of the tempdb tables. In Garching, after checking that the tables do not exist in tempdb one should execute a query listed in the warning email (the example below shows tempdb.. obs_runs and tempdb.. sched_rep):

```
SELECT * INTO tempdb.. obs_runs
FROM opc70.. obs_runs
WHERE (id=75046002)
go
```

```
SELECT * INTO tempdb.. sched_rep
FROM opc70.. sched_rep
WHERE (id=25032286)
OR (id=25032287)
go
```

Creation of the bcp files. After the creation of the tables in tempdb, bcp files need to be created. Checksums of the resulting files should also be calculated and provided to the file recipients.

```
> bcp tempdb..obs_runs out /tmp/obs_runs.bcp -c -t12t3 -r45r6 ....
> bcp tempdb..sched_rep out /tmp/sched_rep.bcp -c -t12t3 -r45r6 ....
> cksum /tmp/obs_runs.bcp
496182963 123 /tmp/obs_runs.bcp
> cksum /tmp/sched_rep.bcp
1833025732 234 /tmp/sched_rep.bcp
```

Putting the bcp files on the ftp server. The files created with bcp need to be made available to the Chilean site.

```
> ftp ftp.eso.org
Name (ftp.eso.org:lskywalker): anonymous
Password: xxxxxxxx
ftp> mkdir /pub/general/dbcm2paranal
ftp> cd /pub/general/dbcm2paranal
ftp> lcd /tmp
ftp> put obs_runs.bcp
ftp> put sched_rep.bcp
ftp> quit
```

Notification. Notify Paranal or La Silla via email. A sample email could look as follows:

Dear,

To enable synchronisation between La Silla and Garching, please please bcp in the following files in the associated tables and databases:

```
opc70 database:
-----
obs_runs.bcp: 1 row
sched_rep.bcp: 2 rows
```

You will find the files on the ftp server (ftp.eso.org) under the directory /pub/general/dbcm2paranal.



```
File checksums (output of cksum):
obs_runs.bcp: 496182963
sched_rep.bcp: 1833025732
```

The following bcp-in parameters and delimiters should be used:

```
bcp opc70..[table_name] in [path of the bcp files] -c \\
-t12t3 -r45r6 -U[user] -SOLASLS
```

9.1.2 Back-end databases

In case of problems with the Chile → Garching replication, the verification script (see Sec. 7.5.18 on p. 44) sends out an email warning with “REPLICATION ERROR!?” subject. The body of the message contains the listing of rows missing in the Garching database.

The procedure for recovery follows the one for the front-end databases, i.e. it requires verification whether the queues are empty and whether there are known problems with databases, followed by generation and transfer of bcp files. Of course, the data flow is opposite, i.e. bcp files need to be generated in Chile and ftp-ed to Garching.

IMPORTANT: The insert trigger is not activated during bcp-in operation. Therefore it is necessary to update `public.f` and `rel_date` for all affected files after bcp-in operation in `data_products` table. This needs to be done manually, and care needs to be taken to properly take into account any special release rights which may happen to apply to some/all of the files.

9.2 What to do when new instrument is identified?

When a previously unknown `olas_id` is entered into `olas_ins` table — which is done by the `data_products` insert trigger — the `new_olas_ids` cronjob will alert DBCM via email.

Further action depends on whether this is a valid `olas_id` (e.g. new instrument).

If yes, the `valid_flag` needs to be set to the appropriate value, '0', 'Y' or 'T' (see Sec. 3.4.6 on p. 25). This will stop the email reports to DBCM. Also, `ins_name` and `tel_seq` must be set appropriately. This must be done in coordination with the DBCM team leader. If there is no valid `ins_name`, then `ins_name` needs to be set to `char(0)`; the reason is to distinguish “empty” from “missing” (the latter is indicated by NULL).

If not, then the entire entry for that `olas_id` should be manually deleted from `olas_ins`. Typical example of this situation is if the data come from obvious tests, like 'DUMMY' or 'TEST'.

Please note that some instruments are special cases, e.g. HARPS and WFCAM.



