Gemini Science Archive

Conceptual Design Document (Revised Initial)

Version 04

22 May 2002

Norman Hill, Séverin Gaudet, Daniel Durand, David Schade, David Bohlender, Pat Dowler

National Research Council of Canada
Herzberg Institute of Astrophysics
Canadian Astronomy Data Centre

Felipe Barrientos, Felipe Richardson

Proyecto Gemini-Chile
Comisión Nacional de Investigatión Científica y Tecnológica

Table Of Contents

Created:August 2, 2001
Modified:August 31, 2001

Gemini Science Archive

Chapter 1 Introduction

1. Purpose

This document describes the conceptual design for the Gemini Science Archive. The design builds on the requirements documented in [5] and [6], and on the GSA to Gemini telescopes interfaces documented in [3].

2. Overview

This document is divided into the following chapters:

- Chapter 1 is an introduction to this document.
- Chapter 2 describes the overall design of the GSA software.
- Chapter 3 through Chapter 10 describe the design of the individual subsystems of the GSA.

3. Abbreviations and Acronyms

ADS	Astronomical Data Service
API	Application Programming Interface.
ASCII	American Standard Code for Information Interchange.
BDS	The abbreviation used for the Gemini Science Archive Bulk Data Storage subsystem.
CADC	Canadian Astronomy Data Centre.
CAT	The abbreviation used for the Gemini Science Archive catalogue subsystem.
CD-R	Compact Disc Recordable.
CD-ROM	Compact Disc Read Only Memory.
CFHT	Canada France Hawaii Telescope.
CGI	Common Gateway Interface.
CGPS	CanadianGalactic Plane Survey.
CoI	Co-Investigator (see glossary).
DAT	Digital Audio Tape.
DBMS	Data Base Management System.
DHS	Data Handling System.
DI	The abbreviation used for the Gemini Science Archive Data Ingest subsystem.
DP	The abbreviation used for the Gemini Science Archive Data Processing subsystem.
DPR	Data Processing Recipe (see glossary).

DR	The abbreviation used for the Gemini Science Archive Data Retrieval subsystem.
DSS	Digitized Sky Survey.
DVD	Digital Versatile Disk (sometimes referred to as Digital Video Disk).
DVD-R	Digital Versatile Disk (sometimes referred to as Digital Video Disk), recordable.
EMSS	Einstein Medium Sensitivity Survey.
ESA	European Space Agency.
FITS	Flexible Image Transport System.
FPRD	Gemini Science Archive Functional and Performance Requirements Document (see glossary).
FTP	File Transfer Protocol.
FUSE	Far Ultraviolet Spectroscopic Expolorer.
GB	Gigabytes (1073741824 bytes).
GEA	Gemini Engineering Archive (see glossary).
GMOS	Gemini Multi Object Spectrograph (see glossary).
GSA	Gemini Science Archive (see glossary).
GSC	Gemini Science Committee.
GSC	Guide Star Catalogue.
GSCG	Gemini Software and Controls Group.
GSRS	Gemini Software Requirements Specification.
GUI	Graphical User Interface.
HIA	Herzberg Institute of Astrophysics.
HROS	High Resolution Optical Spectrograph.
HST	Hubble Space Telescope.
HTTP	HyperText Transfer Protocol.
ICD	Interface Control Document.
IDE	Integrated Drive Electronics (see glossary).
IRAS	Infrared Astronomical Satellite (see glossary).
ISO	International Standards Organization.
IVO	International Virtual Observatory (see glossary).
JCMT	James Clerk Maxwell Telescope.
MB	Megabytes (1048576 bytes).
MEF	Multi-Extension FITS file.
NED	NASA/IPAC Extragalactic Database (see glossary).
NFS	Network File System.
NGST	Next Generation Space Telescope.
NIRI	Near InfraRed Imager (see glossary).
NVO	National Virtual Observatory (see also IVO).
OCDD	Gemini Science Archive Operational Concept Definition Document (see glossary).
OT	Gemini Observing Tool.

PI	Principal Investigator (see glossary).
PSF	Point Spread Function (see glossary).
RAID	Redundant Array of Inexpensive Disks (see glossary).
RDBMS	Relational Database Management System.
S/N	Signal to Noise ratio.
SCSI	Small Computer Systems Interface (see glossary).
SDD	Software Design Description.
SIMBAD	Set of Identifications, Measurements and Bibliography for Astronomical Data (see glossary).
SP	Science Program (see glossary).
SQL	Structured Query Language (see glossary).
TB	Terabytes (~1.1x10^{12} Bytes).
TBD	To Be Determined.
UI	User Interfaces. In this document, this refers to the archive user interfaces to the GSA.
USNO	United States Naval Observatory.
UML	Unified Modelling Language (see glossary).
URL	Universal Resource Locator.
WCS	World Coordinate System.
WPD	Work Package Description.
WWW	World Wide Web.

4. Glossary

Altair — A natural guide star adaptive optics system being built for the Gemini north telescope.

Archive — In the GSA documents, the term archive is the set of scientific astronomical data, intended to be accessible to the general astronomical community. An archive provides the tools necessary to allow an astronomer to easily identify and retrieve those data which will contribute to a specific area of study.

Associated Data Superset — This term refers to the concept of using data processing to merge several datasets into a single result. For example, several datasets containing data for the same field could be co-added to form a single associated data superset. Within the CADC, we tend to refer to associations as image data of the same field, however we would like to generalize the concept to include other associations, such as mosaics of adjoining images, averaging of spectral data of a target (where the spectra cover the same band), and concatenation of spectra of a target (where the spectra do not cover the same band). An associated data superset may be a virtual concept, with only a database entry describing how members could be combined, or it may be a FITS file which contains the results of data processing. See also Data Superset.

Association — See “Associated Data Superset”.

Astronomical Data Service (ADS) — This NASA-funded service provides access to four sets of abstracts: 626,460 abstracts from astronomical articles, 590,197 abstracts from space instrumentation and engineering articles, 928,206 abstracts from physics and geophysics articles, and 3,612 abstracts from the Los Alamos preprint server. They include abstracts from all major journals, many minor journals, conference proceedings, NASA reports, and many PhD theses. Users can query by author, object name (astronomy set only), keywords, words in the title, and words in the abstract text.

Astronomical Server URL — A proposed standard which will allow standardized access to astronomical catalogues over the world wide web. (See http://vizier.u-strasbg.fr/doc/asu.html.)

Attributes — See Descriptors.

Baseline Calibration Programs — An observing program used to collect baseline calibration data. (See also Observing Program.)

Calibration — This is the processing of raw data to remove instrument or environmental signatures. This is usually the first processing step towards making raw data scientifically useful.

Catalogue — There are two types of catalogues which are significant to the GSA.

- An archive catalogue contains descriptor values for each observation in an archive, and it is the primary tool used by astronomers when identifying data for retrieval. The descriptor values are properties of the dataset as a whole, e.g. WCS information, S/N for the data, primary target, etc.
- An object catalogue is a list of astronomical objects, with associated descriptors. In this type of catalogue the descriptors describe the objects. An object catalogue may be derived from the data in a single archive (e.g. an HST object catalogue is under development at the CADC), from a single project (e.g. EMSS), or they may be compiled from a variety of sources (e.g. SIMBAD, or NED). Entries in an object catalogue could have links to an archive catalogue, identifying the data from which the object descriptors were derived.

Classical Observing — This is a mode of observing where the execution of a science program is overseen by a PI or a CoI who is present at the telescope while the science program is being executed. See also Queue Observing, Remote Observing, and Service Observing.

Client — A computer program which uses a **"Server"** to carry out some action which it cannot otherwise do itself. The client commands the server and uses it as a system resource.

Co-Investigator — A person collaborating with a Principal Investigator (see glossary) on a project.

Common Gateway Interface (CGI) — The Common Gateway Interface, or CGI, is a standard for external gateway programs to interface with information servers such as HTTP servers. This interface is used when it is necessary to have web pages with non-static information.

Configuration — A description of the state of one or more components of a system. For example, a Gemini instrument controller configuration might contain the location of the filter wheel, position of the grating etc. A state described by a configuration can be dynamic. For example an instrument could be continuously changing the state of the chopper in one "configuration".

Context diagram — A special case of a "Data Flow Diagram (DFD)" in which a single bubble represents the entire system. All boxes in a context diagram represent objects outside the system being described.

Cross catalogue query — A query that correlates the content of two or more catalogues. An example of a cross archive query would be "find all NIRI observations in the Gemini

Science Archive where there also exists at least one observation in the JCMT SCUBA catalogue with an overlapping field of view". (See also Multi-archive query.)

Cross archive search — See "Cross catalogue query".

Data Handling System (DHS) — That part of the Gemini Control System responsible for displaying, processing, saving and archiving data in a standard way.

Data Flow Diagram (DFD) — A diagram showing how data is handled by a software system.

Data label — A unique label which is used to identify a dataset created by the Gemini Telescopes. A data label can be used to uniquely identify the dataset for as long as the dataset exists.

Data Mining — The extraction of information or knowledge from large databases.

Data Processing Recipe (DPR) — A recipe describing how to process a particular type of data.

Data Processing Recipe Instance (DPRI) — A Data Processing Recipe, applied to a specific set of inputs. A DPRI identifies a recipe, the input data, and any input parameters for the recipe.

Data Quality — A single descriptor which gives an indication of the usability of a dataset. Tentatively, the data quality for data from the Gemini telescopes should categorize the data quality as follows:

- The data does not meet Gemini standards and has no scientific value. This data will not be visible to GSA users.
- The data meets Gemini standards, but does not meet the standards required for the science program.
- The data meets Gemini standards and meets the standards required for the science program.

Data superset — A data superset is either a single dataset (see glossary), or an associated data superset (see glossary).

Dataset — Data created from a single OBSERVE command. A dataset consists of header data describing the dataset and one or more frames. Each of the frames has a separate header containing any information specific to the individual frames. All data from a dataset is eventually stored in a single FITS file, with each frame stored in a FITS extension.

Descriptors — The data which describes a dataset. This would include data from a dataset's FITS file headers, and data from other sources, such as weather data, and proposal information.

DHS database — The DHS database contains a record of datasets collected by the Gemini telescopes, along with a selected set of descriptors, information about archive media created by the DHS, etc.

External Requirements — These are requirements which are beyond the scope of the GSA implementation and operations, and the Gemini project as a whole. See also Functional Requirements and Usability Requirements.

Facility Program — An observing program used to collect system verification observations, or other maintenance observations needed by Gemini staff. (See also Observing Program.)

File Transfer Protocol (FTP) — A widely available protocol for transferring data files over the internet.

FITS extension — A FITS extension is part of a multi extension FITS format file. FITS extensions are used to allow more than one image or table to be stored in a single FITS file. Each FITS extension has a separate header area, which contains meta data describing the data content of the extension.

Note: The content of a FITS extension could be meta-data further describing the data in the primary FITS data, or describing the data in another FITS extension.

FITS format file — A file obeying the Flexible Image Transport System (FITS) standard (see http://fits.gsfc.nasa.gov). For the GSA project, the term “FITS format file” will refer to a file which resides on disk.

FITS header — A FITS header as written to a FITS format file. This consists of a series of 80-byte ASCII card images. Each record contains a unique 8 character keyword identifying the data item, the value of the item encoded as an ASCII string, and an optional comment which describes the item. The value of a data item must be a single integer, floating point or string value.

Functional Requirements — These are the requirements the GSA must meet for all of the science described in the GSA OCDD to be possible and practical. See also Usability Requirements and External Requirements.

Gemini Engineering Archive (GEA) — The purpose of the GEA is to collect data from all of the Gemini systems and to allow efficient retrieval of the data for the purpose of analysing the system behaviour. To avoid duplication, the GEA may contain pointers to data in existing databases.

Gemini Meta-Data Database — This is a Sybase database which stores meta-data describing Gemini datasets. This database is replicated to the GSA, and is the primary method for transferring meta-data from Gemini to the GSA. The Gemini Meta-Data Database is described in the Gemini Telescopes to Gemini Science Archive Interface Control Documents.

Gemini Multi Object Spectrograph (GMOS) — A Gemini instrument that will allow simultaneous collection of spectra from many objects in a field of view.

Gemini Science Archive (GSA) — An archive system intended to preserve scientific data collected by the Gemini Telescopes. After the proprietary period for the data has expired, the data is made available to the general astronomical community.

Gemini Science Archive Functional and Performance Requirements Document (FPRD) — This document describes the functional and performance requirements to be included in the Science Archive, and describes functional and performance requirements required from other Gemini Systems.

Gemini Science Archive Operational Concept Definition Document (OCDD) — This document describes the science cases for which the Science Archive is designed, contains the scenarios which show how a GSA user would interact with the GSA in order to collect the data required for each of the science cases, and lists the user requirements which arise from the scenarios.

Gemini Science Archive User — For the purposes of these documents, GSA users are considered to be professional astronomers, or students in a university level astronomy program. The GSA is not intended to provide data to the general public.

Hipparcos — Hipparcos is a pioneering space experiment dedicated to the precise measurement of the positions, parallaxes and proper motions of the stars. The intended goal was to measure the five astrometric parameters of some 120 000 primary program stars to a precision of some 2 to 4 milliseconds of arc.

IDE — A computer disk drive interface, usually used on personal computers. The IDE interface allows the cost of computer peripherals to be reduced, at the cost of performance.

Infrared Astronomical Satellite (IRAS) — An infrared space telescope. It conducted an all-sky survey at wavelengths ranging from 8 to 120 microns in four broad-band photometric channels centred at 12, 25, 60, and 100 microns. Some 250,000 point sources were detected. A catalogue of small (< 8') extended sources gives the characteristics of some 20,000 objects. An atlas of images covering the entire sky gives the absolute surface brightness at each of the four survey wavelengths.

Instrument — A device, consisting of a detector and a means of dispersing and/or focusing light onto that detector, designed to extract a particular type of information from the radiation gathered by the Gemini telescope. An instrument is specially designed for astronomy research. The wavefront sensors on the Gemini telescope can be regarded as an instrument.

Instrument Control System (ICS) — That part of the Gemini Control System which is responsible for controlling and operating a scientific instrument. There is a separate Instrument Control System for each available scientific instrument.

Interactive Observing — A mode of observing where details of the sequence of events are decided in real-time while performing the observation. See also Planned observing.

Interface Control Document (ICD) — A document which either describes a general interface between Gemini systems, or which describes in detail the interface between two specific Gemini systems.

International Virtual Observatory (IVO) — Also known as the National Virtual Observatory (NVO). This is a proposed virtual observatory which will incorporate catalogues and archive data from a variety of surveys and observatories.

ISO 9660 — The international standard defining the CD-ROM data format.

Logging — The act of keeping a record of happenings during an observing session. “Logging” encompasses the observing log, system history records and any engineering logs made for the day crew.

Meta-data — Meta-data is data which describes data. This can be a recursive concept in that meta-data can be described by meta-data.

Meta-Data Database — See “Gemini Meta-Data Database”.

Multi-catalogue query — A query which spans more that one catalogue, but does not directly correlate the content of the two catalogues. An example of a multi-catalogue query would be “find all images of a specified object in the Gemini catalogue, in the HST catalogue, or in the CFHT catalogue. (See also cross archive query.)

Multi-archive search — See “Multi catalogue query”.

NASA/IPAC Extra Galactic Database (NED) — NED is a catalogue built around a master list of extragalactic objects for which cross-identifications of names have been established, accurate positions and redshifts entered to the extent possible, and some basic data collected. Bibliographic references relevant to individual objects have been compiled, and abstracts of extragalactic interest are kept on-line. Detailed and referenced photometry, position, and redshift data, have been taken from large compilations and from the literature. NED's data and references are being continually updated, with revised versions being put on-line every 2-3 months.

NICMOS — An HST instrument for both imaging and multi-object spectroscopy.

NIRI — A Gemini north instrument designed for near infrared imaging and spectroscopy.

Non-conforming instrument — An "Instrument" which does not conform to the Gemini hardware or software requirements. Such an instrument is only allowed as a **"Visiting instrument".**

Observation — An observation is the smallest schedulable event for the Gemini telescopes. An observing program may consist of multiple observations, and an observation may consist of several datasets, each with a different instrument and/or telescope configuration.

Observing log — The observing log is a list of all observations taken and key attributes of the observations, including time of observation, observer comments, object of observation, etc.

Observing Modes — As defined in the Gemini Software Design Description, p5-46, the observing modes are: interactive, queue-based, service and remote.

Observing Program — A formal description of a plan for using a Gemini Telescope. Observing programs are produced and executed with the Gemini Observing Tool. An observing program is suitable for near-automatic execution. It usually consists of an unordered group of Observations.

There are three types of observing program: Phase II observing programs (also known as science programs), facility programs, and baseline calibration programs. The specifics of each type of observing program is described separately in this glossary.

Observing Tool — The primary astronomer interface of the Gemini control system.

Phase I Proposal — A document created by a PI and submitted to the Gemini Time allocation process. The document is in a standard format, and describes an experiment proposed by the PI. The proposal is used to determine the allocation of Gemini telescope time to experiments.

Phase II Observing Program — This is an observing program developed by a PI to carry out the observations needed for a Phase I proposal. (See also Observing Program and Phase I Proposal.)

Planned Observing — Planned observing is a science program where the details of the observing process are planned in advance. Planned observing is required for both Queue Observing and Service Observing, and is strongly encouraged for classical users. See also Interactive Observing.

Point Spread Function — A measure of image quality.

Preview data — A representation of a dataset intended for display to an archive user. The intention of preview data is to allow the archive user to quickly view an image or spectral plot, giving an indication of the data quality and usefulness. Preview data is a calibrated, compressed (possibly with lossy compression) version of the data.

Principal Investigator (PI) — The Gemini user identified as being the person who has sole access to the science archive data associated with his/her science program for the proprietary period.

Proprietary Period — The period in which data in the GSA is not released to the general public. From the point of view of the GSA operations, the initial public release date for each dataset is set by Gemini, and may be changed by Gemini at any time.

Prototype — An implementation of the functionality of some aspect of the GSA prior to the delivery of the fully finished system. The purpose of the prototype is to clarify the design of the GSA.

Queue Observing — In this mode of observing, an astronomer submits a science program to an observatory, and the science program is added to a list of science programs waiting to be executed. The execution of the science program is overseen by observatory staff. The scheduling of the program execution can by based on many factors, including time in the list, current observing conditions, and the conditions required by the science program. Queue mode observing also requires that the science program consists of planned observations. See also Classical Observing, Remote Observing, and Service Observing.

RAID — A magnetic disk storage system which uses an redundant array of inexpensive disks to provide large, inexpensive, fast, and reliable magnetic disk storage.

Recipe — See Data Processing Recipe.

Remote Observing — This is similar to Classical observing, except that the Observer oversees the execution of the science program from a remote site (usually their home institution). See also Classical Observing, Queue Observing, and Service Observing.

Rock Ridge — An extension to ISO 9660 to allow Unix-specific information such as links, file attributes and native Unix naming to be stored along with the ISO 9660 information.

ROSAT — ROSAT is an X-ray space telescope which went into operation in June 1990. As the primary objective ROSAT has performed the first all-sky survey with an imaging telescope in the soft X-ray band of 0.1 keV - 2 keV (corresponding to wavelengths of 100 Å - 6 Å) as well as in the adjacent extreme ultraviolet region of 0.04 keV - 0.2 keV (corresponding to wavelengths of 300 Å- 60 Å). More than 60,000 X-ray sources have been detected with the ROSAT all-sky survey, larger by almost two orders of magnitude than the 840 sources of the catalogue of the previously largest all-sky survey of the HEAO-I satellite.

Science Program — See Observing Program and Phase II Observing Program.

SCSI — A computer interface system usually used on higher performance computer systems.

Server — Any computer program which carries out some service for one or more client computer programs. Typically, a server will be on a remote machine, and therefore it has direct access to resources not directly available to client programs. A server is completely subservient to its clients. It responds to commands from the clients but does not initiate dialogue with clients.

Service Observing — This is similar to queue observing, in the sense that the execution of a science program is overseen by a staff member from one of the Gemini national project offices, instead of one of the members on the science team who submitted the science program. Service observing also requires that the science program consists of planned observations. See also Classical Observing, Queue Observing, and Remote Observing.

SIMBAD — The SIMBAD astronomical database provides basic data, cross-identifications and bibliography for astronomical objects outside the solar system. SIMBAD can be queried by object name, coordinates and other criteria (filters).

Status — Attributes which define the state of a hardware or software system.

STIS — An HST instrument for both imaging and spectroscopy.

Structured Query Language (SQL) — A standard language, common in commercial database management systems, used for specifying how to search a relational database for items matching certain criteria. The criteria can involve arithmetic and logical combinations of record properties (tuples).

Telescope — In the Gemini software documentation the word "Telescope" is used to refer to the Gemini 8m telescope together with all of its associated peripheral devices. These include the telescope mount, the primary and secondary mirrors, together with their support and control structures, the Cassegrain rotator, and all the devices associated with acquisition and guiding and with the adaptive optics. Some of the latter devices can also be regarded as instruments.

Transportable media — Media that will contain data for transport to the observers home facility, and to the archive centre. This is base-lined to be either DAT or EXABYTE tapes, or CD-ROM.

Unified Modelling Language — A notation for modelling software systems using an object oriented methodology. This methodology is a unification of the methodologies developed separately by James Rumbaugh, Grady Booch and Ivar Jacobson.

Universal Resource Locator (URL) — The unique address of an internet resource, such as a hypertext document, gopher page, mail address or a computer. The address adheres to a communications protocol described in RFC1738 (http://www.cis.ohio-state.edu/htbin/rfc/rfc1738.html).

Unix — A standard host-level operating system. The Gemini baseline for "Unix" is "Solaris" marketed by Sun Microsystems.

Usability Requirement — This is a requirement which does not enable any new functionality, but which make the GSA easier to use.

Visiting instrument — A self-contained **"Instrument"** which is brought to the Gemini telescope for temporary use. This type of instrument is intended only for the use of the instrument owner.

WFPC2 — An HST imaging instrument.

Work package — A well-defined unit of work, described by a workscope document. A work package usually consists of the development of a subsystem or a layer of utilities, which can be devolved to a collaborating group.

World Coordinate System (WCS) — A mapping of the pixel column/row (x/y) coordinate system of an image into a coordinate system that has meaning in the real world. In astronomy, the world coordinate system is most often right ascension and declination, but could also include other coordinate systems, such as galactic coordinates. A WCS could also be associated with a spectral dataset, where one of the world coordinates would be wavelength.

World Wide Web — A part of the internet designed to allow easy navigation of the network of computers through the use of GUI's and hypertext links between different URL's or addresses.

5. References

[1] *Engineering Archive Requirements*, Phil Puxley, Dayle Kotturi, Jim Oschmann, Bret Goodrich, Gemini, Daniel Durand, David Schade, David Bohlender, Séverin Gaudet, HIA, http://www.gemini.edu/documentation/webdocs/spe/spe-c-g0073v02.pdf

[2] *The Gemini Data Handling System User Manual Documents*, Norman Hill, Jennifer Dunn, Séverin Gaudet, Shannon Jaeger, HIA, http://www.hia.nrc.ca/pub/Gemini_HIA/DHS/current/userdoc-v3.pdf

[3] *Gemini to Gemini Science Archive Interface Control Documents*, gsa_ICD/01, Norman Hill, Séverin Gaudet, Daniel Durand, David Schade, David Bohlender, NRC, Felipe Barrientos, Felipe Richardson CONICYT, Kim Gillies, Inger Jørgensen, Gemini, in preparation

[4] *Gemini Science Archive Conceptual Design Study Work Scope* No. 9414257-GEM02012, http://www.hia.nrc.ca/pub/Gemini_HIA/GSA/Workscope/workscope.pdf

[5] *GSA Operational Concept Definition Document*, gsa_OCDD/04, Norman Hill, Séverin Gaudet, Daniel Durand, David Schade, David Bohlender, NRC, Felipe Barrientos, Felipe Richardson CONICYT, http://www.hia.nrc.ca/pub/Gemini_HIA/GSA/OCDD/OCDD_initial.pdf

[6] *GSA Functional and Performance Requirements Document*, gsa_FPRD/03, Norman Hill, Séverin Gaudet, Daniel Durand, David Schade, David Bohlender, NRC, Felipe Barrientos, Felipe Richardson CONICYT

[7] *GSA data rates*, eMail message from Ted von Hippel, July 7 2000, http://www.hia.nrc.ca/pub/Gemini_HIA/GSA/References/datarates.txt

[8] *ICD-03, Bulk Data Transfer*, Norman Hill, Séverin Gaudet, NRC, http://www.gemini.edu/documentation/webdocs/icd/icd_03.pdf

[9] *SExtractor; software for source extraction*, E. Bertin, Institut d'Astrophysique de Paris, S. Arnouts, European Southern Observatory, http://www.hia.nrc.ca/pub/Gemini_HIA/GSA/References/sexarticle.ps

[10] *Galfit*, David Schade, Personal communication

[11] *Gemini Science Archive Prototype Design*, http://www.hia.nrc.ca/pub/Gemini_HIA/GSA/Prototype.

[12] *PowerDesigner V 7.5 On-line Documentation*, http://manuals.sybase.com:80/onlinebooks/group-pd/pdd0750e

[13] *NED and SIMBAD Conventions for Bibliographic Reference Coding*, M. Schmitz, G. Helou, P. Dubois, C. LaGue, B. Madore, H.G. Corwin Jr, S. Lesteven, http://cdsweb.u-strasbg.fr/simbad/refcode.ps

[14] *Task List for Enhancements to DHS 0.18*, Norman Hill, Severin Gaudet, Shannon Jaeger, NRC, http://www.hia.nrc.ca/pub/Gemini_HIA/DHS/proposal/dhs-0.18_enhancements-v2.pdf

[15] *Visual Modeling with Rational Rose and UML*, Terry Quatrani, Addison-Wesley, 1998

[16] *CADC archive home page*, http://cadcwww.hia.nrc.ca

[17] *WDBI - Web DataBase Interface Installation and Users Guide*, Jeff Rowe, http://giants.stanford.edu/wdbi

[18] *The NASA Astrophysics Data System*, http://adswww.harvard.edu.

6. Design Philosophy

The GSA design has the following goals:

- Allow the Gemini user community to retrieve data collected with the Gemini telescopes.
- Easy maintenance and extendability.
- To reuse as many components as possible from existing CADC archive systems and the Gemini Data Handling System, in order to reduce cost and development time.
- Support parallel design and implementation of the GSA.

In order to achieve these goals, a modular design is described here, with well-defined interactions between the various components.

7. Relation to existing archive systems

The GSA will incorporate components from existing archives currently supported by the CADC, and many of the enhancements made to the CADC archiving systems for the GSA will be reflected in the operations of the other archives at the CADC.

The other archives and catalogues located at the CADC provide the potential for cross archive and cross catalogue searches. Other archives hosted at the CADC include the HST, CFHT, FUSE, CGPS, and JCMT archives. Catalogues hosted by the CADC include GSC I, GSC II, USNO, DSS I, DSS II, and a clone of the VizieR catalalogue system.

8. Implementation Technology

Based on the experience of the CADC we are leaning towards the following implementation solutions.

- HTML web pages for user interfaces.
- Apache web server.
- Sun/Solaris & PC/Linux computer systems.

  These systems were selected because they are the technologies currently in use at the CADC.

  – The Sun/Solaris systems provide the reliability and performance required for the data storage, database, and user interface servers required for an archive system.
  – The PC/linux systems provide a large number of affordable CPU cycles required for massive data processing.
- The computer systems will be linked with a LAN, in a configuration which minimizes the number of systems required to maintain user services. The system currently in use as the CADC has three critical computers: the database server, the web server, and the jukebox server. The GSA will be implemented in a way which does not expand the list of critical systems, and if GSA data is stored on magnetic disk, the jukebox server will not be critical to GSA operations.
- Sybase Adaptive Server Enterprise (ASE) relational database server.

  This is the database server currently being used by the CADC for our existing archives. Sybase also has a replication server which can be used to copy databases between sites. This feature is described in [3] as being used to replicate meta-data tables from Gemini to the GSA, and it could also be used to replicate the GSA database tables to mirror sites.

- Bulk data storage.
  - Long term, stable storage of bulk data will be on optical media. The current standard for optical media is 4.7 GB DVD.
  - On-line data storage will be on magnetic disk. The cost of magnetic disk storage is now comparable to near-line DVD jukeboxes, and historically the cost of magnetic disk storage has been dropping faster than the cost of storage in optical disk jukeboxes.
- The languages used to implement the GSA will be java, C++, or C. Java is the preferred language for new development, however some of the system may be developed in C++ or C if it is necessary to maintain compatibility with existing CADC systems. Because of the preference for object oriented languages, the design will be object oriented. (Note that an object oriented design can be implemented with a language that is not object oriented, and so the whole design can be object oriented, even if parts of the code are not.)

9. Document style

9.1 Software design

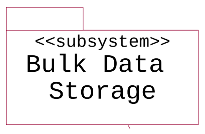
The software components of the GSA are designed with the assumption that they will be implemented in an object oriented language (probably Java or C++). The design is documented using the Rational Rose modelling tool, using UML notation (see [15]). The Object Flow diagrams described in Section 9.1.1 on page 13, and the Component Diagrams described in Section 9.1.2 on page 13 are produced using Rational Rose.

9.1.1 Object Flow Diagrams

Object flow diagrams show the relationships between objects in a system and the activities or processes which act on those objects. Figure 1 on page 14 shows an example of an activity diagram. In the example, the rectangular symbols represent data files, and the rounded symbols represent data processing tasks which will take files as input, and create new files as output. Stacked rectangles indicate that a set of similar objects are being manipulated. The dashed lines represent the objects flowing into the activity, being transformed, and new objects flowing out of the activity.