9.3 How to deal with a non-scientific instrument?

A non-supported instrument should have `olas_ins.valid_flag` set to 'T'. All other `xxxx_flag` entries should all be set to 'N'.

9.4 How does an instrument become “officially supported” by the ESO Archive?

In order for a scientific instrument to be considered fully supported by the ESO Archive, the following prerequisites have to be met:

1. The instrument must have `observations..olas_ins.valid_flag` set to 'Y'.
2. An announcement that an instrument is to be supported must be first made in the *ESO Call for Proposals*.
3. DBCM must receive a letter from the Director of the La Silla Paranal Observatory stating that the instrument entered operations on a specified date (it usually is the beginning of an observatory period, i.e. 1 April or 1 October).
Such letter will likely have to be actively solicited.

The following actions need to take place:

1. An entry with the uppercase name of the instrument must be entered into `/opsw/conf/ISTS.conf`
2. A `/opsw/conf/<INSTRUMENT>.cfg` file must be created, by simply copying any other `.cfg` file in this directory.
3. In `observations..olas_ins`: `trigger_flag`, `in_ist_flag` and `hotf_flag` need to be set to 'Y'.
4. In `observations..data_products`: unless specifically excluded, all data taken after the date specified in the LSP Director's letter need to have `public_f` set to 'T' and `rel_date` set to either `dateadd(yy,1,exp_start)` or `exp_start`, depending on the value of `dp_cat`. This has to also be done for “carried under” data, i.e. Service Mode data taken before their corresponding observatory period actually began; such data can usually be recognised by having regular `prog_id` values.
5. A confirmation letter needs to be send to LSP Director, Instrument Scientist, QC, DFO head and SAO.

9.5 How to access databases in Chile?

For “select” access to Chilean databases, DBCM can use user “checker”.

Any manipulations on the Chilean databases need to be consulted with EDAT/USD/SDD and have to be performed by Paranal/La Silla staff. An exact query to be executed needs to be sent via email. It has to be accompanied by the explicit explanation for the rationale of the intervention.



9.6 How to prepare a data package?

Please note that the use of word “package” in this chapter should not be confused with “PI Packages” (or “PI Packs”), which are sets of data generated during regular observatory operations, prepared by the QC group and sent out to PIs by the SAO group.

The data package is a way of making a set of data public outside of normal observatory operations. The data packages are typically used to release Commissioning or Science Verification data before the instrument enters normal operations. Also, processed data from APEX are sent out as packages. Another typical use is release of non-standard data sets.

The authority to release data packages from Commissioning or Science Verification lies with the Director of the La Silla Paranal Observatory.

To request creation of a new package, the requestor should fill out and submit the form at the following URL:

http://archive.eso.org/archive/packages/packages_request.html

The table `packages` in observations database contains information about the package as a whole. The table `pkg_cont` contains information about each file in a package.

- Login on `acproc`: `ssh archeso@acproc`
 - Filling the `packages` table:
 - `cd /home/archeso/tools/packages/insert`
 - Update and execute `ins_packages1.Raw`. It will insert a new row into the `packages` table and create a new `pkg_id`.
 - Record the new `pkg_id`:

```
dsql -SES0ECF -Uarcheso
select max(pkg_id) from packages
```
 - Filling the `pkg_cont` table:
 - Create a directory for the package of the day, e.g. 06-11-15
 - `cd /home/archeso/tools/packages/insert/06-11-15/`
 - Copy `w.new` from the previous directory and update it with a list of files (names without the `.fits` extension) to be inserted into `pkg_cont`. This list should be provided by the requester of the package.
 - Run `w.new`. This will create `w.txt`, which contains two columns `dp_type` and `dp_id`, e.g.:

```
OBJECT APEXHET.2005-07-17T06:38:35.000
```
 - Add `con_catg` information given by the requester to the `dp_type` column. You can do it using an `awk` command line script:

```
awk 'BEGIN {FS=" "}{print "SPEC,RAW," $1 " " $2}' w.txt > w.txt_raw
```
- The output lines should look as follows:



```
SPEC,RAW,OBJECT APEXHET.2005-07-17T06:38:35.000
```

- Rename the resulting file to, e.g., apex.log.
- Move the file to /home/archeso/tools/packages/insert/.
- cd /home/archeso/tools/packages/insert
- Update and run ins_packages2_1createRaw. You will be asked for the pkg_id. The script will also read your logfile, e.g. apex.log. The script will create a sql file, e.g. ins_packages2_2.apex.
- Run the script created in the previous step. This will insert all rows of the package in pkg_cont.