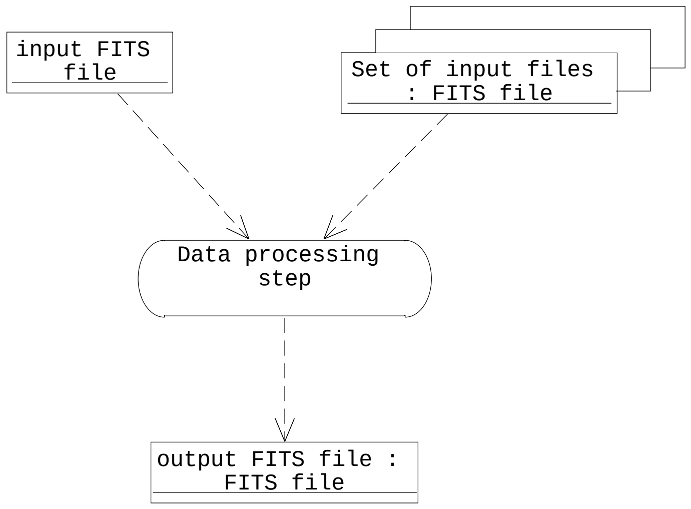
9.1.2 Component Diagrams

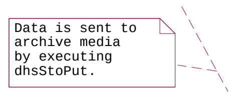
The GSA has been divided into subsystems, which are further divided into components (executable programs or class libraries). The relationships between these components are shown using component diagrams. An example of a component diagram is shown in Figure 2 on page 15. Each of the component diagrams has a shaded area which identifies the parts of the diagram which are included in the subsystem under discussion. The arrows between components are “uses” relationships, indicating that one component uses the services of another component. The icons used in component diagrams are:



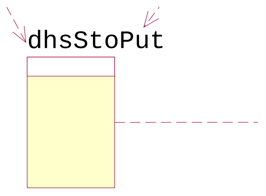
This icon represents a complete subsystem of the GSA. The components which make up the subsystem will be described in the chapter describing that subsystem.

FIGURE 1. Sample Object Flow Diagram

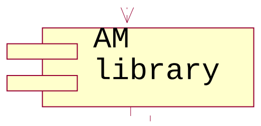




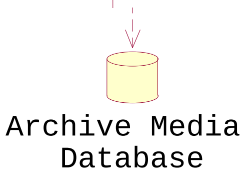
This is a comment describing an object or relationship on the diagram.



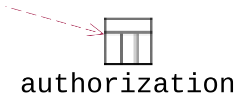
This icon represents an executable program which is part of the subsystem. The name of the program is given above the icon.



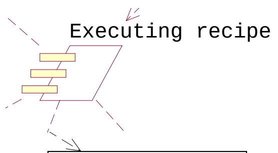
This icon represents a subroutine library or a class library which is a part of the subsystem.



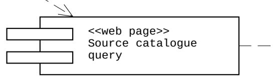
This icon represents one of the GSA databases. The name of the database is given in the text below the icon.



This icon represents a single table in the GSA databases.

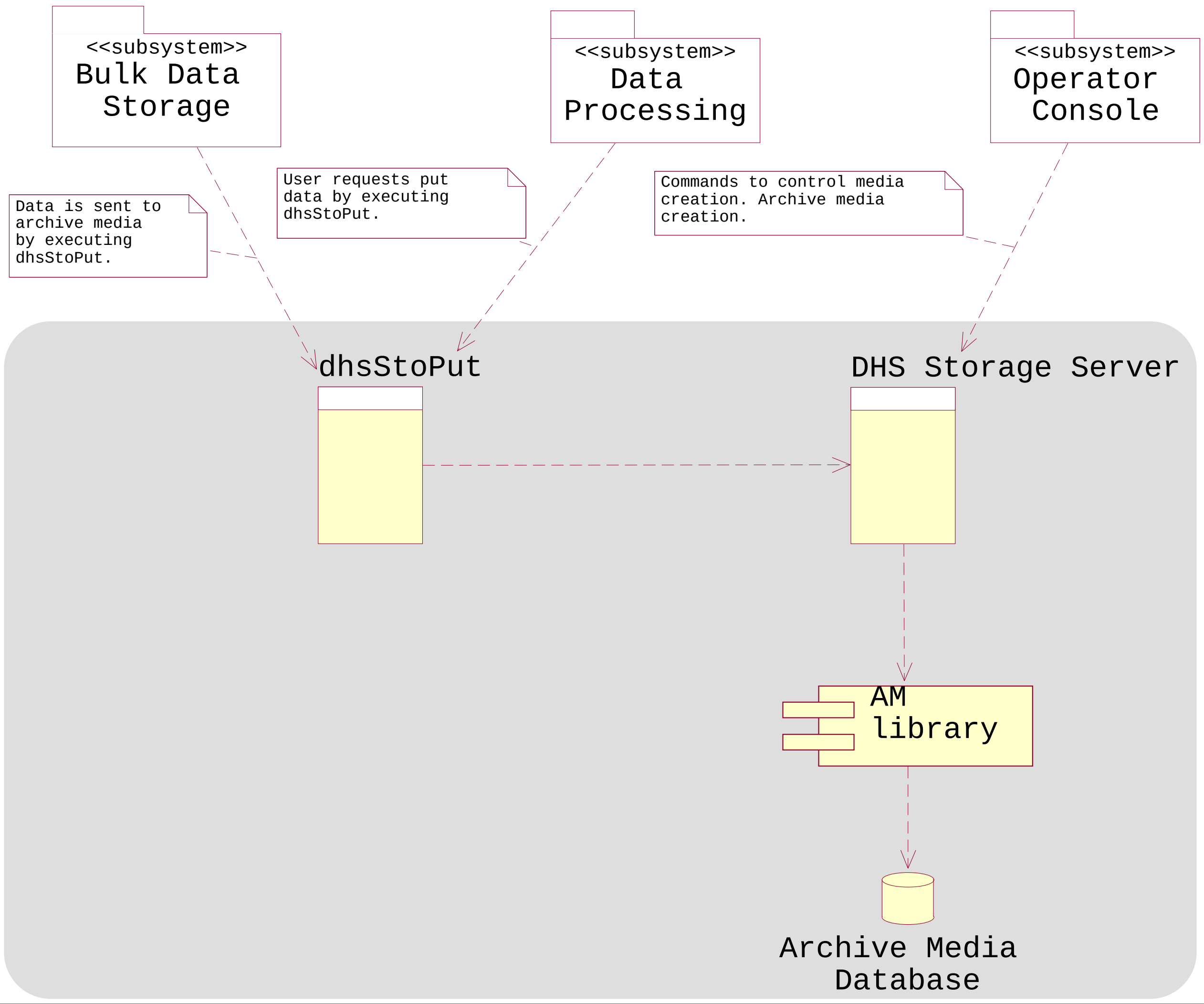


This icon is used in the data processing system chapter to represent a recipe instance that is being executed.



This icon is used to represent a web page.

FIGURE 2. Sample Component Diagram


<<subsystem>>
Bulk Data Storage
<<subsystem>>
Data Processing
<<subsystem>>
Operator Console
Data is sent to archive media by executing dhsStoPut.
User requests put data by executing dhsStoPut.
Commands to control media creation. Archive media creation.
dhsStoPut
DHS Storage Server
AM library
Archive Media Database

9.2 Database Table Descriptions

The database tables of the GSA are described using the Sybase PowerDesigner database design software (see [12].) The table relationships diagrams described in Section 9.2.1 on page 15, and the table descriptions described in Section 9.2.2 on page 16 are produced using PowerDesigner.

For organizational purposes, the tables are grouped into packages. Generally, each package corresponds to a GSA subsystem. The only exception to this is the catalogue subsystem, which further divides the tables into five sub-packages (this is only done because there are a large number of tables in the subsystem). Each package is described in a separate section of the document.

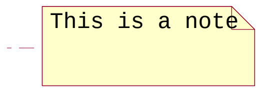
9.2.1 Table Relationship Diagrams

An example of a table relationship diagram is shown in Figure 3 on page 17. Arrows joining the table icons represent references between the tables. At the base of each reference arrow is an indication of the cardinality of the join, for example “1..*” indicates that each row in the first table joins with one or more rows in the second table. Text on the arrows indicates which fields of the tables are being joined.

Each table is represented by a rectangular icon in the diagram. Tables from other packages are usually represented as small rectangles, without any details on the column content. The small symbol in the lower left corner of a table icon indicates the table is defined in another package. The icons representing tables in the current package contain three columns of information:

- The name of each field in the table.
- The data type in the field.
- Information describing the keys on each column in the table. Keys provide information describing which columns in a table are used in joins with other tables. There are three types of keys described in this column:
  - Primary key columns are those columns which are usually used to locate rows in the table. Primary key columns are identified by the abbreviation “pk” in the list of keys.
  - Alternate key columns are those which may also be used to locate rows in the table. Alternate key columns are identified by the abbreviation “ak” in the list of keys.
  - Foreign key columns are columns which provide information which is used to locate data in other tables. Foreign key columns are identified by the abbreviation “fk*n*” in the list of columns, where *n* is an counter to ensure that the foreign key names are unique.

Notes are written in note icons, with a dotted line attached to the item referred to by the note:



9.2.2 Table descriptions

Each database table in the GSA is documented with text describing the function of the database table, a table describing each of the columns in the database table, and an optional table describing the indexes defined for each database table. (Some tables which already exist have indexes defined, other tables will have indexes defined later in the development of the GSA.)

9.2.3 Sybase Data Types

The database tables assume that the Sybase Adaptive Server Enterprise (ASE) server relational database management system will be used to store the database tables. The properties of the Sybase data types specified for the fields in the tables described in this document are described in Table 1 on page 17.

10. Data Supersets

The basic unit of storage in the GSA is the data Superset, and to understand the operation of the GSA, it is necessary to understand what a data superset is. Data supersets include both individual datasets as received from Gemini, and associated data supersets identified by the GSA Data Processing Discovery Agent described in Chapter 5, section 4.3 on page 87. The individual datasets are described elsewhere (see [8]). Associated data supersets will correspond to recipe instances which group several datasets into a single entity. In Figure 14 on page 74, an associated data superset will be assigned to the “Image Stacking” recipe instance. A data Superset name will be assigned to the output of the recipe, and information about the data superset will be the stored in the GSA catalogues. In those cases where the individual datasets have very little value individ-

FIGURE 3. Sample table relationship diagram

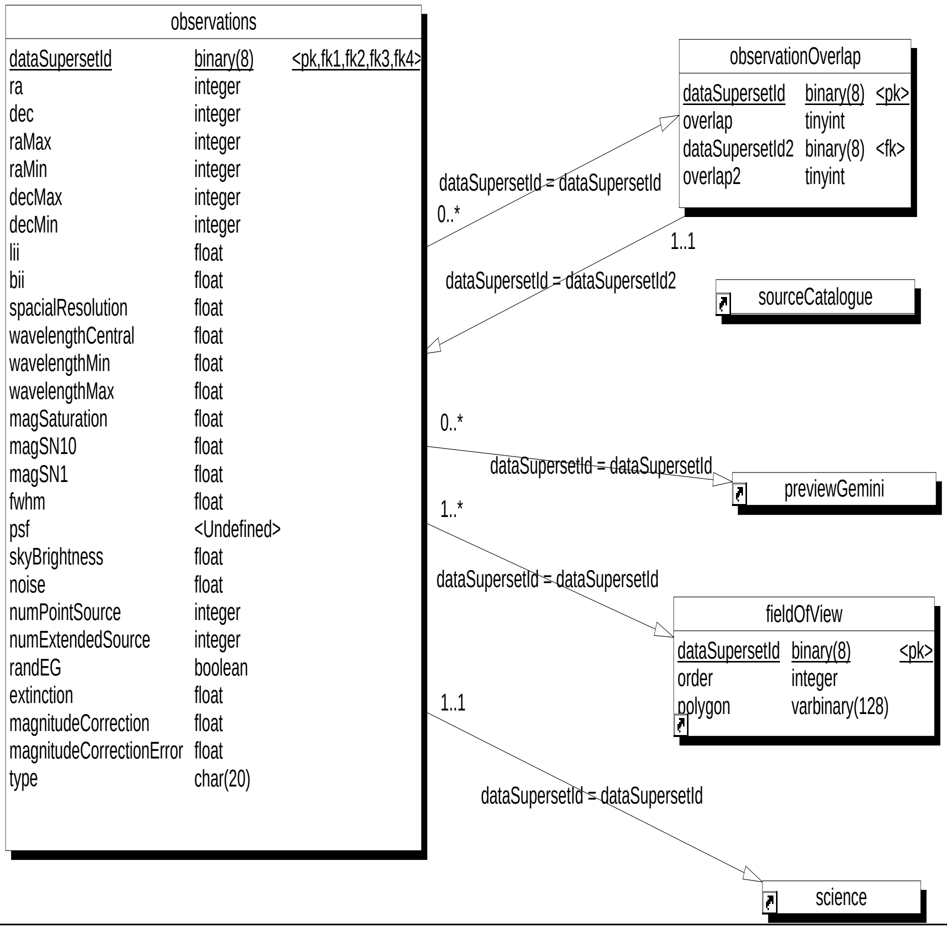


TABLE 1. Sybase Data Types

Data Type	Size	Description
binary(*n*)	*n* bytes	A field containing up to *n* bytes of binary data, where *n* is less than 256 bytes.
char(*n*)	*n* bytes	A field containing up to *n* characters, where *n* is less than 256 bytes. If less than *n* characters are stored in the field, the string is padded to *n* characters with spaces. See also the varchar and text data types.
datetime	8 bytes	A data type to store date and time. Dates between January 1 1753 and December 31 9999 can be stored to an accuracy of 1/300 second.
smalldatetime	4 bytes	A data type to store date and time. Dates between January 1 1990 and June 6 2079 can be stored to an accuracy of 1 minute.
float	8 bytes	Stores floating point numbers to approximately 16 digit accuracy.
integer	4 bytes	Stores integer values between -2^{31} and 2^{31}-1.
tinyint	1 byte	Stores integer values between 0 and 255.

TABLE 1. Sybase Data Types

Data Type	Size	Description
text	variable	This data type stores large (up to 2^{31}-1 characters) character strings. Text data is stored on 2Kilobyte data pages, with up to 1800 characters on each data page. The number of bytes of disk space needed to store a block of text can be calculated from: (numCharacters + 1799)/1800 * 2048, using integer arithmetic, and truncating any fractional numbers. See also the char and varchar data types.
image	variable	The image datatype is used to store larger blocks of binary data on external data pages. An image column can store up to 2,147,483,647 bytes of data on linked lists of data pages separate from other data storage for the table. Image data is stored in the same way as text data.
timestamp	8 bytes	Timestamp values are automatically assigned by the Sybase database server. Timestamp values for a row are updated when the row is inserted or updated. The assigned timestamp values are unique within each database, and each timestamp is guaranteed to be larger than any previously assigned timestamp in the database.
varchar(*n*)	string length + 6 bytes	A variable length field containing up to *n* characters, where *n* is less than 256 bytes. Unlike the char data type, the strings are not padded to *n* characters, however there is a performance cost and an overhead of 6 bytes per value. See also the char and text data types.

ually (e.g. individual, short exposure infrared observations), the associated data superset would replace the original datasets in the GSA science catalogue (the data labels of the individual datasets used to create the data superset would be a property of the associated data superset). In the cases where the members of an associated data-superset have significant individual value, both the original dataset and the associated dataset may be accessible in the GSA science catalogue. The user interface to the catalogues, and the data files delivered to GSA users will have to make the associations clear, to avoid the possibility of GSA users thinking that the associated data superset and the member datasets are independent observations.

Associated data supersets can be heirachical. For example, several infrared observations could be stacked to create an associated data superset which replaces the member datasets in the GSA science catalogues. Several of these stacked associations could then be associated into a mosaic association, where both the stacked associations and the mosaic association are visible in the GSA catalogues.

The same set of datasets may be incorporated into more than one data superset, for example one data superset might co-add a set of images, while another preserves and emphasizes time varying data.

Associated data-supersets may have real, pre-calculated data files associated with them, or they may be virtual entities representing the potential output of data processing recipe instances.

Created:July 12, 2000
Modified:May 21, 2002

Gemini Science Archive

Chapter 2 Top Level Design

This chapter gives a general description of the subsystems of the GSA, concentrating on how the subsystems interact with each other. Each of the subsystems is covered in detail in a separate chapter.

1. The subsystems of the GSA

Figure 4 on page 20 shows the subsystems of the GSA, and how they interact. The operator console is omitted from Figure 4 in order to simplify the diagram. The shaded area in Figure 4 indicates which of the systems are part of the GSA. The subsystems are described in more detail in the Section 1.2 through Section 1.8, and the interactions between the subsystems are described in Section 2. on page 27. The conceptual designs of the subsystems of the GSA are given in Chapter 3 through Chapter 10.

Table 2 on page 21 lists the software requirements from [6] which are met by the GSA design. The columns in Table 2 are:

- *Software Requirement* is the SR number and short description from [6].
- *Level* indicates if the requirement was classified as fundamental, key, or advanced in [6].

  Fundamental — These requirements must be met in order for data to be fully archievable. Data collected before these requirements are met will not be fully archievable.

  Key — These requirements are necessary in order to have a basically functional archive. An archive that meets all of these requirements will be able to ingest data into the archive, allow users to identify data in the archive, and will allow users to retrieve data from the archive.

  Advanced — These are requirements for the GSA to actively manipulate the archived data in order to extract “value added” information.

- *Status* indicates if the capability already exists at the CADC (Exists), is new to the CADC archives (New), or is new to the CADC archives but was demonstrated in a prototype (proto).
- Shared indicates if the implementation of the requirement is inherently specific to the GSA, or is shared with other archives at the CADC.
- *GSA Subsystem* indicates which GSA subsystems address the requirement. The subsystem abbreviations used in the table are:

  UI — User Interface (see Chapter 3).

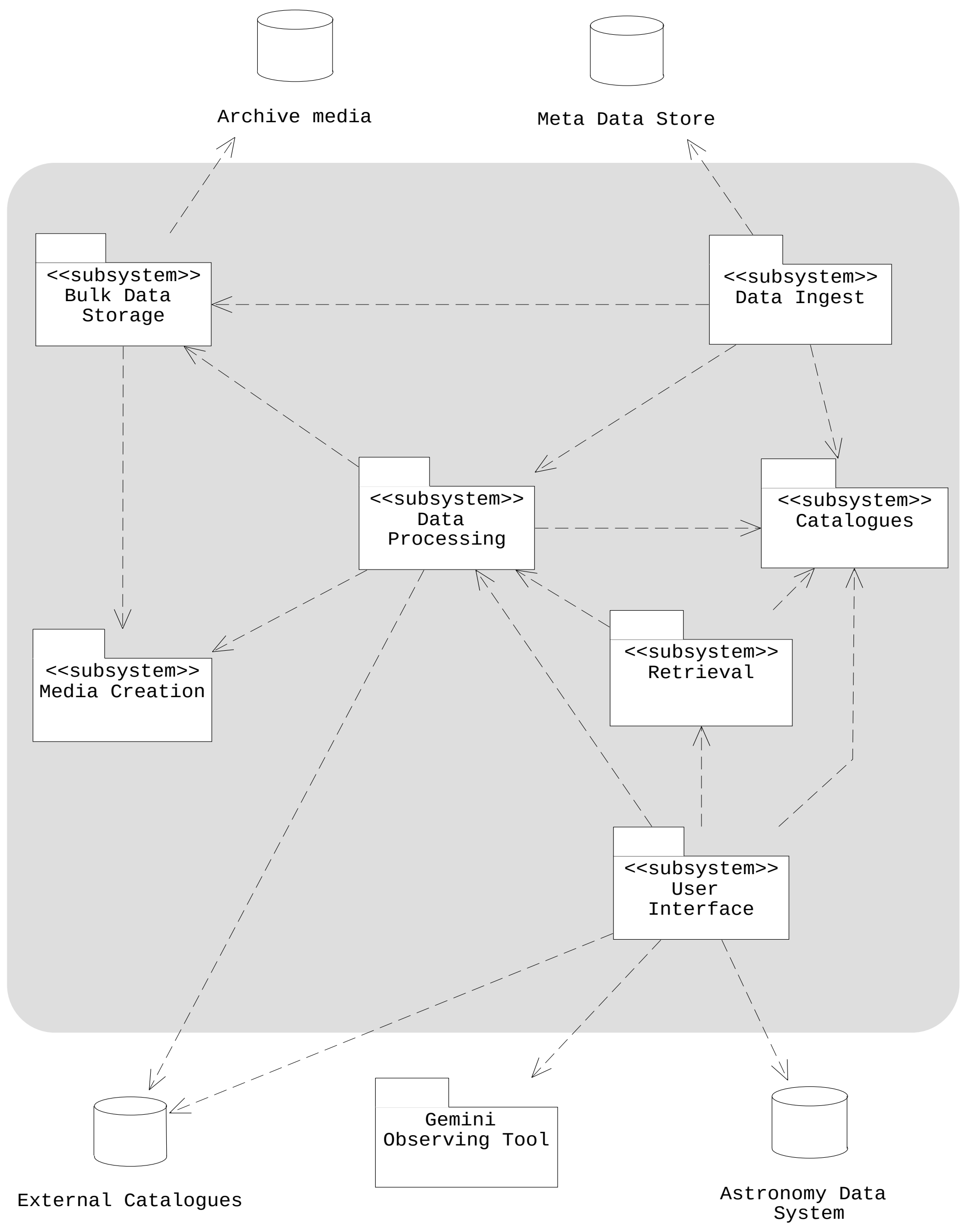
  CAT — Catalogues (see Chapter 4).

  DP — Data Processing (see Chapter 5).

  DI — Data Ingest (see Chapter 6).

  DR — Data Retrieval (see Chapter 7).

FIGURE 4. The subsystems of the GSA


Archive media
Meta Data Store
<<subsystem>> Bulk Data Storage
<<subsystem>> Data Ingest
<<subsystem>> Data Processing
<<subsystem>> Catalogues
<<subsystem>> Media Creation
<<subsystem>> Retrieval
<<subsystem>> User Interface
External Catalogues
Gemini Observing Tool
Astronomy Data System

BDS — Bulk Data Storage (see Chapter 8).

MC — Media Creation (see Chapter 9)

Table 3 on page 24 lists the requirements from [6] which are not implemented by this design, with an explanation of why the requirement was not implemented.

TABLE 2. Software requirements implemented by the GSA design

Software Requirement		Level	Status	Shared	GSA Subsystems
SR1.1	The GSA shall archive all data.	Fundamental	Proto	GSA	CAT, DI, BDS
SR1.2	The GSA shall respect proprietary periods assigned by Gemini.	Key	New	GSA	DI, UI, DP, DR
SR1.3	The GSA shall be able to automatically process data.	Advanced	Proto[a]	Partly shared[b]	DP
SR1.3.1	Data processing shall respect Gemini policy on proprietary periods.	Key	Exists	Partly shared[c]	UI, DP, DR
SR1.3.2	Data processing shall proceed promptly.	Advanced	Proto[a]	Shared	DP, DR, BDS
SR1.4	Preview data should be generated for each dataset.	Advanced	New	GSA	DP
SR1.4.1	Generate previews for associations.	Advanced	New	GSA	DP
SR1.4.2	Remove previews when release date is changed.	Key	New	GSA	CAT, DP, DI
SR1.4.3	Preview should be mirrorable.	Key	Exists	Shared	CAT
SR1.5	Collect information needed to track of publications based on archive data.	Fundamental	New	GSA[d]	DR
SR2.1	The GSA catalogue shall contain all data descriptors.	Key	Proto	GSA	CAT
SR2.2	The catalogue shall include environmental information.	Key	New	GSA	CAT
SR2.3	The catalogue shall include accurate WCS information.	Key	Proto	GSA	CAT, DP
SR2.4	Proposal information should be associated with data supersets.	Key	New	GSA	CAT
SR2.5	The catalogue should include image descriptors derived from the data.	Advanced	New	Partially shared[b,e]	CAT, DP
SR2.5.1	Remove derived image descriptors for proprietary data.	Advanced	New	GSA	CAT, DP, DI
SR2.6	Release date is a data descriptor.	Key	Proto	GSA	CAT, DI
SR2.7	The GSA catalogue should contain associated datasets.	Advanced	New	GSA	CAT, DP
SR2.8	Data quality assessment shall be in the catalogue.	Key	New	GSA	CAT, DI
SR2.9	The science program shall be associated with the catalogue data.	Key	New	GSA	CAT, DI
SR2.10	The electronic observing log shall be associated with the catalogue.	Key	New	GSA	CAT, DI
SR2.11	Proposal id shall be a descriptor.	Key	Proto	GSA	CAT, DI
SR2.12	The GSA catalogue shall be mirrorable.	Key	Exists	GSA[d]	CAT
SR2.13	Calibration data shall be associated with science data.	Key	Exists	Partly shared[f]	DP, DI
SR2.14	The GSA should have an object catalogue.	Advanced	Proto[g]	Partly Shared[b]	CAT

TABLE 2. Software requirements implemented by the GSA design

Software Requirement	Level	Status	Shared	GSA Subsystems
SR2.14.1 Object parameters should be extracted from object catalogues.	Advanced	New	Partly Shared[b]	CAT, DP
SR2.14.2 The GSA object catalogue should be up to date.	Advanced	New	Partly Shared[b]	CAT, DP
SR2.14.3 The object catalogue should include object descriptors derived from data.	Advanced	New	Partly Shared[b]	CAT, DP
SR2.14.4 Remove derived objects for proprietary data.	Advanced	New	Partly Shared[b]	CAT, DP, DI
SR2.15 Publications should be linked to data supersets.	Key	New	GSA[d]	CAT, DI
SR2.16 The GSA shall store Gemini hardware and software versions.	Key	New	GSA	CAT, DI
SR3.1 The GSA shall have an interactive user interface.	Key	Proto	GSA	UI
SR3.2 Select subsets of the GSA catalogue.	Key	Proto	GSA	UI, CAT
SR3.2.1 Select subsets of the GSA catalogue based on coordinate region search.	Key	Proto	GSA	UI, CAT
SR3.2.2 GSA users should be able to select data supersets based on object descriptor.	Advanced	New	GSA	UI, CAT
SR3.2.3 Support cross catalogue searches.	Advanced	New	Shared	UI, CAT
SR3.2.4 The GSA should support a batch mode interface.	Key	New	GSA[d]	UI
SR3.3 View descriptors for a subset of the catalogue.	Key	Proto	GSA	UI, CAT
SR3.3.1 There shall be on-line help describing all catalogue data descriptors.	Key	New	GSA	UI, CAT
SR3.3.2 GSA users can view a list of calibration data supersets.	Key	New	GSA	UI, DP
SR3.3.3 View the list of publications associated with a data superset.	Key	New	GSA[d]	UI, CAT
SR3.3.4 View descriptors of objects in a data superset field of view.	Advanced	New	GSA	UI, CAT
SR3.3.5 View proposal information associated with a data superset.	Key	New	GSA	UI, CAT
SR3.3.6 Sort the displayed subset of the catalogue.	Key	New	GSA	UI
SR3.3.7 Sort the displayed objects.	Key	New	GSA	UI
SR3.3.8 Plot a subset of data supersets in spatial coordinates.	Advanced	New	GSA	UI
SR3.3.9 GSA users shall be able to view the electronic observing log.	Key	New	GSA	UI, CAT
SR3.3.10 View environmental data.	Key	New	GSA	UI, CAT
SR3.3.11 Display science programs.	Key	New	GSA	UI, CAT
SR3.3.12 Display the members of association data supersets.	Advanced	New	GSA	UI, CAT
SR3.3.13 Save a set of data supersets to a file.	Key	New	GSA	UI
SR3.3.14 Load a set of data supersets from a file.	Key	New	GSA	UI

TABLE 2. Software requirements implemented by the GSA design

Software Requirement		Level	Status	Shared	GSA Subsystems
SR3.4	GSA users shall be able to select data for retrieval.	Key	Proto	Partly shared[h]	UI, DR
SR3.4.1	Users shall retrieve data via Internet, EXABYTE, or CD-ROM.	Key	Exists	Shared	UI, DR, MC
SR3.4.2	Only authorized users shall be able to retrieve proprietary data.	Key	New	Shared	UI, DR
SR3.4.3	GSA users should be able to choose to have data processed.	Advanced	New	Partially Shared[b]	UI, DP, DR
SR3.4.4	Data processing shall not use proprietary data as input.	Advanced	New	GSA	UI, DP, DR
SR3.4.5	Processing recipes shall be documented.	Advanced	New	GSA	DP, DR
SR3.4.6	Authorized Gemini staff can retrieve proprietary data.	Key	New	Shared	UI, DP, DR
SR3.4.7	GSA users can retrieve calibration data.	Key	New	GSA	UI, DP, DR
SR3.4.8	Retrieve environmental data.	Key	New	GSA	UI, CAT, DR
SR3.4.9	It shall be possible to retrieve science programs.	Key	New	GSA	UI, CAT, DR
SR3.4.10	It shall be possible to retrieve proposal information.	Key	New	GSA	UI, CAT, DR
SR3.4.11	It shall be possible to retrieve the electronic observing log.	Key	New	GSA	UI, CAT, DR
SR3.4.12	It should be possible for users to monitor the progress of data requests.	Key	New	Shared	UI, DP, DR
SR3.4.13	Batch mode requests.	Key	New	GSA[d]	UI, DR
SR3.4.14	Document user data requests.	Key	New	Shared	DR
SR3.4.15	Retrieve associated data supersets.	Advanced	New	GSA	DR
SR3.4.16	Provide data processing recipes for associations.	Advanced	New	GSA	DR
SR3.5	A general description of the content of the GSA shall be available.	Key	New	GSA	UI
SR3.5.1	A description of the Gemini instruments shall be available.	Key	New	GSA	UI
SR3.5.2	A description of the Gemini facilities shall be available.	Key	New	GSA	UI
SR3.6	Proposal information should be searchable.	Key	New	GSA	UI, CAT
SR3.6.1	Find all data supersets associated with proposals.	Key	New	GSA	UI, CAT
SR3.7	Preview display for each science data superset.	Key	New	GSA	UI, CAT
SR4.1	Maximum and average requirements for catalogue ingest.	Fundamental	New	GSA	CAT, DI
SR4.2	Maximum and average requirements for raw data ingest.	Fundamental	New	GSA	DI, BDS
SR4.3	Maximum and average requirements for data retrieval by users.	Key	Exists	Shared	DP, DR, BDS
SR4.4	Preview should be display promptly.	Key	Exists	Shared	UI, CAT
SR4.5	Access to the GSA catalogue must be interactive.	Key	Proto	GSA	UI, CAT

TABLE 2. Software requirements implemented by the GSA design

Software Requirement		Level	Status	Shared	GSA Subsystems
SR4.6	Data for Internet retrieval should be available promptly.	Key	Exists	Shared	DP, DR, BDS
SR5.3	Data shall be electronically secure.	Fundamental	Exists	Shared	BDS
SR5.7	GSA operations staff shall incorporate evolving technologies into the GSA.	Key	New	Shared	BDS, MC

a. The CADC has existing archive specific automatic data processing. Data processing for GSA will be integrated into a new, shared data processing infrastructure which is currently in the prototype stage.

b. The basic data processing infrastructure will be shared, some data processing recipes will be GSA specific, some data processing recipes will be shared.

c. Existing archives at the CADC contain proprietary data, and some of the checking and enforcement will be shared. The mechanism for determining proprietary status is GSA specific.

d. Although this capability could be shared, it is not a requirement for any other archive at the CADC.

e. The catalogue table structure needed to store the information will be shared.

f. The data processing database tables used to store this information will be shared. The mechanisms to association science data with calibration data will be GSA specific.

g. CADC has developed a prototype of the table structure.

h. The retrieval infrastructure will be shared. Much of the data superset selection mechanism will be GSA specific.