- To check the created package, use the following query form:

```
http://archive.eso.org/wdb/wdb/eso/packages/form
```

```
http://archive.eso.org/wdb/wdb/eso/pkg_cont/form
```

- Before sending the link to retrieve the data of the package to the requester, you should first try to retrieve the data, e.g.:

```
http://archive.eso.org/wdb/wdb/eso/packages/query?pkg_id=4350
```

```
http://archive.eso.org/wdb/wdb/eso/pkg_cont/query?pkg_id=4350&tab_cont_catg=on
```

9.7 How to prepare HARPS GTO files with time information removed?

The scripts in

```
/opsw/dbcm/harps/scramblegto/
```

directory are used in preparation of “scrambled” HARPS GTO files to be released in accordance with the agreement between ESO and the HARPS consortium (see Sec. 8.2 on p. 46).

The scripts are stored in the CVS module HarpsGtoScramble.

The process involves modification of the MJD-OBS keyword to show midnight of the night during which the observation took place and removal of all keywords which either contain direct reference to the detailed observation time or contain clues which would allow restoring this information or restoring the sequence in which the files were taken. See Sec. 8.2 on p. 46 for the list of affected keywords.

The names of the files with the “scrambled” data do not contain the exact time stamp. Instead, they contain a pseudo-random code, for example:

```
Original file: HARPS.2007-04-01T23:45:01.234.fits
```

```
Scrambled file: HARPS.2007-04-02XFD996A7ED96A48.fits
```

Note that in the above example the date in the name of the scrambled file (correctly) refers to 2007-04-02 and not 2007-04-01.

The procedure is split into four separate steps:



1. Identification of files that need to be scrambled.
2. Downloading of original (unscrambled) files from NGAS.
3. Generation of scrambled files *and* of the list of scrambled file codes.
4. Inserting the filename+coded name pairs into the `observations..HARPS_GTO` table.

The procedure is split into separate steps because:

- Each of the first three steps can be time-consuming.
- Files generated in step 3 should be first ingested into NGAS before step 4 is performed. The reason is that step 4 makes files visible for the external users, and this should not be done before the files are physically available for downloading.

Assume we want to prepare scrambled HARPS GTO data from files taken between 2000-01-01 and 2008-01-01. The following example shows in detail the necessary steps:

1. Login to the `acproc` machine as user `archeso`.
2. `mkdir /staging/scrambleharps`
3. `cd /staging/scrambleharps`
4. `lgetharpsgtoids.sh "2000-01-01" "2008-01-01" > harps_gto_unscrambled.lst`
This step identifies the unscrambled data. Note: the script will ignore data already scrambled, thus there is no need to carefully select the start date - just make it far in the past and it'll be ok.
This can take few minutes.
5. `2downloadharps.sh harps_gto_unscrambled.lst`
This step downloads the original files from NGAS. If there are many files to download, it can take a while (it needs a few seconds per file).
6. `3scrambleharps.sh harps_gto_unscrambled.lst harps_gto_codes.lst`
This step generates codes for scrambled files, modifies the headers of the unscrambled files and writes out new FITS files, where file names are replaced with the codes.
7. `mkdir harps2ingest`
8. `mv HARPS.????-??-??X*.fits harps2ingest/`
The above two steps move the scrambled files into a separate directory (to make SAO's life easier).
9. Ask SAO to ingest the files into NGAS, and wait for confirmation that it has been done.
Sample email to SAO:



Dear SAO,

Please ingest scrambled HARPS GTO files into NGAS. They are located in directory `acproc:/staging/scrambleharps/harps2ingest/` and there is N of them.

Please notify DBCM when done.

Cheers

Do not proceed with the next step until confirmation from SAO has been received.

10. `4harpsintodb.sh harps_gto_codes.lst`

This step loads the relevant information into `observations..HARPS_GTO`, making the files visible through the HARPS GTO query form.