TABLE 3. Software requirements not implemented by the GSA design

Software Requirement		Explanation
SR5.1	Participate in catalogue unification efforts.	These are operations issues which do not directly affect the design of the GSA.
SR5.2	Data shall be physically secure.	
SR5.4	GSA staff shall respect proprietary periods.	
SR5.5	GSA operations staff shall support GSA users.	
SR5.6	GSA operations staff shall support ongoing development of Gemini.	
SR5.8	The GSA operations staff should add new features to the GSA.	
SR6.1	GSA development cost.	This issues are dealt with in the GSA Phase II proposal, and do not directly affect the design.
SR6.2	GSA operating cost.	
SR7.*	Requirements on Gemini.	These requirements must be met by Gemini, and do not directly influence the design.
SR8.*	External Requirements.	These requirements are external to the GSA and Gemini, and do not directly influence the design.

1.1 User Interface

GSA users interact with the GSA through the GSA user interface. The user interface allows users to query the GSA catalogues, and submit data to be retrieved from the GSA.

The conceptual design of the user interface is described in Chapter 3.

1.2 Catalogue Subsystem

The GSA catalogues are subdivided into the following subsets:

- A catalogue of all data supersets.
- A catalogue of science data supersets.
- A catalogue of sources extracted from data.
- A catalogue of objects (this includes both objects extracted from NED/SIMBAD, and objects extracted from data). This is similar to the source catalogue, except that all of the sources which describe the same physical object are merged.
- Preview data.

All catalogues stored in a SYBASE database. The conceptual design of the Catalogue subsystem is described in Chapter 5.

1.3 Data Processing Subsystem

The Data Processing subsystem is responsible for doing all of the data processing needed by the GSA. This includes:

- Data processing for user requests, including proprietary status checking.
- Generation of descriptors derived from data.
- Generation of source and object catalogues derived from data.
- Generation of object catalogues from the SIMBAD and NED databases.
- Generation of previews.

Data processing is triggered by the following events:

- Discovery of new datasets in the meta-data store.
- A recipe instance becomes runable when all of its inputs are public.
- Arrival of a user data request.
- Other data processing steps.

The Data Processing system has database tables which record the state of all data processing. These tables will allow data processing to be re-done as necessary when data processing trees are extended and updated.

The conceptual design of the Data Processing system is described in Chapter 5.

1.4 Data Ingest Subsystem

The Data Ingest subsystem is responsible for receiving information from the Gemini telescopes. Specifically, The Data Ingest subsystem performs the following tasks:

- As Gemini adds new datasets to the Gemini meta-data stores, the datasets are incorporated into the GSA catalogues.
- If Gemini makes changes to the Gemini meta-data stores, the changes are incorporated into the GSA catalogues.

- When new datasets are added to the meta-data store or when existing datasets are changed the Data Ingest subsystem starts the appropriate Data Processing Discovery Agents, as described in Chapter 5, section 4.3 on page 89.
- As bulk pixel data is received from Gemini on archive media, the Data Ingest subsystem verifies that the corresponding datasets exist in the GSA catalogues.

The conceptual design of the Data Ingest subsystem is described in Chapter 6.

1.5 Data Retrieval Subsystem

The Data Retrieval subsystem manages requests for data. This subsystem performs the following tasks:

- Uses the bulk data storage subsystem to retrieve data from the archive.
- Uses the data processing system to do any required data processing, which would include packaging the results for each request.
- Manages an FTP directory of user requests.
- Notifies users when requests are complete.
- Enforces proprietary data protection.
- Records request activity in a database.

The conceptual design of the Data Retrieval subsystem is described in Chapter 7.

1.6 Bulk Data Storage Subsystem

The Bulk Data Storage subsystem stores and tracks the files containing the pixel data for the GSA, and provides access to the data for the other systems which need it. This subsystem will also be responsible for managing the migration of Gemini data to new media as it becomes necessary.

The conceptual design of the Bulk Data Storage subsystem is described in Chapter 8.

1.7 Media Creation Subsystem

The Media Creation subsystem is responsible for creating removable media. This will be done for some user requests, and it will be done when it becomes necessary to migrate archive storage media to new technologies, and it may be done to keep a permanent copy of processing results.

The conceptual design of Media Creation subsystem is described in Chapter 9.

1.8 Operator Console Subsystem

The Operator Console subsystem informs the GSA operators of the status of the system, and allows the operators to control the system. The Operator console also provides the interface for any manually controlled GSA operations (e.g. media writing, archive media ingest).

The conceptual design of the Operator Console subsystem is described in Chapter 10.

2. Subsystem Interactions

This sections describes the interactions between the GSA subsystems. In addition to the interactions listed here, the GSA console will interact with most of the subsystem, collecting status information for display, and forwarding commands to appropriate subsystems.

2.1 User Interface to Catalogue

The User Interface will submit queries to the catalogue system. Query results will be returned to the User Interface for display.

The User Interface will send requests for preview data. The result of the request will be a data stream containing a FITS file to be displayed to the user.

2.2 User Interface to Data Retrieval Subsystem

The User Interface sends a list of data supersets with indication of types of data to retrieve, an optional list of data processing to be done on data supersets, and the delivery media type. The request may also include authentication information for retrieving proprietary data.

The User Interface monitors progress of data requests submitted by a user, as reported in the Data Retrieval subsystem database tables.

2.3 User Interface to Data Processing Subsystem

The User Interface uses the Data Processing subsystem, and in particular the data processing catalogues to get information about data supersets. This information includes:

- The list of data processing recipe instances available for data supersets.
- Detailed information about the recipe instances.
- The list of calibration files appropriate for a dataset.
- The member datasets for associated data supersets.

2.4 User Interface to ADS

The User Interface subsystem will use NASA's ADS system to get detailed information about publications.

2.5 User Interface to Gemini Observing Tool

The GSA user interfaces will use the Gemini Observing tool to display observing programs to GSA users. This should also allow observing programs from the archive to be modified and re-introduced into the observing process.

2.6 User Interface to External Catalogues

The user interface will allow links to external catalogue web interfaces from the object detail web page described in Chapter 3. This will allow linking object information extracted from external catalogues back to the source of the information.

2.7 Data Ingest Subsystem to Catalogue Subsystem

The Data Ingest subsystem will add datasets to the database tables in the catalogue subsystem as new datasets are detected, and will update the tables in the Catalogue subsystem when the meta-data for existing datasets is modified.

2.8 Data Ingest Subsystem to Bulk Data Storage Subsystem

The Data Ingest subsystem will use the Bulk Data Storage subsystem to keep track of the locations of files on archive media. This will be used when cross-checking the datasets in the catalogue against the datasets available on archive media.

The Data Ingest subsystem will also be able to extract meta-data from the files stored in the Bulk Data Storage subsystem, although this is intended as a backup to the primary source of meta-data.

2.9 Data Ingest Subsystem to Data Processing Subsystem

The Data Ingest subsystem will queue Data Processing Discovery Agents (see Chapter 5, section 4.3 on page 89) for each new or modified dataset.

2.10 Data Ingest Subsystem to Meta-Data Store

The Data Ingest subsystem will examine the meta-data stores described in [3] to determine if any datasets have been added or modified.

2.11 Data Retrieval Subsystem to Data Processing Subsystem

The Data Retrieval subsystem will use the Data Processing subsystem for the following tasks:

- To do any data processing necessary to for a user data request.
- As a source of information to identify the appropriate calibration data supersets associated with data supersets in a user data request.
- To do the packaging associated with processing a request.
- To get release dates for data processing needed for a user data request.

2.12 Data Retrieval Subsystem to Media Creation Subsystem

The Data Retrieval subsystem uses the Media Creation subsystem to create media to be delivered to users.

2.13 Data Retrieval Subsystem to Catalogue Subsystem

The Data Retrieval subsystem uses the Catalogue subsystem to get release dates associated with data supersets.

2.14 Data Processing Subsystem to Catalogue Subsystem

Recipe instances executed by the Data Processing subsystem will insert various types of data into the catalogues. These types of data will include:

- New associated data supersets detected from the data.

- Data superset descriptors derived from the data.
- Source attributes derived from the data.
- Object attributes derived from the data.

2.15 Data Processing Subsystem to Media Creation Subsystem

The Data Processing subsystem will use the Media Creation subsystem to write data to removable media for user requests, and/or to become a part of the archive data store.

2.16 Data Processing Subsystem to Bulk Data Storage Subsystem

The Data Processing subsystem will retrieve data from the Bulk Data Storage subsystem when it is needed as input to recipe instances.

2.17 Data Processing Subsystem to External Catalogues

The recipes running in the Data Processing subsystem will access external catalogues to extract information about objects in the “field of view” of Gemini observations. This information will be inserted into the Gemini object catalogue tables.

2.18 Bulk Data Storage Subsystem to Archive Media

The Bulk Data storage subsystem will read archive media received from Gemini, and copy the data to on-line media. The Bulk Data storage subsystem will also be able to access archive media in a jukebox as an alternative source of data in the event of a failure of an on-line storage system.

2.19 Bulk Data Storage Subsystem to Media Creation Subsystem

The Bulk Data Storage Subsystem will use the Media Creation Subsystem to create new archive media when it becomes necessary to migrate to new media types.

3. User interactions

This section describes the interactions between GSA users and the components of the GSA.

3.1 User interaction with User Interface.

A user will get to the GSA top level web page through the archive centre web page. The GSA top level web page will lead to the GSA web pages described in Chapter 3.

The user may also interact with the GSA though the e-mail batch interface described in Chapter 3.

3.2 User interaction with Data Retrieval subsystem

When a user’s data request is complete, the Data Retrieval subsystem will send e-mail to the user notify him/her that the data is ready to be retrieve. The message will include instructions describing how the data can be retrieved from the archive centre using FTP.

4. Interaction with Gemini

All interaction between the GSA and Gemini systems will be through the Data Ingest subsystem. The details of the interaction are described in [3].

Created:July 17, 2000
Modified:May 21, 2002

Gemini Science Archive

Chapter 3 *User Interface*

1. Introduction

This chapter describes the conceptual design of the GSA user interface. The purpose of the user interface is to:

- Allow GSA users to view the data stored in the GSA catalogues, and to do cross catalogue queries involving data available from other archives
- Allow users to select Gemini data to be retrieved, including:
  - Raw data files.
  - Calibration data files.
  - Meta-data such as the observing log and observing programs associated with datasets.
  - Processed data, with the type of processing done selected by the user.
- Select delivery media for data requests.
- Provide preview display for datasets in the GSA archive.
- Provide links to documentation of the Gemini telescopes and instruments.

The primary interface to the GSA will be a GUI, however a batch interface which provides most of the same functionality will also be available.

1.1 Requirements on the GSA catalogue subsystem

Table 4 on page 31 lists the software requirements which apply to the GSA User Interface. The table includes a brief description of how the requirement impacts the user interface. Each description also includes the keywords “GUI”, “batch”, or “both” to indicate if the comment applies to the GUI user interface, the batch user interface, or both user interfaces. See [6] for a complete description of these requirements.

TABLE 4. User interface requirements

SR1.2 The GSA shall respect proprietary periods assigned by Gemini. GUI: Display release dates to the user, and warn the user when he/she attempts to access data which he/she is not authorized to access. This is informative only, enforcement will be the responsibility of the Data Retrieval subsystem.
SR1.3.1 Data processing shall respect Gemini policy on proprietary periods. GUI: Indicate which recipes cannot be executed because the user has insufficient authorization to access required input data. This is informative only, enforcement will be the responsibility of the Data Retrieval subsystem.
SR3.1 The GSA shall have an interactive user interface. GUI: The primary user interface shall be interactive, using the Internet for communications.

TABLE 4. User interface requirements

SR3.2 Select subsets of the GSA catalogue. both: Supply the interface for this query.
SR3.2.1 Select subsets of the GSA catalogue based on coordinate region search. both: Supply the interface for this query.
SR3.2.2 GSA users should be able to select data supersets based on object descriptor. both: Supply the interface for this query.
SR3.2.3 Support cross catalogue searches. both: Supply the interface for these queries.
SR3.2.4 The GSA should support a batch mode interface. batch: The above interfaces should be available from a batch mode interface which does not rely on a fast or reliable internet link for interaction between the GSA and the user. The batch mode interface may use the internet for non-interactive communication between the GSA and the user.
SR3.3 View descriptors for a subset of the catalogue. GUI: Provide an interactive display. Batch: Allow a user to retrieve descriptor values to a file in their home computer.
SR3.3.1 There shall be on-line help describing all catalogue data descriptors. GUI: Provide interface to cause descriptor descriptions to be displayed to the user. Batch: There will be a down-loadable source of GSA documentation which includes documentation of data descriptors.
SR3.3.2 GSA users can view a list of calibration data supersets. GUI: Provide interface to display calibration datasets for the user, for any given science dataset. Batch: Provide an option to include information associating calibration data with science data supersets in query results (see SR3.3).
SR3.3.3 View the list of publications associated with a data superset. GUI: Provide an interface which will allow a user to query the GSA catalogues to get a list of publications associated with a data superset, and to display the list of publications. Batch: have an option to retrieve the publications associated with the data supersets in a query result (see SR3.3).
SR3.3.4 View descriptors of objects in a data superset field of view. GUI: Provide an interface which will display list of objects in the field of view, with the object descriptors. Batch: Provide a batch query interface to the object table.
SR3.3.5 View proposal information associated with a data superset. GUI: Provide an interface which will allow a user to query the database to get and display public proposal information associated with any data superset.
SR3.3.6 Sort the displayed subset of the catalogue. GUI: Allow a user to sort the displayed data supersets (see SR3.3).
SR3.3.7 Sort the displayed objects. GUI: Allow a user to sort the displayed objects (see SR3.3.4).
SR3.3.8 Plot a subset of data supersets in spatial coordinates. If this is done, the user interface should do it.
SR3.3.9 GSA users shall be able to view the electronic observing log. GUI: Allow a user to query the database to get and display the electronic observing log entries associated with any data superset. Batch: This information is available through a batch data request.

TABLE 4. User interface requirements

SR3.3.10 View environmental data. GUI: allow a user to query the database to get and display environmental data associated with any data superset. Batch: This information is available through a batch data retrieval request.
SR3.3.11 Display science programs. GUI: Provide interface elements which allow a user to query the database to get and display observing programs associated with any data superset. If more than one observing program is associated with a data superset, the UI should display a list of observing programs, and allow the user to pick the one to be displayed. Batch: This information is available through a batch data retrieval request.
SR3.3.12 Display the members of association data supersets. GUI: Allow a user to query the database to get and display members of associated data supersets. Batch: Have an option which will cause the membership information to be included with the query result (see SR3.3).
SR3.3.13 Save a set of data supersets to a file. GUI: Allow the currently displayed set of data supersets to be saved to a file. This should not require any interaction with any other GSA subsystem.
SR3.3.14 Load a set of data supersets from a file. GUI: Allow a list of data supersets to be read from a file. The list of data supersets will become the currently displayed catalogue subset (see SR3.3). The data superset descriptors displayed to the user will be read from the GSA catalogue, not from the file.
SR3.4 GSA users shall be able to select data for retrieval. GUI: Provide interface elements to pick individual data supersets, or all data supersets in the currently displayed subset of the GSA catalogue, and to submit a data request for processing. Batch: Provide a way to submit a list of data supersets to be retrieved.
SR3.4.1 Users shall retrieve data via Internet, EXABYTE, or CD-ROM. both: Allow users to choose the delivery media for a data request before it is submitted.
SR3.4.2 Only authorized users shall be able to retrieve proprietary data. See SR1.2.
SR3.4.3 GSA users should be able to choose to have data processed. both: Allow user to choose to data processed.
SR3.4.4 Data processing shall not use proprietary data as input. See SR1.3.1.
SR3.4.6 Authorized Gemini staff can retrieve proprietary data. GUI: Provide a way for authorized Gemini staff to identify themselves, and permit them to submit requests for proprietary data.
SR3.4.7 GSA users can retrieve calibration data. both: Provide a way for users to indicate that all calibration data should be retrieved.
SR3.4.8 Retrieve environmental data. both: Provide an option to retrieve environmental data with requested data.
SR3.4.9 It shall be possible to retrieve science programs. both: Provide an option to retrieve observing program data with requested data.
SR3.4.10 It shall be possible to retrieve proposal information. both: Provide an option to retrieve proposal data with requested data.
SR3.4.11 It shall be possible to retrieve the electronic observing log. both: Provide an option to retrieve electronic observing log data with requested data.

TABLE 4. User interface requirements

SR3.4.12 It should be possible for users to monitor the progress of data requests. GUI: Provide an interface which monitors the progress of a requests through the GSA retrieval and processing system.
SR3.4.13 Batch mode requests. Batch: There will be a batch interface to allow data requests to be submitted.
SR3.5 A general description of the content of the GSA shall be available. GUI: Provide a link to a web page describing the GSA. Batch: Provide downloadable help files.
SR3.5.1 A description of the Gemini instruments shall be available. GUI: Provide a link to Gemini's web pages. Batch: Provide pointers to Gemini's downloadable help, if available.
SR3.5.2 A description of the Gemini facilities shall be available. GUI: Provide a link to Gemini's web pages. Batch: Provide pointers to Gemini's downloadable help, if available.
SR3.6 Proposal information should be searchable. both: Provide a proposal search interface.
SR3.6.1 Find all data supersets associated with proposals. GUI: provide a link from the proposal search interface (see SR3.6) to automatically select the subset of the GSA catalogue associated with the proposal. The subset of the catalogue is displayed in the interface described in SR3.3.
SR3.7 Preview display for each science data superset. GUI: Provide elements on interface described in SR3.3 to cause the previews for a data superset to be displayed. Provide a preview display interface.
SR4.4 Preview should be display promptly.
SR4.5 Access to the GSA catalogue must be interactive.

2. User interface elements

2.1 GUI

The Graphical User Interface to the GSA will consist of a set of HTML web pages and CGI scripts, which will be accessed from the CADC's archive page (see [16]). The top level access to the GSA will be through the opening screen, described in Section 2.1.1 on page 37, and all other screens will be accessed via the opening screen.

Access to the database tables will be through CGI scripts and the WDBI Web to Database interface (see [17]).

The relationships between the GUI web pages are shown in Figure 5 on page 35. The following sections describe the individual screens in the GUI. The relationships between the web pages, the other subsystems of the GSA, and components external to the GSA are shown in Figure 6 on page 36. (The user interface web pages are in the shaded areas.)

FIGURE 5. GUI web page relationships

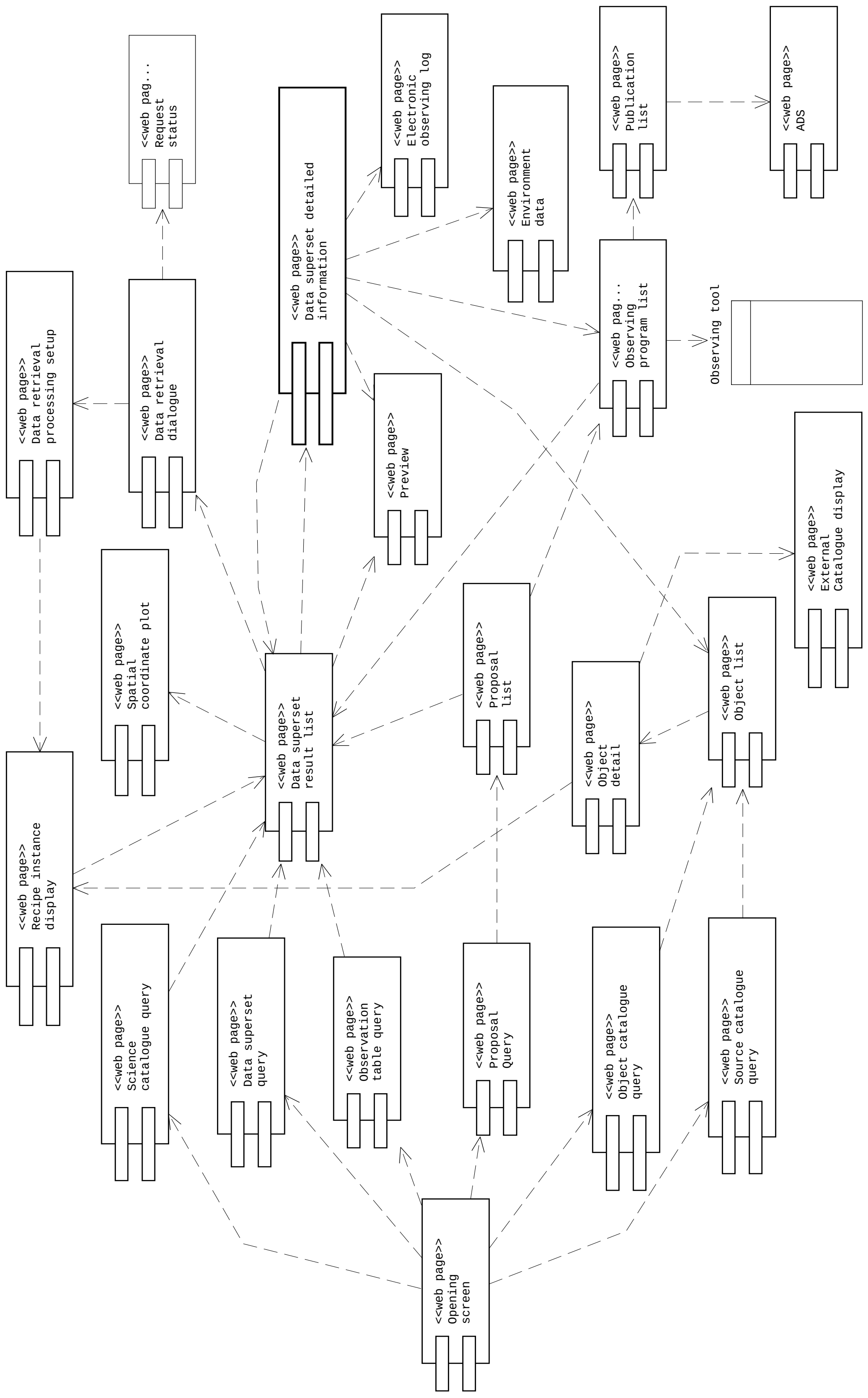
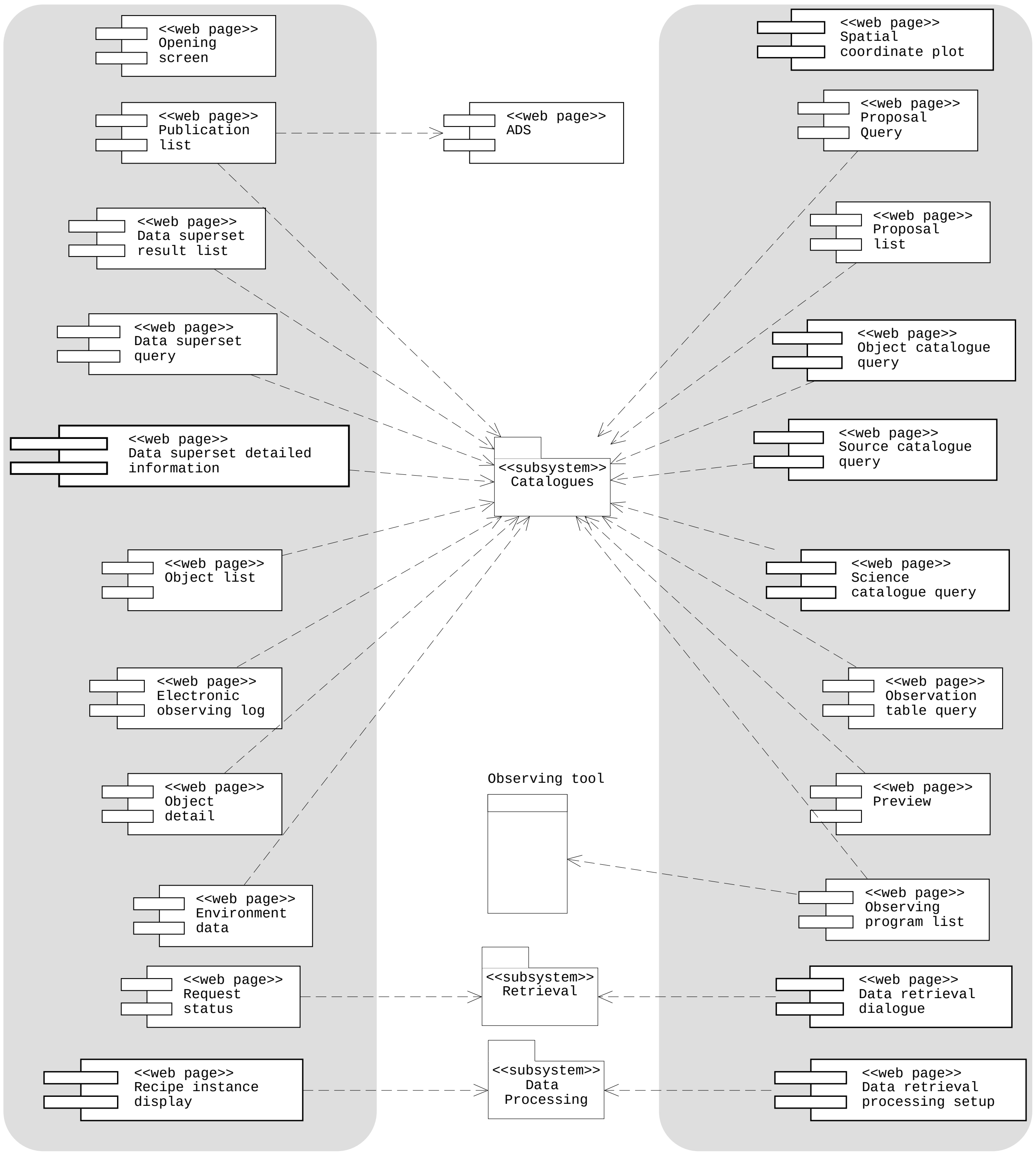


FIGURE 6. Web page relationships to the other components of the GSA


<<web page>> Opening screen
<<web page>> Publication list
<<web page>> ADS
<<web page>> Data superset result list
<<web page>> Data superset query
<<web page>> Data superset detailed information
<<subsystem>> Catalogues
<<web page>> Object list
<<web page>> Electronic observing log
<<web page>> Object detail
<<web page>> Environment data
<<web page>> Request status
<<web page>> Recipe instance display
<<web page>> Spatial coordinate plot
<<web page>> Proposal Query
<<web page>> Proposal list
<<web page>> Object catalogue query
<<web page>> Source catalogue query
<<web page>> Science catalogue query
<<web page>> Observation table query
<<web page>> Preview
Observing tool
<<web page>> Observing program list
<<subsystem>> Retrieval
<<web page>> Data retrieval dialogue
<<subsystem>> Data Processing
<<web page>> Data retrieval processing setup

2.1.1 Opening screen

The GSA opening screen will contain a general description of the content of the GSA archive and the Gemini telescopes and instruments. Detailed descriptions of the telescopes and instruments will be available through links to the Gemini Telescopes web pages.

The opening pages will contain brief descriptions of the catalogue access web pages, with links to detailed descriptions of the pages and to the pages themselves. The catalogue access web pages are:

- The science catalogue query page described in Section 2.1.2 on page 37.
- The dataset query page described in Section 2.1.4 on page 38.
- The object catalogue query page described in Section 2.1.6 on page 39.
- The source catalogue query page described in Section 2.1.9 on page 40.
- The observation query page described in Section 2.1.19 on page 42.
- The proposal search page described in Section 2.1.20 on page 42.