11. Cleanup:

```
cd /staging
\rm -rf scrambleharps/
```

12. Make a note in the DBCM Remedy ticket No. 00011 ("HARPS GTO: scrambling files"). Please leave the status of the ticket as Work In Progress (WIP).

9.8 Maintenance of the qc1 database.

9.8.1 Request for a new table in qc1 database

The tools to use are stored in CVS in the `qc1db` module.

Creation of a new table is done per request from a QC scientist. This request is accompanied by an Excel or csv file with the table definition.

The steps to follow are:

- Retrieve the `qc1db` module from CVS.
- If the request comes with an Excel file it is first necessary to export it into a csv file with the vertical bar '|' as the delimiter. The resulting csv file should have the name:

```
<instrument>_<QC purpose>.csv
```

e.g. `isaac_dark.csv`.

- Apply python script `fix_format.py` to the csv file:

```
python fix_format.py isaac_dark.csv > isaac_dark.csv
```

The script can be found under the directory `Qc1DbUtil`. This script removes any trailing space and prepares the file for the next step.



- Apply python script `make_sql.py` to the csv file:

```
python make_sql.py isaac_dark.csv > isaac_dark.sql
```

The script can be found under the directory `Qc1DbUtil`. This script generates the SQL commands to create the new table and generates all the metadata to be inserted into the `qc1_tables` and `qc1_columns` tables.

- Run the SQL script with a database tool, e.g.:

```
isql -S ES0ECF -U ..... -P ..... -i isaac_dark.sql
```

- Save the csv file used to generate the table inside the `Qc1Db<instrument>` directory (`Qc1DbIsaac` for the above example) submit it to CVS, to keep track of the changes.

9.8.2 Request for table modification in qc1 database

As before, every change a QC scientist submits is accompanied by an Excel or csv file with the table definition. The procedure is identical to the table creation procedure, except that:

- Prior to execution of the SQL script, the entire `create table` statement must be removed. It must be manually replaced with an appropriate `alter table` statement.

9.8.3 Maintenance of the qc1 web application

The code for the web application can be found in the directory `Qc1DbCgi`. The main files to be modified when a change is requested are:

- `qc1_browse.py`: It contains the code to browse the data of a table with given parameters.
- `qc1_hide`: It is used by the QC scientist to hide data from the tables, for example if the data was ingested in error.

The development environment is set under the machine `dmdarc1` in the `/home/web/docs/bin/` directory. To access the development web you need to use the following URLs:

```
http://arcdev.eso.org/bin/qc1_cgi  
http://arcdev.eso.org/bin/qc1_hide
```

The scripts for operation need to be deployed into the `/home/http/docs/bin/` directory on the `acweb` machine, using the procedure described in Sec. 7.1.3 on p. 35.



9.9 Merging user information in databases

Since it is possible for a user to have more than one account in ESO databases, it is sometimes necessary to combine the information spread between various accounts into one account. This is commonly known as “merging users”.

The actions necessary when merging users are:

- `user_portal`: deactivate the obsolete account. Includes adding “_M” to the username.
- `opc70`: assign proposals, web letters and referee information with the obsolete `user_id` to the correct one.
- `obrep2`: assign OBs related to the obsolete `user_id` to the correct one.
- `archive`: assign archive statistical data related to the obsolete account to the correct one.

Users are merged following requests from USD. This request comes in a list in which old and new user IDs are listed; e.g.:

```
old 53118 new 6380
old 4337 new 70600
```

The following steps need to be taken:

- Login to `dbcmdev1` as user `dbcm` and go to the `/home/dbcm/merge_users` directory.
- Convert the information from USD into two-column input file, in which the “new” account is in the first column, and the “old” account is in the second column.

IMPORTANT: care needs to be taken when preparing the input file, since the requests from USD come in free format and may look differently depending on who sends the request.

In the above example, the list to be fed to the merging script should look as follows:

```
6380 53118
70600 4337
```

- Execute the merging script:

```
sh merge_user.sh <your_file_name>
```

- Notify the requester that merging has been completed.



---oOo---