2.1.2 Science catalogue query web page

The science catalogue contains a list of all scientifically interesting Gemini data supersets. The content of the science catalogue is described in Chapter 4, section 4.2 on page 71. This page provides a form which allows the science catalogue to be queried. The form will include:

- A field for every attribute stored in the science table. The form will allow GSA users to qualify any number of fields.
- Fields for position (ra and dec), and a search radius. This will be used to select only those data supersets where the field of view overlaps the specified search region.
- Controls to allow a user to select the attributes to be displayed in the result list.
- Controls to allow a user to set an upper limit on the number of results to return.
- Controls to allow users to submit the query to the archive catalogues, and to view the results in the dataset result list page described in Section 2.1.3 on page 37.
- Controls to allow a list of datasets to be read from a file on the user's local machine. Only datasets contained in the list of datasets will be displayed in the query.
- Each of the fields in the form will have on-line help describing the content of the field.

On-line help will be provided which describes the content of the science table, and in particular which data supersets are not included in the science table.

2.1.3 Data superset result list web page

This page is used to display attributes for a set of datasets in tabular form, with each dataset displayed in a row in the table, and each attribute displayed in a column in the table. The attributes will be extracted from the science or dataset tables described in (Chapter 4, section 4.1 on page 70 or Chapter 4, section 4.2 on page 71).

Each data superset in the table has the following information:

- A link to the detailed data superset information page described in Section 2.1.5 on page 38.
- A control to allow individual datasets to be selected for retrieval. A visual cue will allow the users to identify proprietary data (e.g. colour or icon type).
- A control to allow each dataset's previews to be displayed.

In addition to the controls for each dataset, there will be the following general controls on the page:

- Links to documentation of the attributes in each of the attribute column headers.
- Controls to sort the list of datasets by the values in any displayed attribute column.
- Controls to allow the user to modify the list of displayed attributes.
- Controls to allow the table of datasets and attribute values to be saved to a file on the user's computer.
- Controls to allow all displayed datasets to be selected for retrieval.
- Controls to submit the list of selected datasets for retrieval, using the dialogue described in Section 2.1.16 on page 41.
- A link to a plot of the data supersets in spatial coordinates as described in Section 2.1.24 on page 43.

If an attempt is made to select proprietary data for retrieval, the user will be prompted with a dialogue which allows them choose from the following actions:

- Ignore the proprietary status and select this dataset for retrieval. (The user will be prompted for authorization in the data retrieval dialogue described in Section 2.1.16 on page 41.)
- Ignore the proprietary status for all datasets and select them all for retrieval.
- Do not request this proprietary dataset.
- Do not request any proprietary datasets.

The proprietary data dialogue will be repeated for every proprietary dataset, or until the user selects one of the two "all dataset" options.

2.1.4 Data superset query page

This page is similar to the science catalogue query web page described in Section 2.1.2 on page 37, except that it is not limited to science datasets, and the complete list of attributes may be qualified and retrieved, not just those that are deemed to be scientifically interesting. This page will be much more unwieldy than the science catalogue query web page, and is only expected to be used for special purpose queries. Given the structure of the tables, it is also expected that queries will be slower than queries in the science catalogue query web page.

The results from a query will be displayed in a data superset result list web page, as described in Section 2.1.3 on page 37.

2.1.5 Data superset detailed information

This page is used to display detailed information about a single data superset. All available attributes describing the data superset will be displayed in this page. If necessary the page will be divided into sections based on the category stored in the attribute information table (see Chapter 4, section 3. on page 51).

The page will also contain links to the following information:

- On-line help describing each attribute.
- The recommended calibration data supersets for the current dataset. The calibration datasets will be displayed using the data superset result display described in Section 2.1.3 on page 37.

- A list of objects in the field of view. The objects will be displayed using the object catalogue list web page described in Section 2.1.7 on page 39.
- A list of sources in the field of view. The sources will be displayed using the source catalogue list web page described in Section 2.1.10 on page 40.
- The observing program information associated with data superset. The observing program(s) will be displayed using observing program list display described in Section 2.1.13 on page 40.
- The electronic observing log information entered during the period(s) of data collection. The electronic observing log will be displayed in the web page described in Section 2.1.14 on page 40.
- The environmental data collected during the period(s) of data collection. The environmental data will be displayed in the web page described in Section 2.1.15 on page 41.
- The members of an associated data superset. The members will be displayed using the data superset result list display described in Section 2.1.3 on page 37.
- Any available preview images. The previews will be displayed as described in Section 2.1.22 on page 43.
- A list of all datasets from the observation catalogue (see Chapter 4, section 3. on page 51) which have a field of view which overlaps this data superset's field of view. The list of data-sets will be displayed using the data superset result list display described in Section 2.1.3 on page 37.

In addition to the informational links, there will be a control which allows the data superset to be selected for retrieval.

2.1.6 Object catalogue query page

The object catalogue query page is a form which allows queries of the object catalogue tables described in Section 4.5 on page 73. This form will be configurable, with a default set of fields for qualifying object attributes. The set of attributes in the form can be modified using a pick list of all available attributes. GSA users can qualify values or ranges of values for each attribute on the form. Links to on-line help describing the attributes will be provided from both the fields in the form, and the pick list. Controls will be provided to set a limit on the number of returned objects, and to submit the query to the archive. The result of the query will be display in the object catalogue list web page described in Section 2.1.7 on page 39. The properties displayed on the query form will be those initially displayed in the object catalogue list web page.

2.1.7 Object catalogue list web page

This web page displays a tabular list of objects from the object catalogue. Each row in the table is an object from the object catalogue, and the columns are properties of the objects. There will be controls on the page to:

- Allow the displayed objects to be sorted by the value of a user selected attribute.
- Modify the list of properties displayed for the objects.

There will be links to on-line help pages for each property in the table.

In addition, to the controls described above, there will be links associated with each object. These links will:

- Link an object to all data supersets which contain the object. The data supersets will be dis-played using the data superset result list described in Section 2.1.3 on page 37.

- Link an object to the object detail page described in Section 2.1.8 on page 40.
- Link to the data processing recipe instance that produce each property value. The recipe instances will be displayed using the recipe instance display page described in Section 2.1.12 on page 40.

2.1.8 Object detail page

This page displays all known details about an object stored in the object catalogues. By default, only rank 1 property values will be displayed (rank 1 property values are the are the most reliable values for the property). There will be controls to allow the lower ranked values for the properties to be displayed. There will be links on each property to display an on-line help page describing the property. There will be links on each property value to display the data processing recipe instance that produced the property value, using the recipe instance display page described in Section 2.1.12 on page 40. There will be links to the web interfaces for external object catalogues which will display the information available for the current object.

2.1.9 Source catalogue query page

Identical to object catalogue query page (Section 2.1.6 on page 39), except in content?

2.1.10 Source catalogue list

Identical to object catalogue list (Section 2.1.7 on page 39), except in content?

2.1.11 Source detail page

Identical to object detail page described in Section 2.1.8 on page 40, except in content?

2.1.12 Recipe instance display page

This page is used to display a recipe instance from the data processing catalogues described in Chapter 5, section 3. on page 80. The page will include a link to the on-line documentation for the recipe, and a list of inputs to the recipe (both dataset and recipe instance). Inputs which are other recipe instances will be links to the recipe instance display pages for those recipe instances. There will also be a link which displays all datasets used as input to the recipe. The datasets will be displayed using the data superset result list described in Chapter 3, section 2.1.3 on page 37.

2.1.13 Observing program list page

This page displays a list of Gemini observing programs, and the properties extracted from the observing programs. The page has links to on-line help which describe each of the properties of the observing programs.

Each observing program has the following links:

- A link to all data supersets associated with the observing program. The data supersets are displayed with the data superset result list page described in Section 2.1.3 on page 37.
- A link to a list of publications associated with the observing program. The publications are displayed with the publication display described in Section 2.1.23 on page 43
- A link to display the observing program using the Gemini observing tool.

2.1.14 Electronic observing log display

The electronic observing log display page shows a list of electronic observing log entries. The display will include the date and time the comment was entered, the identity of the person who entered the comment, and the comments entered.

2.1.15 Environmental data display

The environmental data display page shows a table of time stamped environmental data. A preselected subset of the available parameters will be shown by default. A pick list will allow users to modify the list of displayed parameters. There will be links from both the display and from the pick list to on-line help for the parameters.

2.1.16 Data retrieval dialogue

When a user is ready to submit a list of data supersets for retrieval, this dialogue will allow the user to choose the type of data to be retrieved, and how the data will be retrieved. The dialogue will also require authorization for any proprietary data to be retrieved. The dialogue will contain the following components:

- A prompt for entering the archive user name and password to be used for retrieving data. Some users will be privileged, and permitted to retrieve proprietary data from the archive. The Data Retrieval subsystem will be used to verify the user identification and the proprietary data privileges.
- A section listing the proposal id(s) associated with any proprietary data. The user will be prompted to enter the password associated with the proprietary data proposals in order to verify that they are permitted to retrieve the proprietary data.
- A list of data supersets selected to be retrieved.
- A list of data categories available for retrieval. These categories would be:
  - Raw data files should be delivered.
  - Calibrated data files should be delivered.
  - Data processing for associations should be done, and the results delivered. (E.g. if a stacked association is requested, the stacked image will be delivered.)
  - Science data. This data would include the output of sextractor, (see [9]), galfit (see [10]), and any similar processes.

  It will be possible for more than one level of processing to be selected. For example if both “No data processing” and “Basic calibration” are selected, then both the raw data and the basic calibrated data will be delivered.
- A list of auxiliary data available for the requested datasets. This list will allow user’s to select auxiliary data to be retrieved in addition to the data files directly associated with the dataset. The types of auxiliary files available are:
  - Calibration data associated with the requested data.
  - Environmental data associated with the requested data.
  - Observing programs associated with the requested data.
  - Observing logs associated with the requested data.
- A list of possible delivery media will be displayed. The default delivery media for the GSA will be FTP, but the user will have the option to get the data on CD-ROM, DVD, or EXABYTE tape.

In addition to the categories of data processing described above, the list of datasets will contain links to allow selection of data processing to be done for individual datasets. This link will lead to the data retrieval processing setup screen described in Section 2.1.17 on page 42.

On-line help will be available to explain the content of the page.

The result of this dialogue will be a data requested being submitted to the Data Retrieval Subsystem, described in Chapter 7. The user will be notified by email when the request is complete, and they will be given a link to the request status page described in Section 2.1.18 on page 42.

2.1.17 Data retrieval processing setup

In addition to the categories of data processing selectable as described in 2.1.16, users will be able to select specific data processing to be applied to each dataset. The data retrieval processing setup page allows a user to retrieve the output from specific recipe instances. This page would list each of the recipe instances available for a dataset, as extracted from the data processing tables described in Chapter 5, section 3. on page 80. Each recipe instance will have a control allowing the instance to be selected for execution, and a link to the recipe instance display page described in Section 2.1.12 on page 40.

On-line help will be available to explain the content of the page. Documentation of the recipes is available through the recipe instance display page.

2.1.18 Request status page

The request status page will provide the user with information about the progress of a request through the GSA. The page will display the state of each file in the request, and of the request as a whole.

2.1.19 Observation table query page

This page will be similar to the science table query page described in Section 2.1.2 on page 37, except content will come from the observations table described in Chapter 4, section 3. on page 51.

As part of the advanced GSA capabilities, this page will be extended to allow a “self join” based on overlapping field of view. This will allow a user to issue queries like “*find all data supersets like ... where there is a data superset like ... with an overlapping field of view*”. If the observation table is used by other archives, this feature will also allow cross archive queries. This extension will require either changes to the data superset result list to allow a link to the other data supersets, or an entirely new result list.

2.1.20 Proposal query page

This web page is a form which allows searches of the proposals in the proposal table described in Chapter 4, section 3. on page 51. The form will contain query fields for proposal id, science category, keywords, PI name, and any other field in the table. The keywords field may contain multiple entries, separated by commas. A control on the keyword field will allow the user to choose to combine the words entered using either “and” or “or” logic.

The page will contain links to on-line help which describes the content of each of the fields on the page.

The results of the search of the proposals will be displayed in the proposal list page described in Section 2.1.21 on page 42.

2.1.21 Proposal list page

The proposal list web pages contains a list of proposals in tabular form. All of the information stored in the proposal table described in Chapter 4, section 3. on page 51 will be included in as columns in the table.

Each of the proposals will contain a link which will display all datasets collected for the proposal. The datasets will be displayed using the data superset result list web page described in Section 2.1.3 on page 37.

Each of the proposals will have a link which will display all observing programs associated with the proposal. The observing programs will be displayed using the observing program list described in Section 2.1.13 on page 40.

Each of the columns in the proposal list will have links to on-line help describing the content of the column.

2.1.22 Preview display page

Wherever possible, data supersets in the GSA science table will have associated preview data. Preview data are stored in the preview table described in Chapter 4, section 3. on page 51. The preview table may contain more than one preview for each data superset. The default preview displayed the one with the lowest subId in the preview table. The preview display page will have controls which will allow the user to display any of the other previews for the data superset.

All previews will be stored as FITS files, containing either images to be displayed or tables (spectra) to be plotted. A FITS file viewer running on the client's system will be used to display the previews. JSky seems to be a likely candidate for use as the FITS viewer.

2.1.23 Publication list display

This page displays a list of publications associated with an observing program. The list will be displayed in tabular form, and will include basic bibliographic information:

- Title.
- List Of authors.
- Journal name.
- Publication date.
- NED/SIMBAD bibcode (see [13]).

The bibliographic information will be extracted "on-the-fly" from the The NASA Astrophysics Data System (see [18]), based on the ADS references stored in the publications table. Each publication in the list will contain a link to the abstract at the ADS site.

2.1.24 Spatial coordinate plot page

This page will contain a plot of the fields of view of a set of data supersets. The plot will be displayed with the same FITS viewer used for the preview display page described in Section 2.1.22 on page 43. If JSky is used, it will be possible to overlay the field of view plot with catalogue data products available through JSky, including sky survey images.

2.2 Batch Interface

The batch interface to the GSA will use email messages to allow users to send requests and receive responses. The subject of the email messages will control the action of the email message. The supported message types are described in Table 5 on page 44, and are described in detail in the following sections.

TABLE 5. Batch interface messages

Message subject	Description
GSA Batch Help	This message is a request for information about the GSA batch interface. The response to this message will be a pre-formatted help message describing the batch interface to the GSA. This description will include a pointer to a FTP site where a complete set of documentation for the GSA is located. The complete set of documentation will be a tar file, containing a set of HTML pages which describe the GSA, the data contained in the GSA, attributes, etc.
GSA Science Form	This message is a request for an empty science table query form. The response to this message will be an empty science table query form, as described in Section 2.2.1 on page 45. After the form is filled in, it will be used as input to the GSA Science Query message.
GSA Science Query	This is a query to the GSA science table. The science query form is described in Section 2.2.1 on page 45. The response to this message is described in Section 2.2.2 on page 45.
GSA Observation Query Form	This message is a request for an empty observation table query form. The response to this message will be an empty observation table query form, which will be similar to the science table query form described in Section 2.2.1 on page 45. After the form is filled in, it will be used as input to the Observation Query message.
GSA Observation Query	This is a query to the GSA observation table. The response to this message is a list of observations. The list of observations will be similar to the science query response described in Section 2.2.2 on page 45.
GSA Data Superset Form	This message is a request for an empty data superset query form. The response to this message will be an empty dataset table query form, which will be similar to the science query form described in Section 2.2.1 on page 45. After the form is filled in, it will be used as input to the Data Superset Query message.
GSA Data Superset Query	This is a query to the GSA data superset tables. The response to this message is a list of datasets, similar to the science query response described in Section 2.2.2 on page 45.
GSA Source Query Form	This message is a request for an empty source query form. The response to this message will be an empty source query form as described in Section 2.2.4 on page 46. After the form is filled in, it will be used as input to the Source Query message.
GSA Source Query	This is a query to the GSA source tables. The response to this message is a list of sources as described in Section 2.2.5 on page 46.
GSA Object Query Form	This message is a request for an empty object query form. The response to this message will be an empty object query form, similar to the source query form described in Section 2.2.4 on page 46. After the form is filled in, it will be used as input to the Object Query message.
GSA Object Query	This is a query to the GSA object tables. The response to this message is a list of objects as similar to the list of sources described in Section 2.2.5 on page 46.
GSA Data Superset Request Form	This message is a request for an empty data superset request form. The response to this message will be a dataset request form, as described in Section 2.2.3 on page 45. After the form is filled in, it will be used as input to the Data Superset Request message.
GSA Data Superset request	This message is a request for a list of data supersets. This will result in a query being submitted to the Data Retrieval Subsystem described in Chapter 7.

2.2.1 Science query form

The science query form will be a text file containing set of attribute names, with space to enter qualifiers. For example, a section of the blank form might look like:

- instrument: ?
- mode: ?

The user would add qualifications into the file, creating a form that looks like:

- instrument: GMOS
- mode: MASK or IFU

The “?” in the qualifier indicates that any value is acceptable, and that the value of field should be returned in the table of the science query response. A blank value indicates that any value is acceptable, and the value of the field should not be returned in the table of the science query response. The initial science query form will contain a default set of attributes with the “?” value, which the user will modify as required.

A field in the form will allow the user to indicate the maximum number of rows to return in the response. A field in the form will allow the user to retrieve detailed information about each dataset.

After filling in the form, the user will mail it to the GSA in a Science Query message. The result will be returned to the user in a science query response, as documented in Section 2.2.2 on page 45.

2.2.2 Science query response

The science query response is returned in response to the science query described in Section 2.2.1 on page 45. The science query response will contain three sections:

1. A copy of the query form used to submit the query.
2. A tabular list of attributes of each matching dataset. Each row in the table will be one data superset which matches the query, and each column in the table will by an attribute of the data superset. The returned attributes are controlled by the science query form.
3. If specified in the query form, detailed information about each data superset in the table will be included. All attributes will be included for each dataset, not just the attributes requested by the user. The information in this section is equivalent to the information in the data superset detail page described in Section 2.1.5 on page 38, with the following additions:
   - A list of recommended calibration datasets.
   - A list of members of associated datasets.

The tabular list of datasets will be formatted in a way which allows the rows to be pasted into the Data Superset request described in Section 2.2.3 on page 45.

2.2.3 Data Superset Request form

The data superset request form contains fields identifying the user, the users privileges to access proprietary data, the level of data processing to be done, and the list of data supersets to be retrieved.

The data processing levels will be the same as those described in Section 2.1.16 on page 41. Specification of detailed processing, such as done with the data processing setup web paged described in Section 2.1.17 on page 42 will not be possible with the batch interface.

The datasets requested will be listed one per line. To allow simple cut and past creation of a request from lists of datasets, the dataset name will be assumed to be the first word on the line, and any subsequent words will be ignored.

The user will send the form to the GSA in a Data Superset Request message. The result of this message will be a data request being submitted to the Data Retrieval Subsystem, described in Chapter 7. The user will be notified by email when the request is complete.

2.2.4 Source query form

This form will contain fields for each of the source properties stored in the GSA source tables, with a place the user can specify a qualifier. The general format of the form will be the same as the science query form described in Section 2.2.1 on page 45.

There will be a fields in the form which allow the user to limit the number of rows returned, and to indicate if detailed information for each source should be included. Like the science query form, a qualifier of “?” indicates that the value of a field should be returned in the tabular part of the result, and a blank qualifier indicates the value should not be included in the tabular part of the result.

Note: This form may by somewhat unwieldy if there are several hundred properties.

2.2.5 Source query result

The source query result list is returned in response to a source query message sent to the GSA. The message will contain three sections:

1. A copy of the query form used to submit the query.
2. A tabular list of sources and attributes of each matching source. Each row in the table will be one source which matches the query, and each column will be an attribute of the source. The attributes in the table are controlled by the source query form.
3. If specified in the query form, detailed information about each source will be included. All attributes of the source will be returned, not just those requested by the user. The information in this section is equivalent to the information presented in the source detail page described in Section 2.1.11 on page 40

Created:August 31, 2001
Modified:May 21, 2002

Gemini Science Archive

Chapter 4 Catalogues

1. Introduction

This chapter describes the conceptual design of the GSA Catalogues.

The purpose of the catalogue subsystem is to:

- Store meta-data about science datasets in a queryable form.
- Store meta-data about associated data supsersets.
- Store preview images.
- Store environmental data.
- Store meta-data about proposals.
- Store meta-data about science programs.
- Store the electronic observing log.
- Store a source catalogue.
- Store an object catalogue.
- Provide querying capabilities for all of the data listed above.

This subsystem is based on a variety of other systems at the CADC:

- The Gemini Prototype archive developed in parallel with this design (see [11]).
- Existing practices at the CADC.
- A prototype object catalogue developed at the CADC.

1.1 Requirements on the GSA Catalogue subsystem

Table 6 on page 47 list the software requirements which at least partially apply to the GSA Catalogue subsystem. Each requirement is followed by a short description of how it will effect the catalogue subsystem. See [6] for a complete description of these requirements.

TABLE 6. Catalogue requirements

SR1.1 The GSA shall archive all data. The catalogue database schema will handle data from all Gemini instruments. The catalogue will have the storage capacity to store all data provided by Gemini. The catalogue will support the projected data ingest rates.
SR1.4.2 Remove previews when release date is changed. The catalogue will support deletion of preview data. The catalogue may be involved in propagating changes to the release date descriptor to remove derived data products.
SR1.4.3 Preview should be mirrorable. The catalogue database shall support a mirroring mechanism.

TABLE 6. Catalogue requirements

SR2.1 The GSA catalogue shall contain all data descriptors. The catalogue database schema will be able to store all data desciptors supplied by Gemini.
SR2.2 The catalogue shall include environmental information. Store the environmental data.
SR2.3 The catalogue shall include accurate WCS information. Store the WCS data.
SR2.4 Proposal information should be associated with data supersets. Database schema must link data supersets to 0 or more proposals.
SR2.5 The catalogue should include image descriptors derived from the data. Store the derived descriptors.
SR2.5.1 Remove derived image descriptors for proprietary data. Allow descriptors to be deleted. The catalogue may be involved in propagating changes to the release data descriptor to remove derived data products. See SR1.4.2.
SR2.6 Release date is a data descriptor. Store the release date.
SR2.7 The GSA catalogue should contain associated datasets. Store the associations. Associations will complicate relationships to other data.
SR2.8 Data quality assessment shall be in the catalogue. Store the data quality assessment.
SR2.9 The science program shall be associated with the catalogue data. Store the science program. Link science program to data supersets.
SR2.10 The electronic observing log shall be associated with the catalogue. Store the electronic observing log. Link electronic observing log to data supersets.
SR2.11 Proposal id shall be a descriptor. Store the data. There may be more than one proposal for each data superset.
SR2.12 The GSA catalogue shall be mirrorable. The catalogue database shall support a mirroring mechanism.
SR2.14 The GSA should have an object catalogue. Store the object catalogue.
SR2.14.1 Object parameters should be extracted from object catalogues. Store the object catalogue. Store the external (SIMBAD/NED) object catalogues?
SR2.14.2 The GSA object catalogue should be up to date. Update GSA object catalogues when SIMBAD/NED change.
SR2.14.3 The object catalogue should include object descriptors derived from data. Store descriptors supplied by data processing. This implies there will be many descriptors, and they will be sparse.
SR2.14.4 Remove derived objects for proprietary data. Updates to release dates cause derived objects to be removed from the object catalogue.
SR2.15 Publications should be linked to data supersets. Catalogues include information to link data supersets to publications.
SR2.16 The GSA shall store Gemini hardware and software versions. The catalogues store hardware and software versions supplied by Gemini.
SR3.2 Select subsets of the GSA catalogue. The catalogues will support queries. Queries shall execute efficiently.

TABLE 6. Catalogue requirements

SR3.2.1 Select subsets of the GSA catalogue based on coordinate region search. The data superset catalogue will support efficient searches based on the field of view of the data superset overlapping a user specified region.
SR3.2.2 GSA users should be able to select data supersets based on object descriptor. The data superset catalogue and object catalogue will support searches based on object descriptors. This implies a join between the object table and the data superset table, and that the object table can be searched by object descriptor.
SR3.2.3 Support cross catalogue searches. The GSA data superset catalogue can be joined with other catalogues, based on overlapping field of view.
SR3.3 View descriptors for a subset of the catalogue. The GSA will store data superset descriptors, and retrieve them when required.
SR3.3.1 There shall be on-line help describing all catalogue data descriptors. The on-line help could be stored in the catalogue database and retrieved on demand.
SR3.3.3 View the list of publications associated with a data superset. Publications associated with a data superset must be retrievable.
SR3.3.4 View descriptors of objects in a data superset field of view. It must be possible to join the object catalogue with the data superset catalogue, based on field of view. Note that this is not a join based on objects detected on an image, but is objects which should be in the field of view.
SR3.3.5 View proposal information associated with a data superset. It must be possible to join the proposal catalogue with the data superset catalogue. Proposal information will be stored in a catalogue.
SR3.3.9 GSA users shall be able to view the electronic observing log. The content of the electronic observing log must be retrievable. The electronic observing log will be stored in a catalogue.
SR3.3.10 View environmental data. The environmental data must be stored in a catalogue. The environmental data must be retrievable, indexed by date and time. The environmental data catalogue must be joinable with the data superset catalogue.
SR3.3.11 Display science programs. Science programs must be stored in a catalogue. Science programs must be retrievable. The science program catalogue must be joinable with the data superset catalogue.
SR3.3.12 Display the members of association data supersets. The data superset catalogue must be able to join data supersets to their member data supersets.
SR3.4.8 Retrieve environmental data. There must be a catalogue of environmental data. The environmental data catalogue must be searchable by date/time. The environmental data must be retrievable to a FITS file.
SR3.4.9 It shall be possible to retrieve science programs. Science programs must be stored in a catalogue. Science programs must be retrievable. The science program catalogue must be joinable with the data superset catalogue. The science programs must be saveable to a file.
SR3.4.10 It shall be possible to retrieve proposal information. It must be possible to join the proposal catalogue with the data superset catalogue. Proposal information will be stored in a catalogue. Proposals must be retrievable to a file.
SR3.4.11 It shall be possible to retrieve the electronic observing log. The content of the electronic observing log must be retrievable to a file.

TABLE 6. Catalogue requirements

SR3.6 Proposal information should be searchable. Proposal information must be stored in a catalogue. The catalogue must contain searchable fields for keyword and science category.
SR3.6.1 Find all data supersets associated with proposals. The proposal catalogue must be joinable with the data superset catalogue.
SR3.7 Preview display for each science data superset. There must be a preview catalogue. The preview catalogue must be joinable with the data superset catalogue.
SR4.1 Maximum and average requirements for catalogue ingest. Catalogues must support the ingest rates.
SR4.4 Preview should be display promptly. If preview is stored in a catalogue, the data from the preview must be quickly accessible.
SR4.5 Access to the GSA catalogue must be interactive. All data in the catalogues must be quickly available. Catalogues must be quickly searchable.

2. Overview

The GSA catalogues will contain database tables which described the GSA content from five different views:

1. The data superset tables store all information about all data supersets in the GSA. This will include both simple datasets created by Gemini, and associated data supersets created at the GSA.
2. The science table stores scientifically interesting information about scientifically interesting data supersets in the GSA (as determined by Gemini and GSA staff). The information in this table is a subset of the information stored in the data superset tables. This table exists in order to minimize the “clutter” that a GSA user would see when searching for data in the GSA through the data superset tables.
3. The observation table is similar to the science tables, except that the information it contains is stored in an archive independent fashion. The need to derive the archive independent information from pixel data will require that this table only contain public data supersets. This table will be shared with other archives stored at the archive centre, allowing users to perform multi-archive searches.

   The observation overlap table will allow queries which require field of view overlaps between different types of observations. Since the observation tables are archive independent, this will also allow cross-archive searches.
4. The source catalogue tables. These tables describe sources detected in the data supersets. The sources will be detected by data processing recipes, including those which extract catalogue data from SIMBAD and NED.
5. The object catalogue tables. These tables contain a re-interpretation of the data in the source catalogue tables. Information from sources on different datasets will be assembled into a single set of attributes which describe each object in the physical universe. (We haven’t actually figured out how to do this yet, so this should be considered on ongoing project.)

Each of these views describe the GSA content with increasing abstraction. Chapter 3 describes the user interfaces which provide GSA access to each of these views.

3. Catalogue Storage

The GSA catalogues will be stored in a Sybase relational database. Sybase was chosen because it is currently the standard RDMS used by the CADC. An overview of the relationships between all of the tables in the catalogues is impractical due to the number of tables, and the number of relationships between them. Instead, the tables have been grouped into categories, with a diagram for each category. The categories are:

- Data superset tables. See Figure 7 on page 54.
- Science tables. See Figure 8 on page 59.
- Observation tables. See Figure 9 on page 62.
- Source catalogue tables. See Figure 10 on page 65.
- Object catalogue tables. See Figure 11 on page 68.

3.1 Shared tables

These tables are shared by all sub-groups of the GSA catalogue tables.

3.1.1 Table: attribute

The attribute table describes each attribute used in the GSA catalogues. These attributes include attributes of the data supersets, attributes of sources in the source catalogue, and attributes of objects in the object catalogue.

Information about each attribute includes labeling information, information describing the units of measurement, and formatting information.

The columns of table attribute are shown in Table 7 on page 51.

TABLE 7. Columns of table attribute

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
attributeId	The unique attribute identifier.	binary(4)	TRUE	TRUE	FALSE
format	Formatting string to use when displaying data.	char(6)	TRUE	FALSE	FALSE
category	This is the category of the attribute. See the text for a description of the categories.	char(6)	TRUE	FALSE	FALSE

The attribute category gives a hint to the user interfaces about how data should be organized. This should only be considered a hint, since a user interface may choose to arrange data independently of this value. Some examples of categories for Gemini attributes might be:

- Telescope configuration.
- Instrument configuration.
- ALTAIR configuration.
- Environment.
- Data quality.

The format column will contain a C printf style formatting string to be used when formatting data items.

3.1.2 Table: fieldOfView

This table describes the field of view of each data superset. The field of view is defined as one or more simple closed polygons in ra and dec which describe the area of the sky observed in a data superset.

The columns of table fieldOfView are shown in Table 8 on page 52.

TABLE 8. Columns of table fieldOfView

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
dataSuper-setId	The data superset associated with this field of view.	binary(8)	TRUE	TRUE	FALSE
order	The order in which the field of view records should be assembled. See the text for a description of this column.	integer	TRUE	FALSE	FALSE
polygon	The polygon field contains the points that form the polygon describing the field of view.	varbi-nary(128)	TRUE	FALSE	FALSE

In those cases where the field of view is too complex to fit into a single array of 16 pairs of points, more than one row can be used to describe the polygon. The order column indicates the order in which the rows should be assembled to create the final polygon.

The field of view polygon is described in the polygon field. This field is a set of up to 16 pairs of 4 byte integer coordinates, in ra and dec (both described in milliseconds of arc). If a field of view must be described with several polygons, a coordinate value of 0xffff, 0xffff can be inserted into the coordinate list to indicate that the current polygon should be closed, and the next coordinate will be the first point in a new polygon.

All polygons will be closed automatically, and so it is not necessary to have the last point equal to the first point.

3.1.3 Table: previewGemini

This table contains the preview images for the Gemini archive. The preview data are FITS files stored in the database, containing either a single image, or a FITS table binary or ASCII to be plotted as a line graph. The FITS file should be sized to allow a typical user to download the image in less that 10 seconds. With current network bandwidths, that would be smaller than about 500 Kilobytes, however the maximum preview size will grow as bandwidth increases. Each data superset may have more than one preview image, each of which will be stored in one row of the preview table.

The columns of table previewGemini are shown in Table 9 on page 53.

The valid compression types stored in the mode column are:

"GZIP" — The file is compressed using the gzip program.

"HCMP" — The file is compressed using the hcompress program.

TABLE 9. Columns of table previewGemini

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
dataSuper-setId	This is the name of the preview data superset. This is joined with the dataSupersetId column in the science table.	varchar(32)	TRUE	TRUE	FALSE
subId	This value allows each data superset to have more than one preview. The different previews will have different subIds.	tinyint	TRUE	TRUE	FALSE
title	This is the title to be used for the preview.	varchar(64)	TRUE	FALSE	FALSE
format	This is the type of data stored in the preview image. Valid values are "IM" if the value is an image, or "SP" if the value is a spectrum.	char(2)	TRUE	FALSE	FALSE
mode	This is the compression mode. See the text for a list of valid compression types.	char(4)	TRUE	FALSE	FALSE
creation_date	The date the preview was created.	int	TRUE	FALSE	FALSE
ingest_date	This is the date the preview was inserted into the data-base.	int	TRUE	FALSE	FALSE
nbytes	This is the compressed size of the preview file.	int	TRUE	FALSE	FALSE
data	This is the preview data file.	image	TRUE	FALSE	FALSE
tuple_id	This is a unique identifier which allows the preview table to be mirrored between sites. It will automatically be assigned a value when new rows are inserted into the table.	timestamp	TRUE	FALSE	FALSE
obytes	This is the uncompressed size of the preview file.	int	TRUE	FALSE	FALSE

"UCMP" — The file is compressed using the Unix compress program.

In the cases where more than one preview is available for a data superset, it should be possible to present the user with a list of the preview titles to allow them to select the desired preview.

The indexes of table previewGemini are shown in Table 10 on page 53.

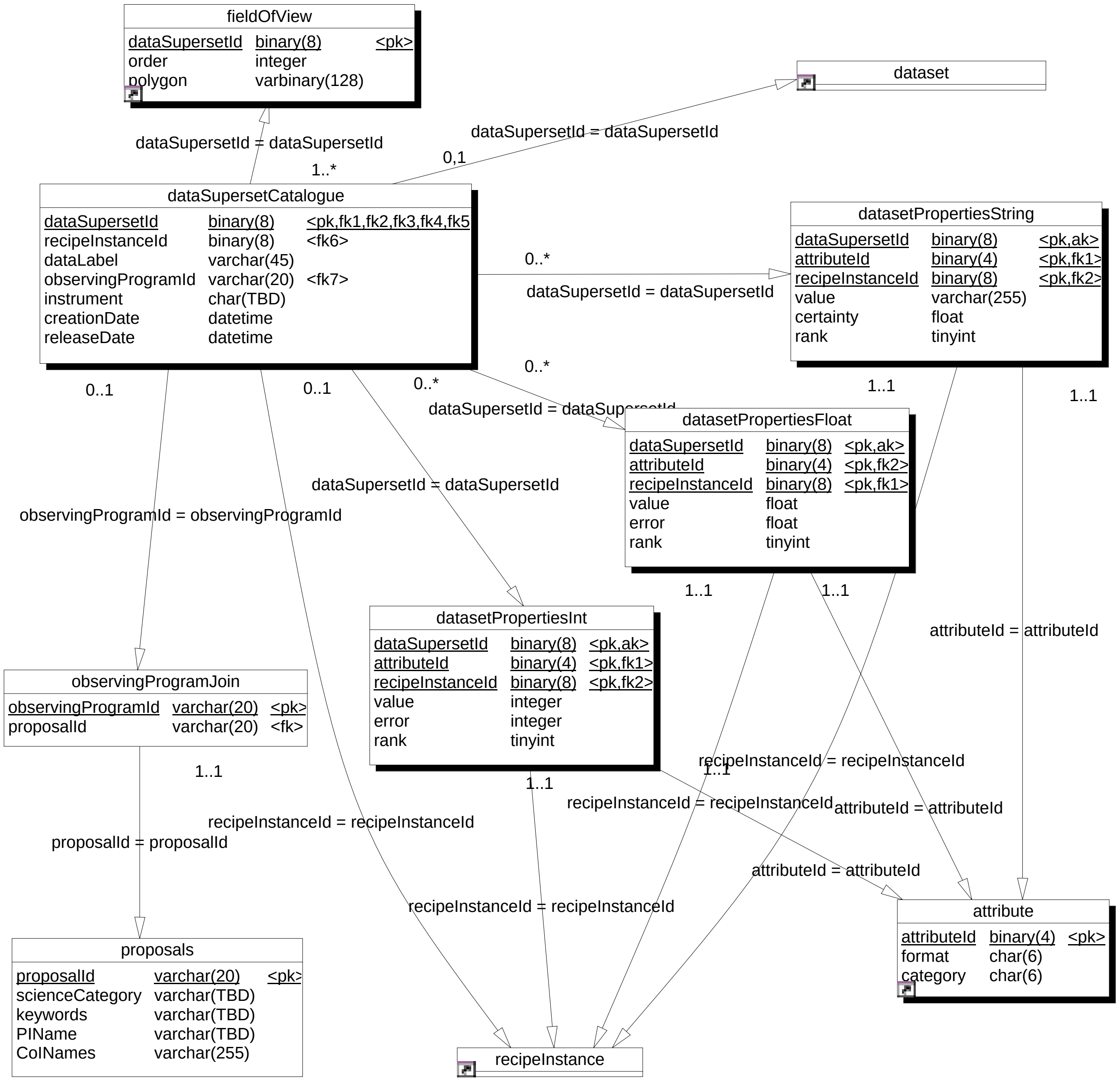
TABLE 10. Indexes of table previewGemini

Name	Comment	Unique	Cluster
preview_nam e_idx	This index allows previews to be found by data superset name.	TRUE	FALSE
preview_tupl e_index	This index allows previews to be found based on the tuple id.	TRUE	FALSE

3.2 Data superset tables

These tables contain all information about all data supersets. This will include science and calibration datasets provided by Gemini, and associated data supersets identified at the GSA.

FIGURE 7. Data superset table relationships


fieldOfView
dataSupersetId binary(8) <pk>
order integer
polygon varbinary(128)
dataset
dataSupersetId = dataSupersetId
1..*
dataSupersetId = dataSupersetId
0,1
dataSupersetCatalogue
dataSupersetId binary(8) <pk,fk1,fk2,fk3,fk4,fk5>
recipeInstanceId binary(8) <fk6>
dataLabel varchar(45)
observingProgramId varchar(20) <fk7>
instrument char(TBD)
creationDate datetime
releaseDate datetime
datasetPropertiesString
dataSupersetId binary(8) <pk,ak>
attributeId binary(4) <pk,fk1>
recipeInstanceId binary(8) <pk,fk2>
value varchar(255)
certainty float
rank tinyint
0..*
dataSupersetId = dataSupersetId
0..*
0..1
0..1
0..*
dataSupersetId = dataSupersetId
1..1
1..1
datasetPropertiesFloat
dataSupersetId binary(8) <pk,ak>
attributeId binary(4) <pk,fk2>
recipeInstanceId binary(8) <pk,fk1>
value float
error float
rank tinyint
dataSupersetId = dataSupersetId
observingProgramId = observingProgramId
1..1
1..1
attributeId = attributeId
datasetPropertiesInt
dataSupersetId binary(8) <pk,ak>
attributeId binary(4) <pk,fk1>
recipeInstanceId binary(8) <pk,fk2>
value integer
error integer
rank tinyint
observingProgramJoin
observingProgramId varchar(20) <pk>
proposalId varchar(20) <fk>
1..1
1..1
recipeInstanceId = recipeInstanceId
1..1
recipeInstanceId = recipeInstanceId
attributeId = attributeId
recipeInstanceId = recipeInstanceId
proposalId = proposalId
attributeId = attributeId
recipeInstanceId = recipeInstanceId
attribute
attributeId binary(4) <pk>
format char(6)
category char(6)
proposals
proposalId varchar(20) <pk>
scienceCategory varchar(TBD)
keywords varchar(TBD)
PIName varchar(TBD)
CoINames varchar(255)
recipeInstance

3.2.1 Table: dataSupersetCatalogue

This table contains one row for every data superset in the GSA. Data supersets in this table will also include non-science data supersets.

The columns of table dataSupersetCatalogue are shown in Table 11 on page 55.

In addition to being used as a join value within the data processing catalogue tables, the recipe instance id may be used as an identifier in other parts of the GSA, whenever it is necessary to

TABLE 11. Columns of table dataSupersetCatalogue

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
dataSuper-setId	The unique identifier for this data superset.	binary(8)	TRUE	TRUE	TRUE
recipeInstan-ceId	This is a unique identifier for this recipe instance, and is used when joining with the recipe input table. See the text for a description of the use of this value.	binary(8)	FALSE	FALSE	TRUE
dataLabel	The data label associated with this data superset.	varchar(45)	TRUE	FALSE	FALSE
observing-ProgramId	The unique identifier for an observing program.	varchar(20)	FALSE	FALSE	TRUE
instrument	The instrument which produced the data. This may be NULL if multiple instruments contribute to a data superset.	char(TBD)	TRUE	FALSE	FALSE
creationDate	The date the data superset was created. For a simple dataset this will be the date and time the FITS file was created, as indicated in the FITS header. For an associated data superset, this would be the date and time the association was identified.	datetime	TRUE	FALSE	FALSE
releaseDate	This is the date and time the data superset becomes available to the general public.	datetime	TRUE	FALSE	FALSE

identify specific recipe instances. One example of this would be in the GSA science catalogues, where it is necessary to associate a derived parameter value with the recipe instance which produced the value.

3.2.2 Table: datasetPropertiesFloat

This table contains the floating point properties of a data superset.

The columns of table datasetPropertiesFloat are shown in Table 12 on page 55.

TABLE 12. Columns of table datasetPropertiesFloat

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
dataSuper-setId	The data superset id associated with this attribute.	binary(8)	TRUE	TRUE	FALSE
attributeId	The id of this attribute.	binary(4)	TRUE	TRUE	TRUE
recipeInstan-ceId	This value identifies the recipe instance which produced this value.	binary(8)	TRUE	TRUE	TRUE
value	The value of the attribute.	float	TRUE	FALSE	FALSE
error	The error associated with the attribute.	float	TRUE	FALSE	FALSE
rank	The rank of this value relative to other known values for this attribute.	tinyint	TRUE	FALSE	FALSE

The attribute id is used to join with the attribute table, which provides detailed information about the attribute.

The error column should give an indication of the accuracy of the attribute's value. In general, a 3 sigma error bound should be used, however given the wide variety of processing that will be used to produce attributes, it may not be possible to use 3 sigma in all cases.

The rank will be used when there are several different sources for the same attribute. The data processing software will evaluate the confidence in each of the values, and assign a rank based on the relative levels of confidence. The most certain value will be given a rank of 1, the second most certain will be given a rank of 2, etc.

3.2.3 Table: datasetPropertiesInt

This table contains the integer properties. This table is similar to the datasetPropertiesFloat table described in Section 3.2.2 on page 55.

The columns of table datasetPropertiesInt are shown in Table 13 on page 56.

TABLE 13. Columns of table datasetPropertiesInt

Name	Comment	Data Type	Mandatory	Primary	Foreign Key
dataSupersetId	The data superset id associated with this attribute.	binary(8)	TRUE	TRUE	FALSE
attributeId	The unique identifier for this attribute.	binary(4)	TRUE	TRUE	TRUE
recipeInstanceId	The unique identifier of the recipe instance which produced this value.	binary(8)	TRUE	TRUE	TRUE
value	The attribute's value.	integer	TRUE	FALSE	FALSE
error	The error associated with this value.	integer	TRUE	FALSE	FALSE
rank	The rank of this value relative to other values for this attribute.	tinyint	TRUE	FALSE	FALSE

3.2.4 Table: datasetPropertiesString

Stores string properties of data supersets. This is similar to the datasetPropertiesFloat table described in Section 3.2.2 on page 55.

The columns of table datasetPropertiesString are shown in Table 14 on page 56.

TABLE 14. Columns of table datasetPropertiesString

Name	Comment	Data Type	Mandatory	Primary	Foreign Key
dataSupersetId	The data superset id associated with this attribute.	binary(8)	TRUE	TRUE	FALSE
attributeId	The id of this attribute.	binary(4)	TRUE	TRUE	TRUE
recipeInstanceId	This value identifies the recipe instance which produced the value.	binary(8)	TRUE	TRUE	TRUE
value	The value of the attribute.	varchar(255)	TRUE	FALSE	FALSE

TABLE 14. Columns of table datasetPropertiesString

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
certainty	This is an estimate of the degree of certainty for this value.	float	TRUE	FALSE	FALSE
rank	The rank of this value relative to other known values for this attribute.	tinyint	TRUE	FALSE	FALSE

The attribute id is used to join with the attribute table, which provides detailed information about the attribute.

The certainty value should be 1.0 if it is certain that the value is correct. This value would be used when data processing can not resolve categorizations, and for example can only conclude that there is certainty of .6 that on object is a spiral galaxy, and a certainty of .4 that it is an irregular galaxy.

3.2.5 Table: observingProgramJoin

This table is used to identify the proposals associated with an observing program. This information must be stored in a separate table because there are instances where one observing program may be associated with several proposals

The columns of table observingProgramJoin are shown in Table 15 on page 57.

TABLE 15. Columns of table observingProgramJoin

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
observing-ProgramId	The unique identifier for an observing program	varchar(20)	TRUE	TRUE	FALSE
proposalId	The unique identifier for a proposal.	varchar(20)	TRUE	FALSE	TRUE

3.2.6 Table: observingPrograms

This table contains the observing programs executed by Gemini. The information in this table will be extracted from the observing programs supplied by Gemini in the meta-data database.

The columns of table observingPrograms are shown in Table 16 on page 57.

TABLE 16. Columns of table observingPrograms

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
observing-ProgramId	This is the unique identifier Gemini has assigned to each observing program.	varchar(20)	TRUE	TRUE	FALSE
programText	The is the observing program.	text	TRUE	FALSE	FALSE
timestamp	This is the timestamp taken from the Gemini meta-data store.	timestamp	TRUE	FALSE	FALSE

The programText column will contain an XML document describing the observing program. Only public parts of the observing program will be included in this document. The document will be suitable for loading into the Gemini Observing tool.

3.2.7 Table: proposals

This table contains information describing each Gemini proposal. The information in this table is extracted from the XML observing programs, as originally provided by Gemini, in the Meta-Data Database. Only one row is included in the table for each proposal, even if a proposal is associated with several observing programs.

The columns of table proposals are shown in Table 17 on page 58.

TABLE 17. Columns of table proposals

Name	Comment	Data Type	Mandatory	Primary	Foreign Key
proposalId	This is the unique identifier for the proposal.	varchar(20)	TRUE	TRUE	FALSE
scienceCategory	This is the science category for the proposal.	varchar(TBD)	TRUE	FALSE	FALSE
keywords	The keywords assigned to the proposal.	varchar(TBD)	TRUE	FALSE	FALSE
PIName	The name of the principal investigator.	varchar(TBD)	TRUE	FALSE	FALSE
CoINames	The names of the co-investigators.	varchar(255)	FALSE	FALSE	FALSE

3.3 Science Tables

The science table contains scientifically interesting information about scientifically interesting data supersets.

3.3.1 Table: science

The science table contains information about all of the "scientifically interesting" data supersets in the GSA. This table will be one of the primary sources of information for the use interface. Datasets which have no "stand alone" science value, and whose science value is superseded by associated data supersets will not be included in this table (e.g. an associated data superset made up of many short exposure time NIRI observations would be put into the science table, and the individual datasets which make up the association would be omitted from the table). Figure 8 on page 59 shows how the science table relates to the other tables in the GSA databases. New fields will be added to the science table as new "interesting" attributes are identified.

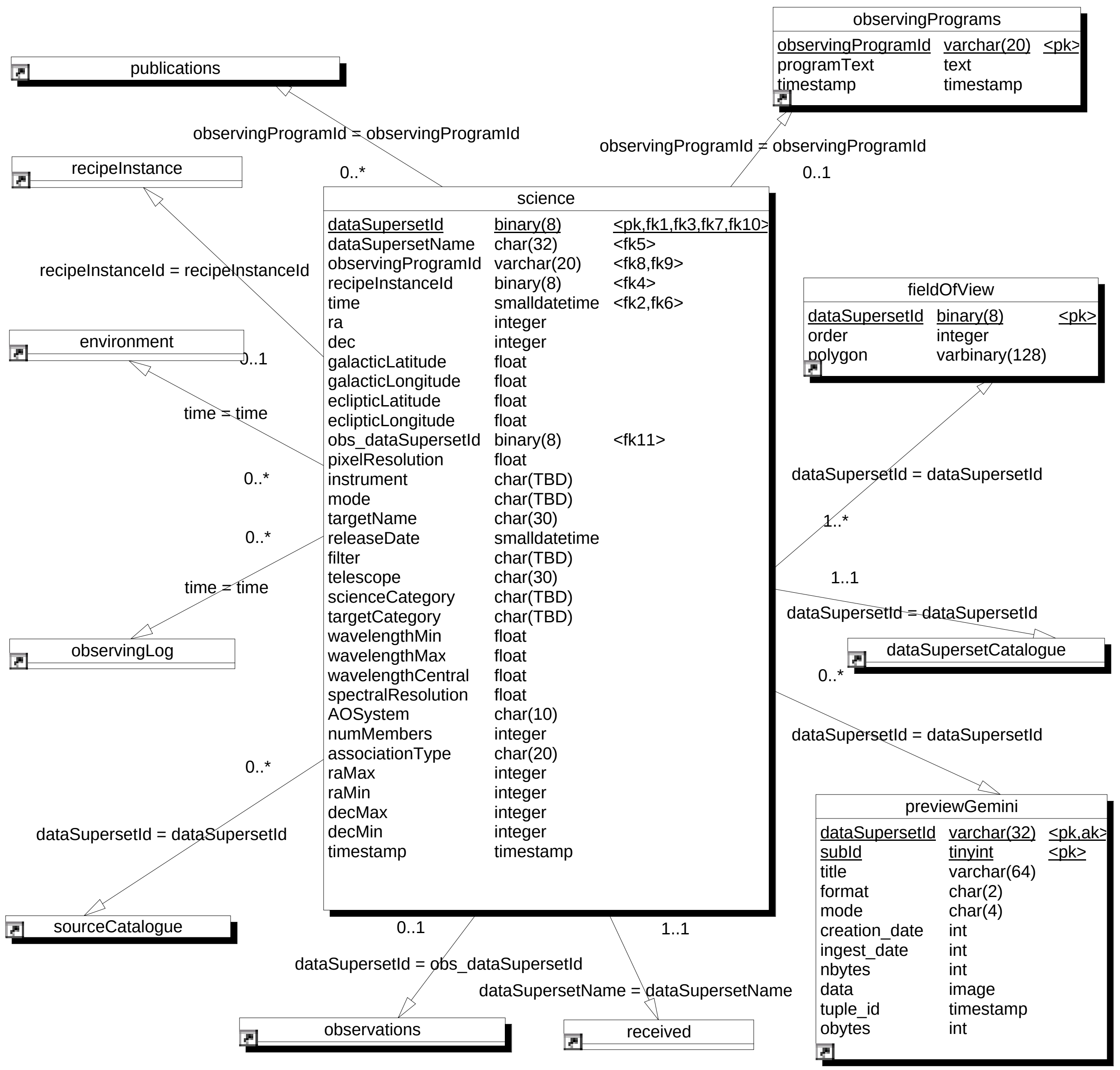
The columns of table science are shown in Table 18 on page 60.

Some typical association types might be:

"Stack" — The association is a stack of images. The recipe instance would be the recipe which co-adds the members.

"Mosaic" — The association is a mosaic of images. The recipe instance would be the recipe which assembles the members into a single image.

FIGURE 8. Science table relationships



The types of associations should be re-used whenever possible, and extended when necessary.

The mode column contains the mode the instrument was in when the data was collected. For example, GMOS can operate in imaging mode, mask mode, or integral field unit mode. When an instrument has no modes, this value should be NULL.

In addition to being used as a join value within the data processing catalogue tables, the recipe instance id may be used as an identifier in other parts of the GSA, whenever it is necessary to identify specific recipe instances. One example of this would be in the GSA science catalogues,

TABLE 18. Columns of table science

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
dataSuper-setId	This is the unique data superset id for this data superset.	binary(8)	TRUE	TRUE	TRUE
dataSuperset-Name	The name of the data superset. For simple datasets, this will be the data labels created at the Gemini telescopes. For associated data supersets, this will be a label assigned by the GSA.	char(32)	TRUE	FALSE	TRUE
observing-ProgramId	This is the observing program Id associated with this data superset. A NULL value indicates that no observ-ing program is associated with the data. A value of "*" will be used when an associated data superset incorpo-rates data from more than one observing program.	varchar(20)	FALSE	FALSE	TRUE
recipeInstan-ceId	This is a unique identifier for this recipe instance, and is used when joining with the recipe input table. See the text for a description of the use of this value.	binary(8)	FALSE	FALSE	TRUE
time	The time of the start of the observation.	smalldate-time	FALSE	FALSE	TRUE
ra	Right ascension of the center of the observation, in mil-liseconds of arc.	integer	TRUE	FALSE	FALSE
dec	Declination of the center of the image, in milliseconds of arc.	integer	TRUE	FALSE	FALSE
galacticLati-tude	Galactic latitude of the center of the image in degrees.	float	TRUE	FALSE	FALSE
galacticLon-gitude	Galactic longitude of the center of the image, in degrees.	float	TRUE	FALSE	FALSE
eclipticLati-tude	Latitude in ecliptic plane coordinates.	float	TRUE	FALSE	FALSE
eclipticLon-gitude	Longitude in ecliptic plane coordinates.	float	TRUE	FALSE	FALSE
obs_dataSupe rsetId	The unique identifier for this data superset.	binary(8)	FALSE	FALSE	TRUE
pixelResolu-tion	Pixel resolution in arc seconds.	float	TRUE	FALSE	FALSE
instrument	The name of the instrument which collected the data.	char(TBD)	TRUE	FALSE	FALSE
mode	The mode of the instrument which collected the data. If an instrument has multiple levels of modes, then the mode strings will be concatenated.	char(TBD)	FALSE	FALSE	FALSE
targetName	The name of the target of the observation, as taken from the proposal. This will be null if the observation is an association of datasets with different targets.	char(30)	FALSE	FALSE	FALSE
releaseDate	The public release date of the data superset.	smalldate-time	TRUE	FALSE	FALSE
filter	The name of the filter used for the observation. If more than one filter is in use, the filter names should be con-cantenated.	char(TBD)	TRUE	FALSE	FALSE

TABLE 18. Columns of table science

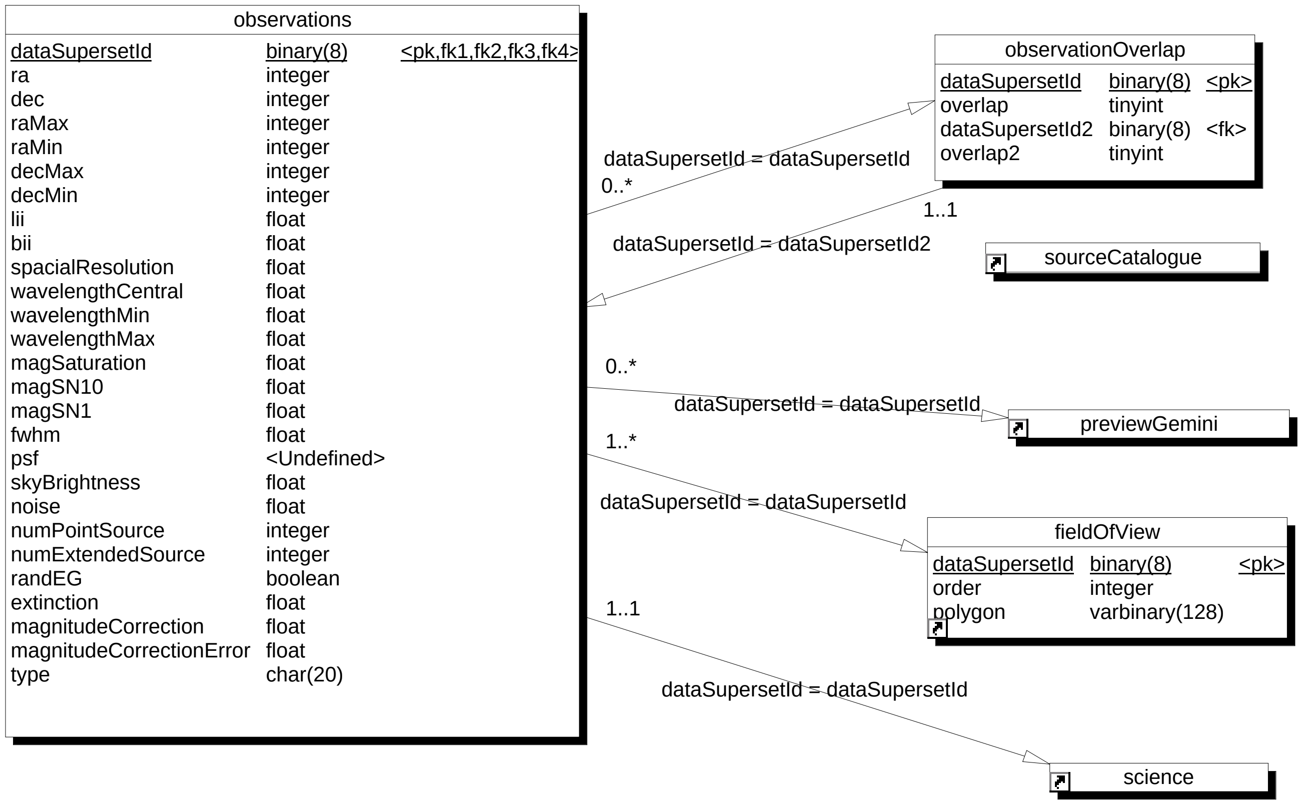
Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
telescope	The identity of the telescope. This should be "Gemini North" or "Gemini South".	char(30)	TRUE	FALSE	FALSE
scienceCate-gory	The category of science from the proposal. This is NULL if there is no associated proposal, or if the data superset is an association of data from proposals with different science categories.	char(TBD)	FALSE	FALSE	FALSE
targetCate-gory	The category of the primary target of the observation as given in the observing program. This will be NULL if no category is available, or if the data superset is an association of several observations with different target categories.	char(TBD)	FALSE	FALSE	FALSE
wavelength-Min	The minimum wavelength covered in the data.	float	FALSE	FALSE	FALSE
wavelength-Max	The maximum wavelength covered in the data.	float	FALSE	FALSE	FALSE
wavelength-Central	The central wavelength covered in the data.	float	FALSE	FALSE	FALSE
spectralReso-lution	The spectral resolution of the data.	float	FALSE	FALSE	FALSE
AOSystem	Indicates which adaptive optics system is active. This should be on of "ALTAIR", or "MCAO", or NULL (No adaptive optics).	char(10)	FALSE	FALSE	FALSE
numMembers	This is the number of member data supersets in this data superset. If this is not an associated data superset, this value should be 1.	integer	TRUE	FALSE	FALSE
association-Type	This is the type of the association.	char(20)	FALSE	FALSE	FALSE
raMax	The maximum right ascension in the field of view.	integer	TRUE	FALSE	FALSE
raMin	The minimum right ascension in the field of view.	integer	TRUE	FALSE	FALSE
decMax	The maximum declination in the field of view.	integer	TRUE	FALSE	FALSE
decMin	The minimum declination in the field of view.	integer	TRUE	FALSE	FALSE
timestamp	This value is automatically incremented when rows in the science table are incremented or updated.	timestamp	TRUE	FALSE	FALSE

where it is necessary to associate a derived parameter value with the recipe instance which produced the value.

3.4 Observation tables

These tables contain an archive independent view of the Gemini archive content. These tables will also be used by other archives at the archive center, allowing cross archive searches.

FIGURE 9. Observation table relationships



3.4.1 Table: observationOverlap

This table is used to facilitate a self join on the observations table. This table identifies data supersets with overlapping fields of view. The table will be used to efficiently process queries such as "give me all observations of type a, where there is at least one overlapping observation of type b".

This table will either contain one row for each pair of overlapping data supersets, or it will contain two rows for each pair of overlapping datasets. If only one row is used for each pair, an index must be created for both datasets in the row, and a query to the table must check for a dataset in both sides of the pair.

Note that since this table and the observation table are not archive specific, this allows cross-archive searches.

The columns of table observationOverlap are shown in Table 19 on page 63.

3.4.2 Table: observations

This table provides a telescope independent view of the Gemini observations. Observational parameters will be transformed into image content and image quality descriptions, which will allow archive users to locate observations based on telescope independent parameters.

This table will be shared with all archives at the archive center, allowing cross archive searches.

TABLE 19. Columns of table observationOverlap

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
dataSuper-setId	The Id of the first data superset in the pair.	binary(8)	TRUE	TRUE	FALSE
overlap	The amount of the first dataset which is overlapped by the second dataset.	tinyint	TRUE	FALSE	FALSE
dataSupersetId2	The second data superset in the pair.	binary(8)	TRUE	FALSE	TRUE
overlap2	The amount of the second dataset which is overlapped by the first dataset.	tinyint	TRUE	FALSE	FALSE

The columns of table observations are shown in Table 20 on page 63.

TABLE 20. Columns of table observations

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
dataSuper-setId	The unique identifier for this data superset.	binary(8)	TRUE	TRUE	TRUE
ra	Right ascension in milliseconds of arc.	integer	TRUE	FALSE	FALSE
dec	Declination in milliseconds of arc.	integer	TRUE	FALSE	FALSE
raMax	Maximum right ascension in the field of view, in milli-seconds of arc.	integer	TRUE	FALSE	FALSE
raMin	Minimum right ascension in the field of view, in milli-seconds of arc.	integer	TRUE	FALSE	FALSE
decMax	Maximum declination in the field of view, in millisec-onds of arc.	integer	TRUE	FALSE	FALSE
decMin	Minimum declination in the field of view, in millisec-onds of arc.	integer	TRUE	FALSE	FALSE
lii	Galactic longitude.	float	TRUE	FALSE	FALSE
bii	The galactic latitude.	float	TRUE	FALSE	FALSE
spacialReso-lution	The spacial resolution of this image in seconds of arc.	float	TRUE	FALSE	FALSE
wavelength-Central	The central wavelength of the observations.	float	FALSE	FALSE	FALSE
wavelength-Min	The minimum wavelength usefully covered by the observation, in angstroms.	float	TRUE	FALSE	FALSE
wavelength-Max	The maximum wavelength usefully covered by the observation, in angstroms.	float	TRUE	FALSE	FALSE
magSatura-tion	The magnitude at which point sources are saturated.	float	FALSE	FALSE	FALSE
magSN10	The magnitude at which the signal to noise ratio is 10.	float	FALSE	FALSE	FALSE
magSN1	The magnitude at which the signal to noise ratio is 1.	float	FALSE	FALSE	FALSE

TABLE 20. Columns of table observations

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
fwhm	The full width half maximum in arc seconds.	float	FALSE	FALSE	FALSE
psf	An estimate of the point spread function.	<Unde-fined>	FALSE	FALSE	FALSE
skyBright-ness	The sky brightness, in magnitude / (arc sec **2).	float	FALSE	FALSE	FALSE
noise	The sky noise.	float	FALSE	FALSE	FALSE
numPoint-Source	The number of point sources visible in the observation.	integer	FALSE	FALSE	FALSE
numExtend-edSource	The number of extended sources visible in the observa-tion,	integer	FALSE	FALSE	FALSE
randEG	This is true if the data contains a random sample of extra galactic objects.	boolean	FALSE	FALSE	FALSE
extinction	This is an estimate of atmospheric extinction at the cen-tral wavelength.	float	FALSE	FALSE	FALSE
magnitu-deCorrection	A correction factor for converting observed magnitudes into apparent magnatudes.	float	FALSE	FALSE	FALSE
magnitu-deCorrection-Error	An estimate of the error in the magnitude correction value.	float	FALSE	FALSE	FALSE
type	The type of the observation. See the text for a descrip-tion of observation types.	char(20)	FALSE	FALSE	FALSE

For images, the spacialResolution column would be the pixel size. For one dimensional spectra, the spacialResolution column would be ??. For two dimensional spectra, the spacialResolution column would contain the pixel size. All resolutions are measured in arc seconds.

The type column indicates the type of the observation. The types of observations foreseen for Gemini are:

Image — The observation is an image observation.

1D Spectrum — The observation is a two dimensional spectrum.

2D Spectrum — The observation is a two dimensional spectrum.

Field spectrum — The observation is a data cube containing a spectrum of a field.

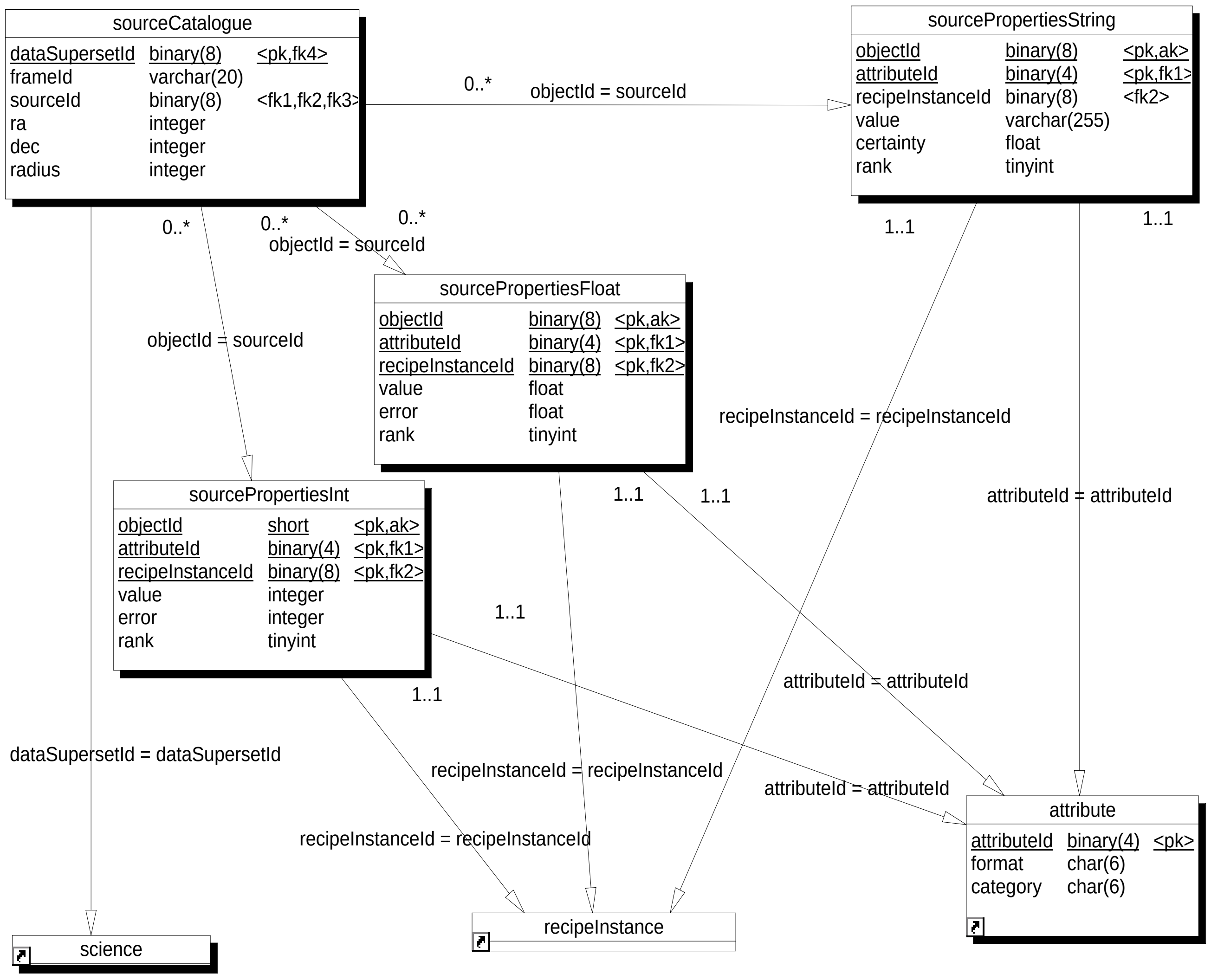
3.5 Source tables

The source tables contain information about each source detected in the datasets. These tables will also contain data extracted from the SIMBAD catalogues for sources which should be seen in the data.

3.5.1 Table: sourceCatalogue

This table contains a list of sources detected in each data superset. The sources will be detected by processing data with programs such as sextractor and galfit, and inserting the results into these

FIGURE 10. Source catalogue relationships


sourceCatalogue
dataSupersetId binary(8) <pk,fk4>
frameId varchar(20)
sourceId binary(8) <fk1,fk2,fk3>
ra integer
dec integer
radius integer
sourcePropertiesString
objectId binary(8) <pk,ak>
attributeId binary(4) <pk,fk1>
recipeInstanceId binary(8) <fk2>
value varchar(255)
certainty float
rank tinyint
sourcePropertiesFloat
objectId binary(8) <pk,ak>
attributeId binary(4) <pk,fk1>
recipeInstanceId binary(8) <pk,fk2>
value float
error float
rank tinyint
sourcePropertiesInt
objectId short <pk,ak>
attributeId binary(4) <pk,fk1>
recipeInstanceId binary(8) <pk,fk2>
value integer
error integer
rank tinyint
attribute
attributeId binary(4) <pk>
format char(6)
category char(6)
science
recipeInstance
0..*
objectId = sourceId
1..1
recipeInstanceId = recipeInstanceId
attributeId = attributeId
dataSupersetId = dataSupersetId

tables. The sources in these catalogues will not be correlated with sources detected in other datasets.

The columns of table sourceCatalogue are shown in Table 21 on page 65.

TABLE 21. Columns of table sourceCatalogue

Name	Comment	Data Type	Mandatory	Primary	Foreign Key
dataSupersetId	The data superset on which the source was identified.	binary(8)	TRUE	TRUE	TRUE
frameId	The frame in which the source was found.	varchar(20)	FALSE	FALSE	FALSE
sourceId	The identifier for the source.	binary(8)	TRUE	FALSE	TRUE

TABLE 21. Columns of table sourceCatalogue

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
ra	The right ascension for the source in milliseconds of arc.	integer	TRUE	FALSE	FALSE
dec	The declination of the source in milliseconds of arc.	integer	TRUE	FALSE	FALSE
radius	The radius of the source in milliseconds of arc.	integer	TRUE	FALSE	FALSE

3.5.2 Table: sourcePropertiesFloat

This table contains the floating point source properties. This table has the same structure as the datasetPropertiesFloat table described in Section 3.2.2 on page 55.

The columns of table sourcePropertiesFloat are shown in Table 22 on page 66.

TABLE 22. Columns of table sourcePropertiesFloat

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
objectId	This is a unique source id assigned to this source.	binary(8)	TRUE	TRUE	FALSE
attributeId	This is the attribute id for this attribute.	binary(4)	TRUE	TRUE	TRUE
recipeInstan-ceId	This identifies the recipe which produced this value.	binary(8)	TRUE	TRUE	TRUE
value	This is the value for this attribute.	float	TRUE	FALSE	FALSE
error	The error associated with this attribute.	float	TRUE	FALSE	FALSE
rank	The rank of this value relative to other values for the same attribute.	tinyint	TRUE	FALSE	FALSE

3.5.3 Table: sourcePropertiesInt

This table contains the source properties with integer values. This table has the same structure as the datasetPropertiesInt table described in Section 3.2.3 on page 56.

The columns of table sourcePropertiesInt are shown in Table 23 on page 66.

TABLE 23. Columns of table sourcePropertiesInt

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
objectId	The unique object id assigned to this object.	short	TRUE	TRUE	FALSE
attributeId	The attribute id.	binary(4)	TRUE	TRUE	TRUE
recipeInstan-ceId	This identifies the recipe instance which produced this value.	binary(8)	TRUE	TRUE	TRUE
value	The attribute's value.	integer	TRUE	FALSE	FALSE

TABLE 23. Columns of table sourcePropertiesInt

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
error	The error associated with this value.	integer	TRUE	FALSE	FALSE
rank	The rank of this value relative to other values for this attribute.	tinyint	TRUE	FALSE	FALSE

3.5.4 Table: sourcePropertiesString

This table contains string source properties. This table has the same structure as the datasetPropertiesString table described in Section 3.2.4 on page 56.

The columns of table sourcePropertiesString are shown in Table 24 on page 67.

TABLE 24. Columns of table sourcePropertiesString

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
objectId	The unique identifier for each object.	binary(8)	TRUE	TRUE	FALSE
attributeId	The attribute id.	binary(4)	TRUE	TRUE	TRUE
recipeInstan-ceId	This is the unique identifier of the recipe instance which generated the value	binary(8)	FALSE	FALSE	TRUE
value	The attribute's values.	var-char(255)	TRUE	FALSE	FALSE
certainty	This is an estimate of the degree of certainty for this value.	float	TRUE	FALSE	FALSE
rank	The rank for this value relative to other values for this attribute.	tinyint	TRUE	FALSE	FALSE

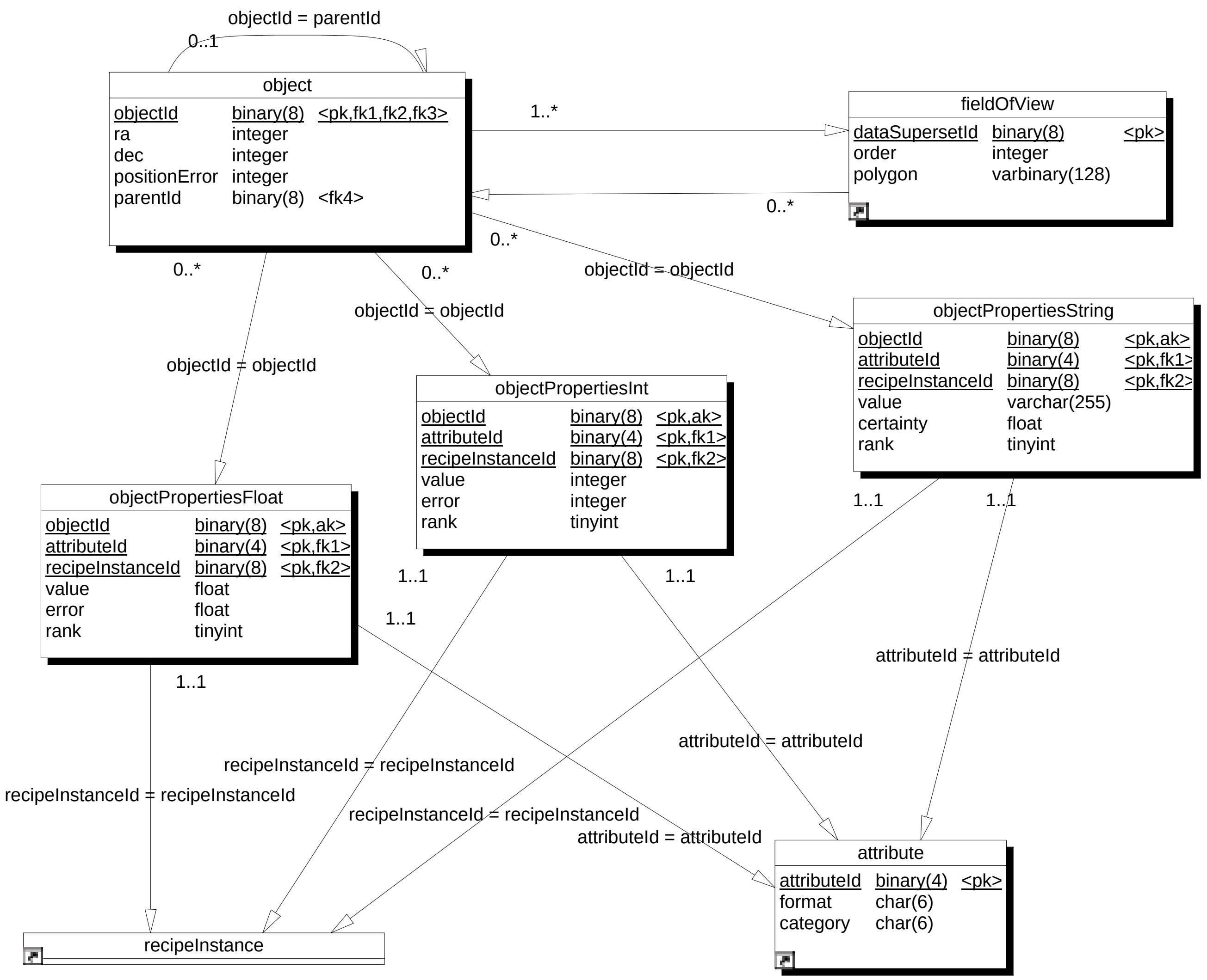
In addition to being used as a join value within the data processing catalogue tables, the recipe instance ID may be used as an identifier in other parts of the GSA, whenever it is necessary to identify specific recipe instances. One example of this would be in the GSA science catalogues, where it is necessary to associate a derived parameter value with the recipe instance which produced the value.

3.6 Object Tables

The object tables contain information about objects identified in the source catalogues. The attribute values stored in these tables will generally be identical to attribute values stored in the source table, except that the attributes will be associated with objects in the universe, and not with sources in a data superset.

The cases where attributes are not directly copied from the source tables are those where new attributes can be derived based on multiple observations of an object, which could not be derived from individual observations. One example of this situation would be variable objects - several individual data supersets may give values for the magnitude of an object, which when taken together lead to the conclusion that the object is variable. Attribute values could then be calculated which described the variability.

FIGURE 11. Object table relationships


objectId = parentId
0..1
object
objectId binary(8) <pk,fk1,fk2,fk3>
ra integer
dec integer
positionError integer
parentId binary(8) <fk4>
1..*
fieldOfView
dataSupersetId binary(8) <pk>
order integer
polygon varbinary(128)
0..*
0..*
0..*
0..*
objectId = objectId
objectId = objectId
objectId = objectId
objectPropertiesString
objectId binary(8) <pk,ak>
attributeId binary(4) <pk,fk1>
recipeInstanceId binary(8) <pk,fk2>
value varchar(255)
certainty float
rank tinyint
objectPropertiesInt
objectId binary(8) <pk,ak>
attributeId binary(4) <pk,fk1>
recipeInstanceId binary(8) <pk,fk2>
value integer
error integer
rank tinyint
objectPropertiesFloat
objectId binary(8) <pk,ak>
attributeId binary(4) <pk,fk1>
recipeInstanceId binary(8) <pk,fk2>
value float
error float
rank tinyint
1..1
1..1
1..1
1..1
1..1
1..1
attributeId = attributeId
attributeId = attributeId
attributeId = attributeId
recipeInstanceId = recipeInstanceId
recipeInstanceId = recipeInstanceId
recipeInstanceId = recipeInstanceId
attribute
attributeId binary(4) <pk>
format char(6)
category char(6)
recipeInstance

3.6.1 Table: object

This table contains information about each object known to the archive. The information in this table will be generated by data processing tasks, including tasks which query other object catalogues, such as Simbad and NED. This table is shared among all archives stored at the archiving center.

The columns of table object are shown in Table 25 on page 69.

The objects in the object catalogue are hierarchical. This heirarchical structure matches the physical structure of the universe. The parent column is used to indicate the position of the object in this hierarchy, for example star's parent might be a stellar cluster, the stellar cluster's parent might be a galaxy, and a galaxy's parent might be a cluster of galaxies, etc.

TABLE 25. Columns of table object

Name	Comment	Data Type	Mandatory	Primary	Foreign Key
objectId	This is a unique identifier assigned to each object.	binary(8)	TRUE	TRUE	TRUE
ra	Right ascension of the object in milliseconds of arc.	integer	TRUE	FALSE	FALSE
dec	Declination of the object in milliseconds of arc.	integer	TRUE	FALSE	FALSE
positionError	Positional error in milliseconds of arc.	integer	TRUE	FALSE	FALSE
parentId	The parent object of this object. See the text for a description of the object heirarchy.	binary(8)	FALSE	FALSE	TRUE

3.6.2 Table: objectPropertiesFloat

This table stores the floating point object properties. This table has the same structure as the datasetPropertiesFloat table described in Section 3.2.2 on page 55.

The columns of table objectPropertiesFloat are shown in Table 26 on page 69.

TABLE 26. Columns of table objectPropertiesFloat

Name	Comment	Data Type	Mandatory	Primary	Foreign Key
objectId	The unique identifier for this object.	binary(8)	TRUE	TRUE	FALSE
attributeId	The unique identifier for this attribute.	binary(4)	TRUE	TRUE	TRUE
recipeInstanceId	The unique identifier of the recipe instance which produced this value.	binary(8)	TRUE	TRUE	TRUE
value	The value of this attribute.	float	TRUE	FALSE	FALSE
error	The error associated with this attribute's value.	float	TRUE	FALSE	FALSE
rank	The rank of this value relative other values for the same attribute.	tinyint	TRUE	FALSE	FALSE

3.6.3 Table: objectPropertiesInt

This table stores the integer object properties. The structure of this table is identical to the datasetPropertiesInt table described in Section 3.2.2 on page 55.

The columns of table objectPropertiesInt are shown in Table 27 on page 69.

TABLE 27. Columns of table objectPropertiesInt

Name	Comment	Data Type	Mandatory	Primary	Foreign Key
objectId	This is the unique identifier assigned to each object.	binary(8)	TRUE	TRUE	FALSE
attributeId	This is the attribute id.	binary(4)	TRUE	TRUE	TRUE
recipeInstanceId	This identifies the recipe instance which produced this value.	binary(8)	TRUE	TRUE	TRUE
value	The value of the attribute.	integer	TRUE	FALSE	FALSE

TABLE 27. Columns of table objectPropertiesInt

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
error	The error associated with this value.	integer	TRUE	FALSE	FALSE
rank	The rank of this value relative to other known values for this attribute.	tinyint	TRUE	FALSE	FALSE

3.6.4 Table: objectPropertiesString

This table contains string values for object properties. The structure of this table is identical to the datasetPropertiesString table described in Section 3.2.4 on page 56.

The columns of table objectPropertiesString are shown in Table 28 on page 70.

TABLE 28. Columns of table objectPropertiesString

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
objectId	The unique object id.	binary(8)	TRUE	TRUE	FALSE
attributeId	The attribute id.	binary(4)	TRUE	TRUE	TRUE
recipeInstan-ceId	The recipe instance which produced this value.	binary(8)	TRUE	TRUE	TRUE
value	The value of the attribute.	var-char(255)	TRUE	FALSE	FALSE
certainty	This is an estimate of the degree of certainty for this value.	float	TRUE	FALSE	FALSE
rank	The rank of this value relative to other values for the attribute.	tinyint	TRUE	FALSE	FALSE

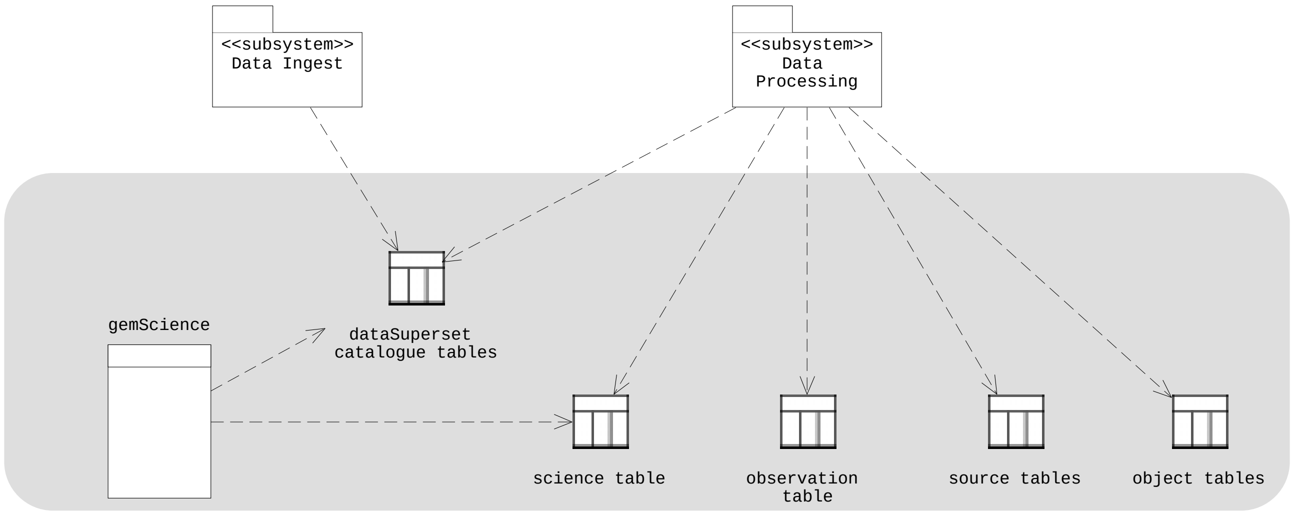
4. Catalogue maintenance

The maintenance of the data in the catalogues will be done by a variety of tasks described in the following sections. An overview of the catalogue subsystem is shown in Figure 12 on page 71. The shaded area indicates the components which are a part of the catalogue subsystem. The data flow through the catalogue tables is shown in Figure 13 on page 72.

4.1 Data superset catalogue

The data superset catalogue will be maintained by the Data Processing Discovery Agents described in Chapter 5, section 4.3 on page 89, and by the Data Ingest task described in Chapter 6, section 4.1 on page 97. The Data Ingest task will add simple datasets to the data superset table, and the Data Processing Discovery Agent tasks will identify associations and insert them into these tables. Some of the information destined for these tables may be produced by further data processing.

FIGURE 12. Catalogue Subsystem Overview



4.2 Science table

The science table will be maintained by the gemScience program. This program will examine the data superset catalogue created by the GSA Data Ingest subsystem, and the data processing subsystem. The program will have rules to determine which data supersets sets are “science” data, and will move these datasets into the science table. This program will be run periodically (daily) to update the science catalogue by the GSA data processing system. The data in the science table is always public, and so this table will be populated as soon as Gemini inserts catalogue data into the meta-data store.

The rules for determining when a data superset is useful for science will be developed in cooperation with Gemini. The rules used by gemScience will by similar to the rules used by other archives at the CADC. Some examples of the rules used by this program would be:

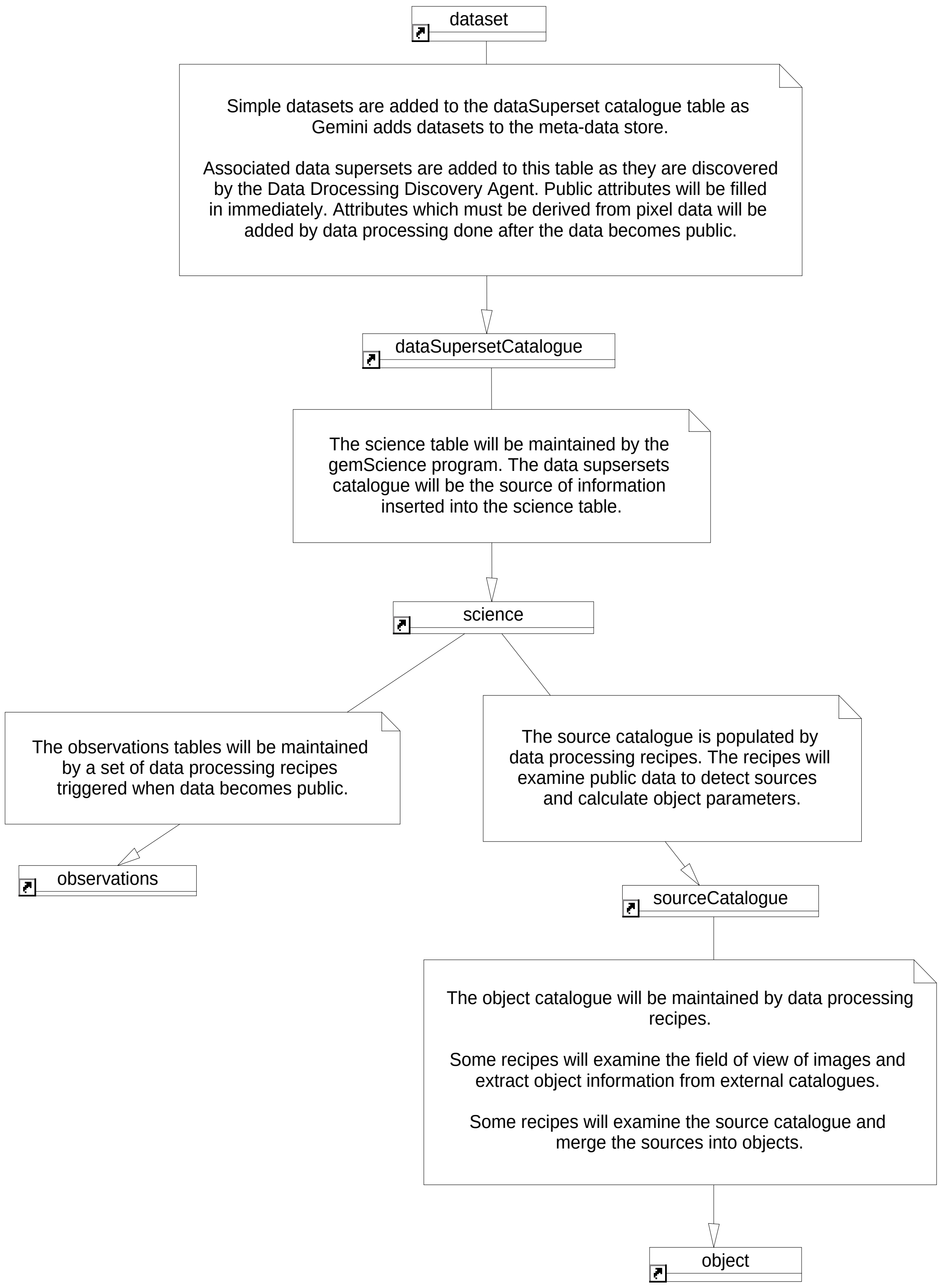
- Examining observation type and observing program id to eliminate calibration data.
- Checking for the existence of associations which should replace member datasets.
- Checking data quality to remove data which is not scientifically useful.

In addition to the above rules, new rules will be created as experience is gained with the Gemini data, and with the content of the science table.

4.3 Observations table

The observations table will be maintained by a set of data processing recipes. These recipes will be executed by the data processing system as each dataset becomes public. The recipes may be instrument specific, and will calculate the data to be inserted into the observations table based on information in the science table, and when necessary, based on the pixel data. As each row is inserted into the observations table, the recipes will search for any other observations with overlapping fields of view, and insert appropriate rows into the observationOverlap table.

FIGURE 13. Data flow through the catalogue tables


dataset
Simple datasets are added to the dataSuperset catalogue table as Gemini adds datasets to the meta-data store.
Associated data supersets are added to this table as they are discovered by the Data Drocessing Discovery Agent. Public attributes will be filled in immediately. Attributes which must be derived from pixel data will be added by data processing done after the data becomes public.
dataSupersetCatalogue
The science table will be maintained by the gemScience program. The data supsersets catalogue will be the source of information inserted into the science table.
science
The observations tables will be maintained by a set of data processing recipes triggered when data becomes public.
The source catalogue is populated by data processing recipes. The recipes will examine public data to detect sources and calculate object parameters.
observations
sourceCatalogue
The object catalogue will be maintained by data processing recipes.
Some recipes will examine the field of view of images and extract object information from external catalogues.
Some recipes will examine the source catalogue and merge the sources into objects.
object

4.4 Source catalogue

The source catalogue will be maintained by a set of data processing recipes. These recipes will be executed as each dataset becomes public. The recipes will examine the calibrated pixel data of the science observations, and produce source parameters based on the calculations. These recipes will be data class specific (i.e. some will process spectral data, others will process image data), but will probably not be instrument specific. Initially, the recipes will execute the sextractor program to detect sources [9], and the galfit program to determine the galaxy morphology [10]. The recipes used to populate the source catalogue will be expanded over the life of the archive.

4.5 Object catalogue

The object catalogue will be maintained by a set of data processing recipes. There will be two categories of recipes to maintain the object catalogue:

- Recipes which examine the science table and extract objects from other catalogues (Simbad and NED) which are in each observation's field of view. These recipes will only use field of view information which is always public, and so the data processing system can execute these recipes as soon as datasets are inserted into the science table.
- Recipes which examine the content of the source catalogue, merge attributes which describe the same object into a single set of attribute-values describing the object, and rank the various values for each attribute. These recipes will be dependant on the source catalogue content, which is in turn dependant on the pixel data being public.

Each of the data processing recipes will use a shared object identification algorithm to determine which objects are the same, and which objects are different.

Each of the data processing recipes will use a shared attribute ranking algorithm to determine the relative goodness of attributes taken from various sources.

The object identification algorithm is a challenging task which may not be available with the initial release of the system.

5. Hardware Requirements

This sections describes the hardware needed to support the GSA catalogues.

5.1 Estimated Storage Needs

The storage space requirements of the significant users of database disk space are described in the following sections. The total storage space needs for the catalogues will be between 44 and 68 Gigabytes per year.

5.1.1 Data superset catalogue

TBD

5.1.2 Science table

Each row in the science table will require about 500 bytes of data, including index space. The estimated number of datasets per year given in [7] is between 84000 and 134000. Based on past experience with other archives, approximately half of these datasets will be classified as science datasets. This results in 40 to 65 Megabytes per year of data for the science table.

5.1.3 Preview table

The preview table will contain one row for each data superset in the science table, and so will grow by between 84000 and 134000 rows per year. If the average size of a preview image is 300 Kbytes, preview will need between 24 and 38 Gigabytes per year.

5.1.4 Observations table

Each row in the observations table will require about 200 bytes, including index space. There will be one row in the observations table for each public data superset in the science table. This will result in between 16 and 26 Megabytes per year of data for the observations table.

5.1.5 Source catalogue

The source catalogue consists of a source table, and a set of three property tables. It is difficult to estimate the number of sources per dataset since there is a wide range of instrument types in use at Gemini. These calculations assume there will be an average of 200 sources detected in each dataset, and each source will have an average of 20 properties associated with it. This results in between 8,400,000 and 13,400,000 sources per year, and between 168,000,000 and 268,000,000 properties measured each year.

Each row in the source table will required about 70 bytes of data, including index space. This will result in between 560 and 900 Megabytes of storage per year.

Each row in a source properties table will require about 60 byte of data, including index space. This will result in between 9,600 and 15,000 Megabytes of storage per year.

5.1.6 Object catalogue

The object catalogue will be similar to the source catalogue, with an object table and a set of three property tables. The size of the object tables will be similar to the source table, with the exception that sources will be merged into objects. Eventually the object table may contain significantly fewer entries than the source table, as objects are re-observed, but this is difficult to estimate. The object properties tables will be the same size as the source properties tables, since all source properties will be included in the object properties tables (the rank of duplicated properties will be assigned based on the relative reliability of the values). The upper limit on the size of the object catalogue tables will be the size of the source catalogue tables, or between 560 and 900 Megabytes per year for the object table, and between 9,600 and 15,000 Megabytes per year for the properties tables.

5.2 Database server requirements

For most of the queries required to support the GSA catalogues, the existing CADC Sybase systems will be adequate. The queries needed to support the source and object tables can be much more difficult to execute, but our current thinking is that they can be formulated in a way which will allow them to be executed in a reasonable time on our existing hardware.

The GSA will provide fractional support for ongoing maintenance of the CADC database system. The details of this support will be determined in the proposal for operating the GSA

Created:July 17, 2000
Modified:May 21, 2002

Gemini Science Archive

Chapter 5 Data Processing

1. Introduction

This chapter describes the conceptual design of the GSA Data Processing subsystem.

The basic data processing tasks done by the data processing subsystem are:

- Data processing for user requests, including initiating media creation when necessary.
- Generation of previews for datasets and data supersets.
- Identification of associations and the appropriate data processing for the associated data.
- Generation of image descriptors for datasets and data supersets.
- Extraction of object descriptors from SIMBAD and NED.
- Update of object descriptors from SIMBAD and NED.
- Generation of object descriptors from datasets and data supersets.

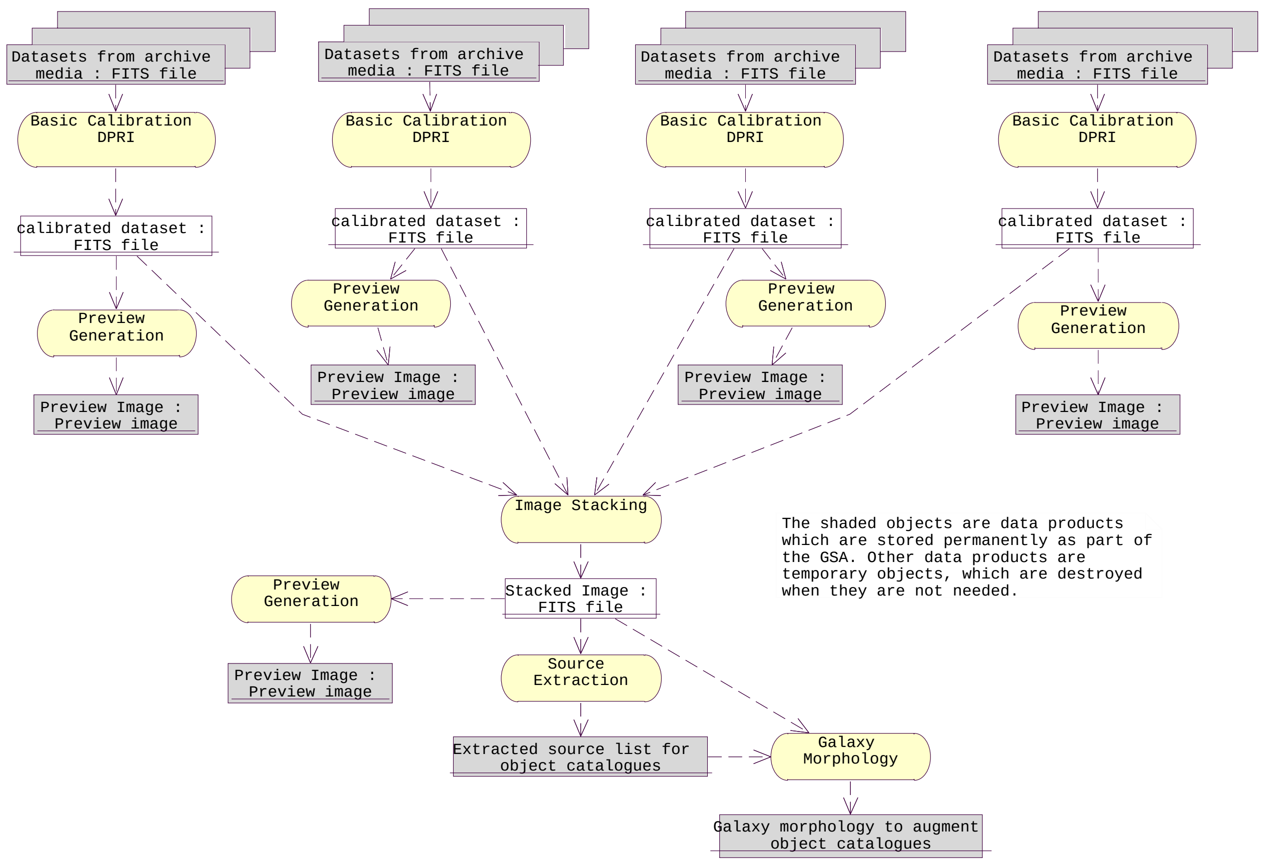
Data processing will be done by data processing recipes (DPRs), which are a set of general rules for processing data. A data processing recipe will consist of an executable process to do the processing, and a set of rules for identifying situations where and when the executable process could be used. The executable processes will make heavy use of the IRAF package developed by Gemini, but will also include tasks from other sources. A few simplified examples of DPRs might be:

- Given a GMOS science dataset, if there is a bias and flat dataset available (selected by comparing the meta-data stored in the catalogues), then a flat/bias correction executable process can be run. The process would probably use IRAF to do the processing, and may use IRAF tasks developed by Gemini. These tasks should be executed only when the output is needed for an archive user data request, or when the output is needed by another task.
- Given a GMOS spectral mode dataset which has been flat and bias corrected, and a mask definition file, a spectral extraction recipe can be run. The process would use IRAF tasks developed by Gemini to do the actual spectral extraction. These tasks should only be executed when the output is needed for an archive user data request, or when the output is needed by another task.
- Given a set of “compatible” flat/bias corrected GMOS image mode datasets, a task can be run to stack the datasets into an associated data superset. The “compatibility” of the datasets would be determined from the meta-data stored in the catalogues. These tasks should only be executed when the output is needed for an archive user data request, or when the output is needed by another task.
- Given a stacked GMOS image mode dataset, a task can be executed to identify sources on the image, and insert the list of detected sources into the GSA’s object catalogue. These tasks should be queued for execution when all of the input data become public.

Data processing recipes are heirachical, and to encourage re-use of recipe components, should be done as a series of simple steps, instead of as complex monolithic processes. In the above examples, the flat/bias correction recipes might supply data to be delivered to an archive user, or they might supply input data to the stacking recipe. It is also probable that the output of some Data Processing Recipe Instances (DPRIs) could be used as input by more than one DPRI. In order to optimize resource usage, the relationships between DPRIs must be taken into account when scheduling execution of DPRIs.

Figure 14 on page 76 shows the data flow through the DPRIs for an example of an image data processing sequence. See Chapter 1, section 9.1.1 on page 13 for a description of the notation used for object flow diagrams. In this example, datasets from the archive are calibrated, and then previews are generated. The calibrated datasets are then stacked to form an associated data superset, and a preview is generated for the association. The stacked image is then processed with a source extraction task, and then with a galaxy morphology task.

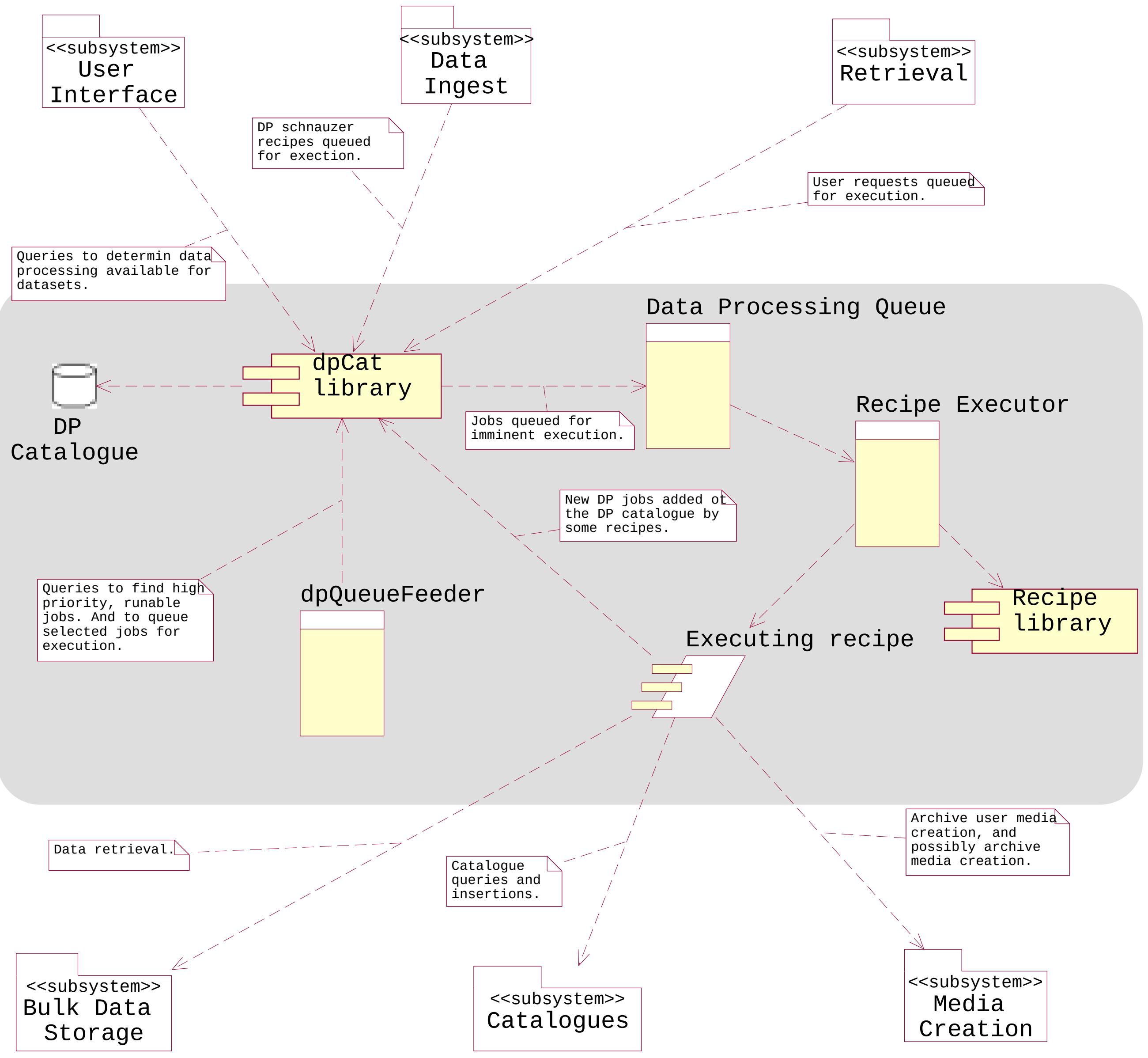
FIGURE 14. Data processing sequence example


Datasets from archive media : FITS file
Basic Calibration DPRI
calibrated dataset : FITS file
Preview Generation
Preview Image : Preview image
Image Stacking
Stacked Image : FITS file
Source Extraction
Extracted source list for object catalogues
Galaxy Morphology
Galaxy morphology to augment object catalogues
The shaded objects are data products which are stored permanently as part of the GSA. Other data products are temporary objects, which are destroyed when they are not needed.

As data arrives at the GSA, DPRIs will be identified and stored in the data processing catalogue (see Section 3.1 on page 80). DPRIs are specific instances of DPRs, for example given the flat/bias correction recipe in the above examples, a corresponding DPRI would contain the information that the flat/bias task can be run with dataset *x* as the science dataset, dataset *y* as the flat field dataset, and dataset *z* as the bias dataset.

Figure 15 on page 77 shows a component diagram describing the components of the Data Processing subsystem as they relate to each other and to the other subsystems of the GSA. See Chapter 1, section 9.1.2 on page 13 for a description of component diagrams. The shaded area denotes the components which are part of the Data Processing subsystem.

FIGURE 15. Data Processing subsystem overview


<<subsystem>> User Interface
<<subsystem>> Data Ingest
<<subsystem>> Retrieval
DP schnauzer recipes queued for exection.
User requests queued for execution.
Queries to determin data processing available for datasets.
Data Processing Queue
dpCat library
DP Catalogue
Jobs queued for imminent execution.
Recipe Executor
New DP jobs added ot the DP catalogue by some recipes.
Queries to find high priority, runable jobs. And to queue selected jobs for execution.
dpQueueFeeder
Recipe library
Executing recipe
Data retrieval.
Archive user media creation, and possibly archive media creation.
Catalogue queries and insertions.
<<subsystem>> Bulk Data Storage
<<subsystem>> Catalogues
<<subsystem>> Media Creation

The data processing system is based on two prototype systems developed at the CADC, and the design of the DHS Data Processing system:

- A prototype was developed which demonstrates the functionality of the DP catalogue library, and the DP catalogue database. The prototype was implemented for the CFHT and HST archives.

- A prototype demonstrates the functionality of the Data Processing queue and data processing executor.

Scheduling of DPRIs for execution will be triggered in several different ways:

- Some DPRIs will be executed on demand (data processing requested by archive users, or when the output of a recipe instance is required as input to another recipe instance that is being executed).
- Some DPRIs will become executable as soon as data is inserted into the GSA catalogues (identifying data supersets, or using WCS data from dataset headers to identify objects on the image based on the SIMBAD and NED catalogues). To ensure proprietary periods are respected, these recipes will only use public meta-data from the GSA catalogues as input.
- Some recipe instances will become executable when release date for the recipe is reached (preview generation and data mining).
- Some recipe instances will be executed periodically (e.g. checking for and propagating updates to the SIMBAD and NED catalogues).

Tasks will be prevented from running until all of their required inputs are available. Data processing recipe instances will only be executed before their release dates if the data is to be delivered to an archive user who has verified that he/she has permission to access all proprietary data used as input to the recipe.

2. Requirements on the GSA Data Processing Subsystem

Table 29 on page 78 lists the software requirements which at least partially apply to the GSA Data processing system. Each requirement is followed by a short description of how the requirement will affect the Data Processing subsystem. See [6] for a complete description of these requirements.

TABLE 29. User interface requirements

SR1.2 The GSA shall respect proprietary periods assigned by Gemini. Automatic data processing will not be initiated if it conflicts with proprietary periods. Data processing done for user requests will only use input data that the user is authorized to have.
SR1.3 The GSA shall be able to automatically process data. GSA will have a data processing system that will automatically process data for both user requests and for operational needs.
SR1.3.1 Data processing shall respect Gemini policy on proprietary periods. See SR1.2.
SR1.3.2 Data processing shall proceed promptly. The data processing system will have a priority queue, allowing user requests to be given priority. User requests will be processed in an order which prevents small user requests from being blocked by large requests. More than one user request can be processed at any given time.
SR1.4 Preview data should be generated for each dataset. The data processing system will be able to automatically process data as it becomes public, in order to create preview images. The data processing system will be able to insert previews into the preview catalogue. There will have to be recipes to generate preview images.

TABLE 29. User interface requirements

SR1.4.1 Generate previews for associations. See SR1.4.
SR1.4.2 Remove previews when release date is changed. The data processing system will have to figure out which data superset previews became proprietary, and must be able to cause preview to be removed from the catalogues.
SR2.3 The catalogue shall include accurate WCS information. DP recipes must preserve WCS information when appropriate.
SR2.5 The catalogue should include image descriptors derived from the data. The data processing system will be able to automatically process data as it becomes public. Data processing recipes must be able to augment the descriptors for a data superset. There must be data processing recipes to calculate the derived image descriptors.
SR2.5.1 Remove derived image descriptors for proprietary data. The data processing system will have to figure out which derived descriptors have become proprietary, and must cause the proprietary descriptors to be removed from the catalogue. See also SR1.4.2
SR2.7 The GSA catalogue should contain associated datasets. The data processing system will have to detect potential associations and changes to existing associations. The data processing system will have to run recipes to determine if the datasets really can be associated. The data processing recipes will be able to insert newly detected associations into the catalogues, and update existing associations. There must be recipes to verify associations.
SR2.13 Calibration data shall be associated with science data. As new associated data supersets are created, any appropriate calibration data must be identified and inserted into the catalogues.
SR2.14.1 Object parameters should be extracted from object catalogues. The DP system may be required to identify objects in external catalogues. If this is the case, the DP system would have to initiate the appropriate data processing recipes for each data superset, as soon as accurate WCS coordinates are available in the catalogue. There will have to be a recipe to extract objects from the external catalogues.
SR2.14.2 The GSA object catalogue should be up to date. If the DP system identifies objects in external catalogues (See SR2.14.1) then it will have to either detect changes in the external catalogues, and update the list of objects associated with data supersets in the GSA, or periodically re-generate the list of objects associated with each data superset.
SR2.14.3 The object catalogue should include object descriptors derived from data. The DP system will have to trigger data processing when data becomes public. Data processing recipes will have to be able to insert new objects into the object catalogues. There will have to be data processing recipes to do the object extraction.
SR2.14.4 Remove derived objects for proprietary data. The data processing system will have to determine which derived objects have become proprietary, and cause the proprietary derived objects to be removed from the object catalogue. See also SR1.4.2.
SR3.3.2 GSA users can view a list of calibration data supersets. The calibration database must be available through the user interface. The calibration database must be interactively searchable.
SR3.4.3 GSA users should be able to choose to have data processed. DP may be triggered by user requests. There will be recipes which are useful to archive users.
SR3.4.4 Data processing shall not use proprietary data as input. See SR1.2.

TABLE 29. User interface requirements

SR3.4.5 Processing recipes shall be documented. Recipes shall follow standards describing the kinds of documentation they will produce. The DP system will provide recipe documentation as part of user requests.
SR3.4.6 Authorized Gemini staff can retrieve proprietary data. The data processing system will allow authorized users to use proprietary data as input to data processing.
SR3.4.7 GSA users can retrieve calibration data. The calibration database is accessible from the user interface. The calibration database can return a list of calibration files required for any data superset.
SR3.4.12 It should be possible for users to monitor the progress of data requests. The DP system will provide status information allowing the progress of data processing to be monitored. The status information will be available to the user interface.
SR4.3 Maximum and average requirements for data retrieval by users. Data processing for user requests should be able to keep up with data requests. Data processing should process more than one request at a time. There may need to be more than one system dedicated to data processing.
SR4.6 Data for Internet retrieval should be available promptly. Data processing shouldn’t take too long.

3. Data Processing Infrastructure

This section describes the components of the GSA data processing infrastructure. This infrastructure is intended to be a shared system, which will be used by all archives at the CADC.

3.1 Data Processing Catalogue Database

The GSA Data Processing subsystem has a catalogue which contains a complete list of DPRIs which could be executed on Gemini data. As each dataset is received by the Data Ingest subsystem, a “Gemini Data Processing Discovery Agent” DPRI for the dataset will be inserted into the data processing catalogues, and flagged for immediate execution. When executed, the schnauzer recipe will identify all other data processing which could be done using the dataset. See the section describing the data processing schnauzer for more details. All DPRIs will be assigned a public release date which will be based on the maximum public release date of all inputs used by the DPRI.

This database will serve several purposes:

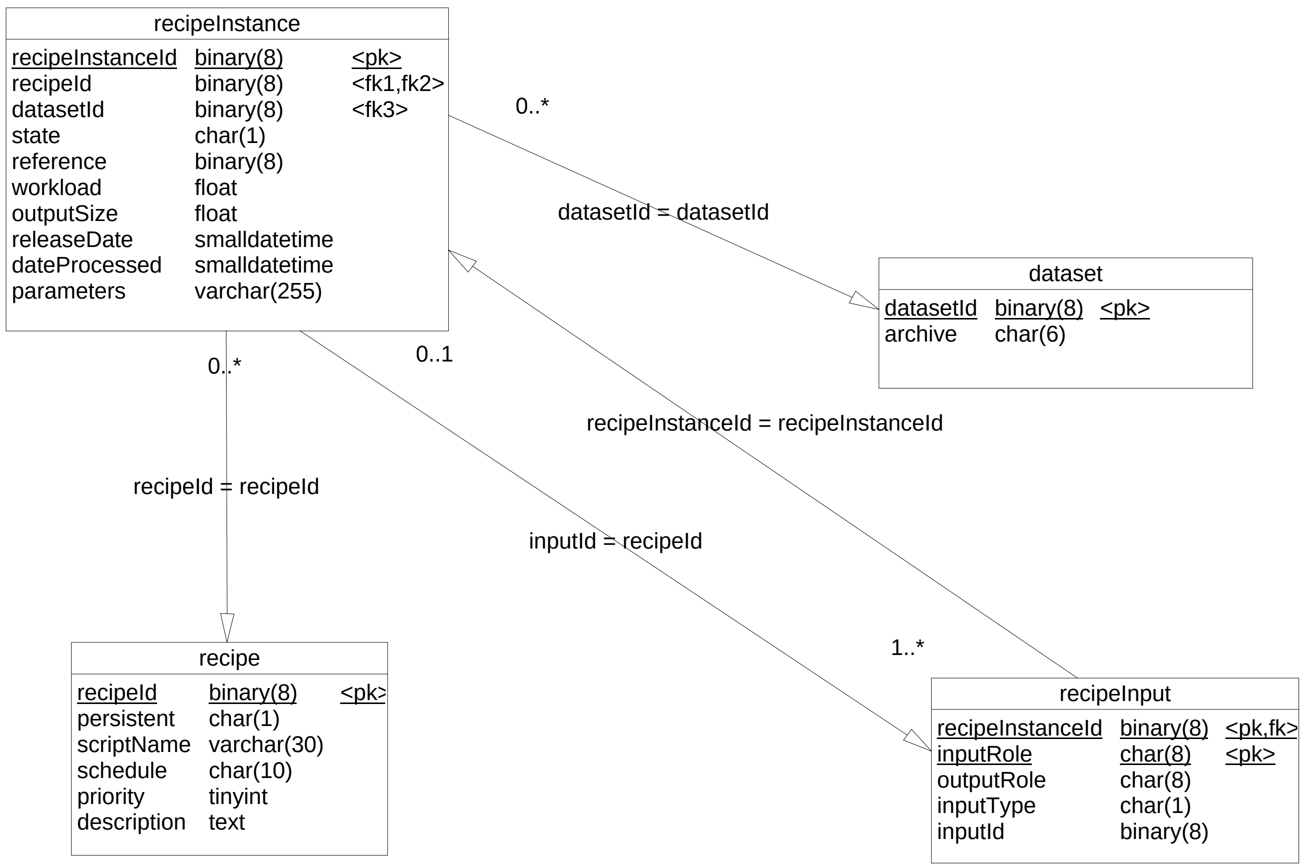
- The database will allow the user interface to determine which data processing recipes could be applied to any given data superset.
- This database will allow the creation of a processing dependancy tree which can be submitted to the Data Processing subsystem for exection.
- This database will allow tracking of the data processing done automatically by the archive centre.
- When recipes are modified, other recipes which use the recipe's output can be identified using this database, allowing recipes to be re-executed if necessary.

- When raw datasets are modified, recipes which use the raw dataset, either directly or indirectly can be identified using this database, allowing the recipes to be re-executed if necessary.

The data processing recipe instances will be stored in a set of SYBASE database tables in the data processing catalogue. The tables store information about data processing recipes, data processing recipe instances, and about the inputs used by the DPRIs. The purpose of the tables is explained in more detail in the following sections.

Most, but not all data processing will be described in the data processing catalogue. Some simple data processing may be created on-the-fly by other subsystems of the GSA. In particular, the Data Retrieval subsystem may create simple data processing steps to “package” data requests and inform users about the requests status, which would be wrapped around data processing recipes taken from the Data Processing Catalogues. These recipes would be inserted directly into the data processing queue, without ever being inserted into the data processing catalogue.

FIGURE 1. Data processing table relationships



3.1.1 Table: dataset

This table allows an archive independent mapping from dataset id to data superset name. The datasetId column is used to identify a dataset in the data processing tables, and the datasetName column identifies the data superset in the archive specific catalogues.

The columns of table dataset are shown in Table 30 on page 82.

TABLE 30. Columns of table dataset

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
datasetId	This identifies a dataset known to the data processing system.	binary(8)	TRUE	TRUE	FALSE
archive	The archive which contains the dataset.	char(6)	TRUE	FALSE	FALSE

The indexes of table dataset are shown in Table 31 on page 82.

TABLE 31. Indexes of table dataset

Name	Comment	Unique	Cluster
datasetName	This index allows dataset id to be determined given the archive and dataset name.	FALSE	FALSE
datasetId	This index allows dataset name and archive to be determined, given a dataset id.	FALSE	FALSE

3.1.2 Table: recipe

The recipe table contains one row for each recipe supported by the data processing system. The row contains information which describes the recipe, how it is executed, when it is executed, and a description of what the recipe does. Rows will be added to this table manually, as new recipes are implemented at the GSA.

This table will also be used as a source of information for the GSA user interface when a user requests documentation for a recipe.

The columns of table recipe are shown in Table 32 on page 82.

TABLE 32. Columns of table recipe

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
recipeId	This is a unique identifier for this recipe.	binary(8)	TRUE	TRUE	FALSE
persistent	This column indicates if the output of the recipe is persistent or not. See the description of persistent recipes.	char(1)	TRUE	FALSE	FALSE
scriptName	This is the name of the script to execute in order to initiate the recipe processing.	varchar(30)	TRUE	FALSE	FALSE
schedule	This indicates the schedule for running the recipe. See the description of recipe scheduling.	char(10)	TRUE	FALSE	FALSE
priority	This is the priority for the execution of this recipe. See the description of recipe priority.	tinyint	FALSE	FALSE	FALSE
description	This column contains a description of the recipe, usable for the GSA user interface. The description should be formatted in HTML.	text	FALSE	FALSE	FALSE

The indexes of table recipe are shown in Table 33 on page 83.

TABLE 33. Indexes of table recipe

Name	Comment	Unique	Cluster
recipeId	This index allows recipe information to be located given the recipe id. This index may not be necessary, since it is expected the recipe table will remain small.	TRUE	FALSE

3.1.3 Table: recipeInput

Each of the inputs to a recipe instance is either a dataset from the archive, or the output from another recipe instance. This table describes each of these inputs. The type of the input is indicated by the inputType field, and the specific input is identified by the inputId field. Dataset inputs are identified by joining the inputId field with the datasetId field of the dataset table. Recipe instance inputs are identified by joining the inputId field with the recipeInstanceId field of the recipe instance table. The inputRole field is used to identify how the input will be used by the DPRI. (E.g. if a DPR takes as input, an object dataset, a flat field dataset, and a bias dataset, the inputRole would specify which role (object, flat or bias) each input has in the processing.)

The columns of table recipeInput are shown in Table 34 on page 83.

TABLE 34. Columns of table recipeInput

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
recipeInstan-ceId	This identifies the recipe instance associated with this input. This value is used to join with the recipe instance table.	binary(8)	TRUE	TRUE	TRUE
inputRole	This is used to clarify how the recipe will use the input.	char(8)	TRUE	TRUE	FALSE
outputRole	Used to identify which part of an output should be used. For example which of the HST files associated with a dataset, which frames from a Gemini observation, which output from a recipe instance which produces multiple outputs.	char(8)	FALSE	FALSE	FALSE
inputType	This is the type of input. Options are “R” if the input is another recipe instance, or “D” if the input is a dataset.	char(1)	TRUE	FALSE	FALSE
inputId	The identifier of the input. If inputType is “R”, this is a recipeInstanceId used to join with the recipe instance table. If inputType is “D”, this is a dataset id, used to join with the dataset table.	binary(8)	TRUE	FALSE	FALSE

The indexes of table recipeInput are shown in Table 35 on page 84.

3.1.4 Table: recipeInstance

The recipe instance table has one row which describes each Data Processing Recipe Instance known to the data processing system.

TABLE 35. Indexes of table recipeInput

Name	Comment	Unique	Cluster
inputId	This index allows all recipes which use a given input to be located.	FALSE	FALSE
recipeInstanceId	This index allows the inputs used by a recipe instance to be identified.	FALSE	FALSE

The table contains information which describes the inputs to a recipe instance (via a join with the recipe input table), the output from a recipe instance, and the state of processing of the recipe instance.

The columns of table recipeInstance are shown in Table 36 on page 84.

TABLE 36. Columns of table recipeInstance

Name	Comment	Data Type	Mandatory	Primary	Foreign Key
recipeInstanceId	This is a unique identifier for this recipe instance, and is used when joining with the recipe input table. See the text for a description of the use of this value.	binary(8)	TRUE	TRUE	FALSE
recipeId	This is used to join with the recipe table, to identify the recipe associated with this recipe instance.	binary(8)	FALSE	FALSE	TRUE
datasetId	This identifies a dataset known to the data processing system.	binary(8)	FALSE	FALSE	TRUE
state	This is the state of the recipe execution. See the text for a list of possible states.	char(1)	TRUE	FALSE	FALSE
reference	Used to identify groups of recipe instances, for the Data Processing Discovery Agent.	binary(8)	TRUE	FALSE	FALSE
workload	This is a rough estimate of the amount of CPU time needed to execute this recipe step. This value will be used by the scheduler to determine the cost of executing the recipe instance.	float	TRUE	FALSE	FALSE
outputSize	This is a rough estimate of the size of the output created by this recipe, in Megabytes.	float	TRUE	FALSE	FALSE
releaseDate	This is the date when the output of the recipe becomes public. If NULL, the output is always public.	smalldatetime	FALSE	FALSE	FALSE
dateProcessed	This is the most recent date and time the recipe instance was executed. (The date is updated even if the recipe failed.)	smalldatetime	FALSE	FALSE	FALSE
parameters	This contains the parameters to be passed to the recipe script when the recipe is executed.	varchar(255)	FALSE	FALSE	FALSE

The dataset ID will be either will be either the only dataset used as input to the recipe instance, or an associated data superset created to represent a group of datasets. The only types of recipe instances which might not have a datasetId are those which do generic processing, for example examining all datasets on a night (Some Data Processing Discovery Agents may work like this).

In addition to being used as a join value within the data processing catalogue tables, the recipe instance ID may be used as an identifier in other parts of the GSA, whenever it is necessary to identify specific recipe instances. One example of this would be in the GSA science catalogues, where it is necessary to associate a derived parameter value with the recipe instance which produced the value.

The precise use of the reference field will be determined by each Data Processing Discovery Agent recipe. For example, one recipe may need to look at all data for a night as a unit, while another looks at all data for a science program. Each of the recipes could generate a reference id associated with its unit of work (eg. night number or science program ID), which would then be used to determine which existing recipe instances need to be taken into consideration when processing data.

The state column has the following possible values:

" "	The DPRI has never been executed.
"E"	Executing this DPRI resulted in an error that is expected to persist.
"Y"	The DPRI has been successfully executed at least once.
"Q"	The DPRI is queued for execution for the first time.
"R"	The DPRI is queue for re-execution. This state is only used when a recipe that produces persistent results is scheduled for re-execution. This state will probably only be set more-or less manually by archive staff (e.g. if a preview recipe is significantly changed, archive staff may decide that some or all of the associated DPRIs should be re-executed).

The indexes of table recipeInstance are shown in Table 37 on page 85.

TABLE 37. Indexes of table recipeInstance

Name	Comment	Unique	Cluster
recipeInstanceId	The recipeInstance id is used to join with the recipe input table and allows location of a recipe instance given its recipe instance id.	TRUE	TRUE
reference	This index is used by data processing schnauzers, to identify existing recipe instances.	FALSE	FALSE
datasetId	This index allows identification of recipe instances associated with a dataset.	FALSE	FALSE

3.2 Data Processing Catalogue Database Access

A library will provide access to the data processing catalogues. The library will provide the following functionality:

- Create a new data processing recipe instance in memory.
- Add new DPRIs to the catalogue.
- Add new datasets to the catalogue.
- Delete DPRIs from the catalogue.
- Update DPRI state.

- Identify DPRIs which are runable, ordered with highest priority jobs first. When several jobs have the same priority, the oldest jobs will be executed first.
- Identify data processing recipe instances associated with given a data superset. Given a data superset name, this function would:
  - return a recipe instance which re-calibrates the data superset.
  - return a recipe instance which creates a preview from the re-calibrated data.
  - not return a recipe instance which uses the data superset to create a super flat.
  - not return a recipe which stacks the calibrated dataset with several other datasets.
- Identify data processing recipes which use a specific data superset as input. This function is similar to the above function, except this function would return any recipe instance which uses the data superset as input, including those noted as not being returned by the above function.
- Add a DPRI to the data processing queue described in Section 3.3 on page 86.

3.3 Data Processing Queue

The data processing queue is populated by the Data Retrieval subsystem described in Chapter 7, and the queue feeder described in Section 3.4 on page 86. The queue stores DPRIs with associated priority and resource requirement descriptors and manages the distribution of DPRIs to multiple systems where they are executed by a Recipe Executor (see Section 3.5 on page 87). The queue also manages dependencies between tasks, provides the infrastructure for the flow of data from one task to the next within a processing tree, and ensures that DPRIs are only executed once all required input data is available. Finally, the queue ensures that tasks are executed in near-optimal order and in a capable environment by using priorities and matching resource requirements with the capabilities of Recipe Executor host systems.

The Data Processing Queue implementation is described in detail at http://cadcwww.hia.nrc.ca/software/distributed/.

3.4 Queue Feeder

The Data processing queue feeder monitors the data processing catalogue described in Section 3.1 on page 80, and the state of the data processing queue described in Section 3.3 on page 86. When the data processing queue is nearly out of tasks, the queue feeder extracts some new runable jobs from the data processing catalogue and inserts them into the execution queue.

As shown in Figure 14 on page 76, some executable DPRIs will require the output from other DPRIs before they can be executed, and the output of some DPRIs will be used as input by more than one DPRI. In these cases, it is possible to avoid wasting compute cycles by processing the whole structure as part of a single batch, and only executing each DPRI once. The single batch approach may conflict with the need to process higher priority jobs first. In the example in Figure 14, the galaxy morphology DPRI is a low priority task, and the cost of executing it might be measured in hours of CPU time, where all of the other DPRIs have much higher priority and the cost of executing them is measured in seconds of CPU time. In order to ensure the high priority jobs are executed first, the Queue feeder will have to use a heuristic algorithm for determining how to organize the queued data processing, taking into account the benefit of executing a DPRI, as measured by its priority (stored in the recipe table), and the cost of executing the DPRI, as measured by its predicted execution time (stored in the recipe instance table). In the example, it may be better to execute the preview generation and source extraction as one relatively quick high priority job, and leave the galaxy morphology step to execute as a separate low priority job, even though it would require re-executing the calibration and stacking DPRIs.

The data processing queue described in Section 3.3 on page 86 takes care of executing batches of DPRIs in the order which makes pseudo-optimal use of available resources, and making output of DPRIs available to the DPRIs which use the output.

The queue feeder will be aware of persistent recipes as described in Section 4.2 on page 89. If in the example in Figure 14 on page 76, the image stacking step is persistent, the processing tree created by the queue feeder must depend on whether the persistent stacked image is available or not, since if it is available, the data processing need not be repeated. This can be determined by examining the “persistent” flag in the recipe table, and the dateProcessed flag in the recipe instance table.

In the event that a persistent recipe has already been executed, the queue feeder will not queue any data processing needed as input to the recipe. The recipe itself will be queued and executed with the “-fromStore” flag set, and will be expected to retrieve its results from the persistent store (see Section 4.2 on page 89).

The recipe execution schedule is indicated by the **schedule** field in the recipe table. The possible values for the schedule field are:

- Demand: This recipe is executed when its output is needed as input to another recipe, or if its output is needed for delivery to a user.
- Public: A superset of demand where the recipe will also be automatically queued for execution when the data becomes public.
- Immediate: A superset of demand where the recipe will also be automatically queued for execution as soon as possible.

3.5 Recipe Executor

The Recipe Executor (RE) is a program which takes the highest priority available task in the data processing queue, executes it locally, and puts the results back into the queue system (for input into other tasks).

The RE must satisfy the following requirements:

- The RE is portable and able to run in a variety of common computing environments (computer hardware and operating systems). Executable code that is part of the task is either assumed to be installed on these systems or it is dynamically loaded over the network.
- There may be many REs running in parallel on different systems, making up a distributed computing system.
- RE host systems may have different capabilities (processing power, memory, local storage, network access, etc.).
- REs can be configured to only process high priority jobs, or only process jobs with certain resource requirements.
- If an RE fails due to a network outage, software crash, host system crash, or power failure it cannot cause the loss of data or tasks or permanently lock up resources or data structures in the data processing queue system.

The RE implementation is described at http://cadcwww.hia.nrc.ca/software/distributed/.

4. Data Processing Recipes

This section will also outline the standards and conventions which recipes must follow to be integrated into the GSA. These will include handling of WCS information, fault detection, error propagation, and run-time documentation (log file) generation.

4.1 Data Processing Recipe Scripts

DPRSs are executable programs which take standard command line arguments, and return results in a standard way. It is expected that most DPRS's will be simple wrapper scripts, which parse the command-line arguments, execute another task (usually an IRAF task), parse the results from the task, and return the results in the standard form. DPRSs are executed with a command line in the form:

scriptName [-fromStore] [*parameters*] [-input [-role *roleName*] *URL* ...] [-input [-role *roleName*] *URL* ...] ...

The "-fromStore" option is described in Section 4.2 on page 89.

The "-input" option specifies an input to the recipe. The input could identify a raw dataset to be read from the data archive, or the result of another data processing recipe.

The "-role" option specifies the role the input will play in the data processing, as taken from the recipe input table. This is only necessary when the data processing software uses different inputs in different ways, and cannot determine how each input should be used from the data. (E.g. it should be possible for a flat/bias correction DPRS to determine what roles its inputs play, but a DPRS which subtracts one image from another would have to be told how the inputs are to be used.)

Note: Could get rid of -role and have a simpler command line: [-input *roleName URL URL* ...]. *Roles are not always necessary, but you can probably always think of something to put there*.

The URLs indicate how the inputs to the recipe should be obtained. When the input is from another recipe, the URL is in the form "*hostName:path*" (i.e. a file or directory name usable by the UNIX rcp program). When the input is a dataset from the archive, the URL is in the form "*archive fileid*" (i.e. a fileid usable by the acp program described in Chapter 8, section 5.1 on page 133). Standard data retrieval recipes will retrieve data from the archive and provide the data files as output to be used by other recipes, and so other recipes will not need to directly access the archive.

The parameters will be those stored in the recipe instance table, or supplied by the application in those cases where the recipe instance was not stored in the data processing catalogue. The parameters will be script dependant, and must not conflict with the standard options.

All output from DPRSs should be sent to standard output. There are three categories of output:

Log entries — These are strings in the form "log: *message*". These will be permanently recorded in the data processing system log file.

Error messages — These are strings in the form "error: *message*". These will be permanently recorded in the data processing system log file. If a DPRS produces any error messages, all output from the DPRS will be considered unreliable.

Result URLs — These are results which can be used as input by another DPRS. Result URLs are in the form “output: *[role] URL*”. Where role identifies the type of output, and path is a file or directory where the data is located.

The role is used when recipe produces more than one distinct result (for example?). In these cases, other recipes identify the output they want by matching the “outputRole” value from the recipe input table, with the role reported in the output from the recipe.

In addition to the explicit outputs, a DPRS can have “side” effects outside of the data processing system. These side effects may include:

- Sending data files to the bulk data subsystem for permanent storage in the archive.
- Sending data files to be written to user media.
- Sending data files to an user request FTP area.
- Inserting or updating information in the archive catalogues.

If the recipe is able to retrieve all of its results from a permanent store (i.e. either the archive catalogues, or the archive bulk data store), then the recipe may be categorized as a “persistent recipe” as described in Section 4.2 on page 89.

Some data processing recipes may also create new DPRIs and insert them into the data processing catalogues. These will be the Data Processing Discovery Agent recipes described in Section 4.3 on page 89.

4.2 Persistent Recipes

Persistent recipes save their output to a persistent store where the output can be retrieved at a later time (generally on archive media or in a database). Persistent recipes are identified by the **persistent** field in the recipe table. All persistent recipe scripts take a parameter “-fromStore” which when used indicates that the recipe should retrieve previously generated output from the persistent store. Persistent recipes are responsible for storing their output to the persistent store, and retrieving the results when the “-fromStore” option is used. If the “-fromStore” option is used, and the results are not available, the recipe script will return an error.

4.3 Data Processing Discovery Agent

The Data Processing Discovery Agent is a set of recipes which identify reasonable data processing sequences, and inserts DPRIs into the data processing catalogues. The Data Processing Discovery Agent also sets an appropriate priority and public release date for the DPRIs it creates.

The Discovery Agent recipes may have to become involved when release dates (and other metadata values) are updated by Gemini, since these changes will have to propagate through the data processing.

Data processing tasks identified by Discovery Agent must be easily extendible as new data processing options are identified.

The Data Processing Discovery Agent will be a set of data processing recipes, which examine each data superset and determines which data processing is possible and appropriate. Some of the Discovery Agent recipes will be archive and/or instrument specific, and others will be generic. Discovery Agent recipe instances should be added to the data processing queue (see Section 3.3

on page 86) whenever new data supersets or associations are added to the GSA catalogues. Some typical examples of the analysis done by the Discovery Agent recipes are:

- Examine each science exposure to find appropriate basic calibration data. If calibration data is found, add the basic calibration data processing recipe instance to the data processing catalogues.
- Examine each calibrated data superset and determine if it is a member of an existing association of stacked images, or if a new association can be created with other data supersets. This may be a two step process, where one recipe identifies “potential” associations based on meta-data in the GSA catalogues, followed by a more detailed analysis recipe which cross-correlates the images to determine if they really can be stacked. The analysis data processing recipes would insert image stacking data processing recipe instances into the data processing catalogues.
- Examine each calibrated data superset and determine if sextractor and galfit should be executed, and if necessary, add the data processing recipe instances to the data processing catalogues.
- Examine each calibrated data superset to determine if a preview image should be generated. If the preview should be generated, appropriate data processing recipe instances will be inserted into the data processing catalogue.

Data Processing Discovery Agent recipes may be nested in the sense that one Discovery Agent recipe may insert other Discovery Agent recipe instances into the data processing catalogues. There are two possible reasons for nested Discovery Agents:

1. To create a hierarchy of simple Discovery Agent recipe instances instead of a single complex recipe instance. For example, the top level Discovery Agent recipe instance may only examine the archive associated with a dataset, and insert one or more archive specific Discovery Agent recipe instances. Likewise, an archive specific Discovery Agent recipe instance may only determine which instrument is associated with a dataset, and insert instrument specific Discovery Agent recipe instances.
2. The first step in identifying stacked associations will be to do a relatively simple examination of the catalogues to determine potential members of an association. The potential members may then be cross-correlated to determine if they can actually be stacked. Since the cross-correlation step could involve a large amount of work, and will involve looking at potentially proprietary pixel data, this could be implemented as a second Discovery Agent recipe instance inserted into the data processing catalogues when the first Discovery Agent recipe instance detects a potential association.

There are several factors which must be taken into account by the Data Processing Discovery Agent, and which will complicate the design:

- Datasets needed as input to recipes may be collected over a long period of time, e.g. calibration data may be collected days or weeks after science data is collected. This may mean that existing calibration recipes instances may need to be updated.
- Determining if processing is possible may itself require considerable processing, i.e. determining for sure if a dataset is a member of a stacked association may require cross-correlating the members of the association.
- Discovery Agent recipes which need to look at proprietary data cannot be run until data becomes public.

- There are interdependencies between recipes. For example if a preview can be produced for an association there may be no reason to produce previews for the members of the association.
- Some sequence optimization may be difficult to do. For example, if cross-correlation is required for evaluating associations, much of the work needed to generate a preview must also be done before doing the cross-correlation. It would be good to avoid re-doing this work, but that would require modifying a data processing tree after its execution is started.

5. Hardware Requirements

This section describes the expected hardware requirement needed to support GSA data processing.

5.1 Database space

The maximum and average number of datasets per year predicted from Gemini, as given in [7] are 134000 and 84000 respectively. Given these values, Table 38 on page 91 shows the expected disk space growth and number of rows expected in each of the data processing catalogue tables (the recipe table is not included since its size will be very small, and mostly static). The worst case estimates assume that datasets will have an average of 5 recipes, and each recipe will have an average of 5 inputs. The most likely estimates assume that every dataset will have an average of 1 recipe, and each recipe will have 2 inputs. Although there is a wide range in the predicted space needs, even the worst case can be handled without significant impact on the operational cost.

TABLE 38. Predicted data processing database usage

Table	yearly database usage	Maximum Year		Average year	
		Worst case	Likely case	Worst case	Likely case
dataset table	rows/year	134000	100000	84000	60000
	MB/year	14	10	8	6
recipe instance table	rows/year	670000	100000	420000	60000
	MB/year	80	12	50	8
recipe input table	rows/year	3350000	200000	2100000	120000
	MB/year	200	12	125	8
Total MB/year		294	34	183	22

5.2 Processing nodes

The GSA will need computers to do the data processing for the archive and for data requests from Gemini archive users. The most cost effective way to provide these CPU cycles is with multiple, relatively inexpensive PCs, running LINUX. Each processing node would require:

- Enough disk space to process the largest reasonable processing task.
- A reasonable amount of memory.
- Access to IRAF, the required packages, and any other necessary software.

The number of computers needed depends on the amount of data processing done for standard archive centre processing, and for archive user requests. In Chapter 7, section 4.3 on page 123, specifies that one CPU would be required for processing user requests. The data rates specified in [7], indicate that there will be about 325 datasets per night. We can assume that the minimum basic processing (preview) will require ~3 minutes per dataset, for a total of about 16 hours per day or one mostly dedicated processing node. It is difficult to estimate the needs for advanced processing (e.g. object detection), since it will be done on a subset of the datasets. A reasonable guess would be one to two processing nodes to do the advanced processing.

In summary, a total of 3 processing nodes should be able to handle all of the archive centre processing for Gemini (excluding processing done for user requests).

The data processing nodes will not be dedicated systems. They will be shared between the archives supported by the archive centre, and will process whatever tasks have highest priority, and so all available processing nodes may be processing a user data request, or they may all be doing archive centre processing, depending on the current load on the system.

Chapter 6 *Data Ingest*

1. Introduction

This chapter describes the design of the GSA Data Ingest System. The Data Ingest system is responsible for the GSA side of the interface between Gemini and the GSA. This system is based on the prototype described in [11], although much of the functionality of the prototype will be integrated into the Gemini DHS Data Server running at the Gemini telescopes.

The functions of the data Ingest system are:

- Merge multiple the Gemini meta-data stores described in [3] into the GSA catalogues described in Chapter 4.
- Initiate the data processing discovery agents described in Chapter 5.
- Verify that all datasets in the Gemini meta-data store have corresponding raw data files in the bulk data store.
- Allow recovery of the meta-data stores from raw data files.
- Verify that copies of data are available on all required media types.

2. Requirements on the Data Ingest Subsystem

Table 39 on page 93 list the software requirements which apply to the GSA Data Ingest subsystem. Each requirement is followed by a short description of how the requirement will affect the Data Ingest subsystem. See [6] for a complete description of these requirements.

TABLE 39. User interface requirements

SR1.1 The GSA shall archive all data. All data supersets received from gemini will be ingested into the catalogues, and will be available through the bulk data retrieval system.
SR1.2 The GSA shall respect proprietary periods assigned by Gemini. The data ingest system monitors the meta-data database and propagates the release date value into the GSA catalogues.
SR1.4.2 Remove previews when release date is changed. When release date is updated for a dataset, the catalogue update system should cause the DP Discovery Agent to re-evaluate the proprietary status of all data products derived from the now proprietary dataset.
SR2.5.1 Remove derived image descriptors for proprietary data. See SR1.4.2.
SR2.6 Release date is a data descriptor. Release date will be ingested into the GSA catalogues, and updates to release date will be reflected in the GSA catalogue content.

TABLE 39. User interface requirements

SR2.8 Data quality assessment shall be in the catalogue. Data quality will be ingested into the GSA catalogues. This is not described explicitly in this chapter, but the data quality will be stored in the FITS headers, and incorporated in the meta-data store, and therefore will be included in the GSA.
SR2.9 The science program shall be associated with the catalogue data. Science programs will be ingested into the GSA catalogues, and a link will be established between science program and datasets collected under the science program, or associated data supersets which incorporate data collected under the science program.
SR2.10 The electronic observing log shall be associated with the catalogue. The electronic observing log will be ingested into the database. An association will be established between the electronic observing log and the data supersets in the GSA catalogue (based on exposure start/stop time, and the time-stamps associated with entries in the electronic observing log).
SR2.11 Proposal id shall be a descriptor. Proposal id will be ingested into the GSA catalogues.
SR2.13 Calibration data shall be associated with science data. When data supersets are ingested into the GSA catalogue, links will be established to the appropriate calibration data.
SR2.14.4 Remove derived objects for proprietary data. See SR1.4.2.
SR2.15 Publications should be linked to data supersets. When publication information is received, links will be established to data supersets used in the publication. This requirement has been “de-scoped” to link science programs with publications.
SR2.16 The GSA shall store Gemini hardware and software versions. Version information will be ingested into the GSA catalogues. This is not described explicitly in this chapter, but the versions will be stored in the FITS headers, and incorporated in the meta-data store, and therefore will be included in the GSA.
SR4.1 Maximum and average requirements for catalogue ingest. The ingest software must support the data rate requirements.
SR4.2 Maximum and average requirements for raw data ingest. The ingest software must support the data rate requirements.

3. Database Schema

The data ingest subsystem uses the tables in the Gemini meta-data store as the source of catalogue data, the GSA catalogue tables as the destination for catalogue data, and the GSA data processing tables to set up data processing. The meta-data store tables are described in [3], the catalogue tables are described in Chapter 4, and the data processing tables are described in Chapter 5, and the descriptions will not be repeated here.

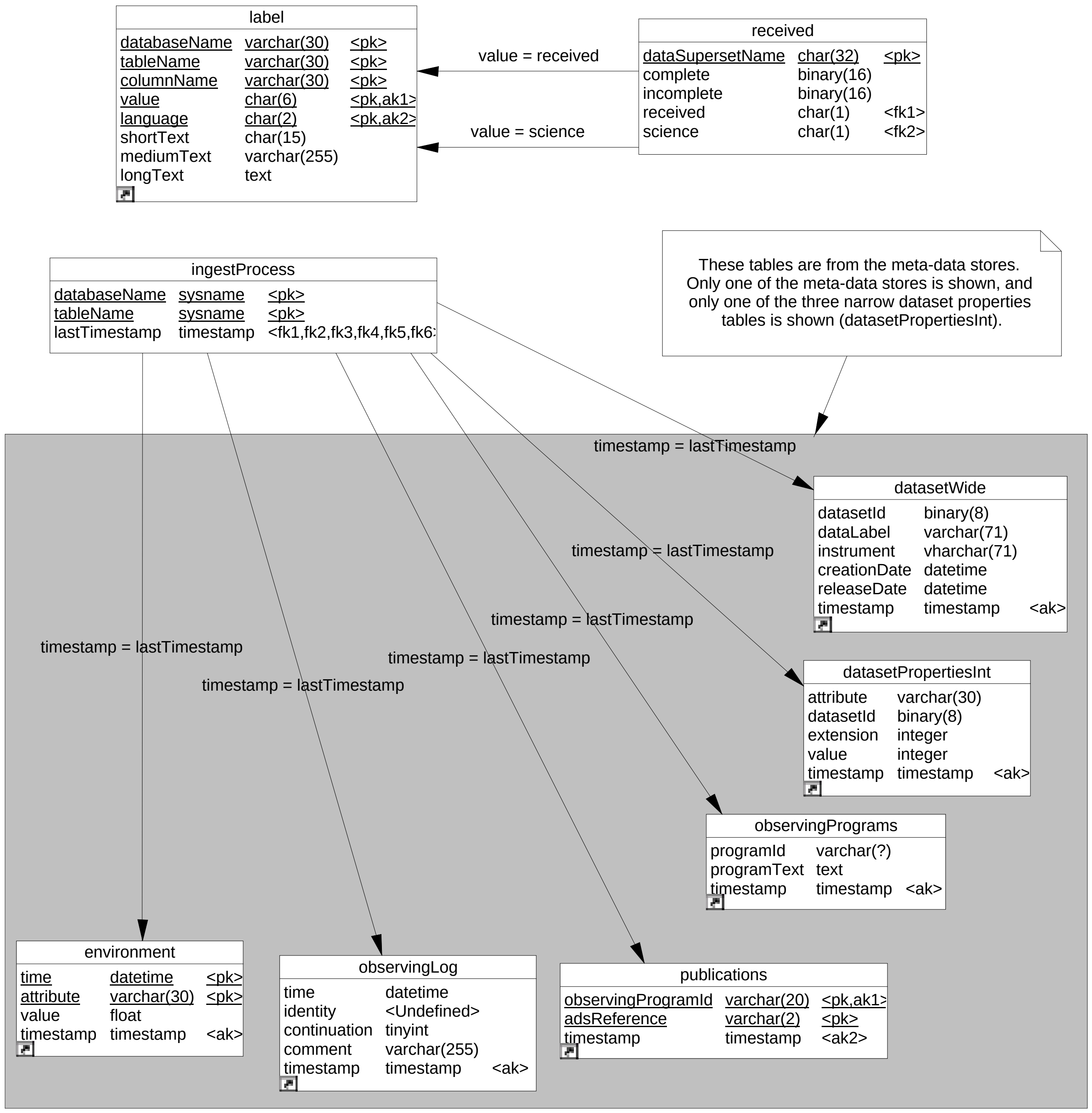
3.1 Data Ingest table storage

The data ingest catalogue will be a Sybase relational database. An overview of the data ingest tables is shown in Figure 16 on page 95.

3.1.1 Table: ingestProcess

This table is used to control the processing of the gemIngest program, allowing the program to efficently process any new data added to the meta-data store, without doing complicated queries

FIGURE 16. Ingest tables



to determine which data is already in the GSA catalogues. As gemIngest processes each row in the meta-data tables, it records the timestamp of the last row processed, and when the processing is finished, saves that timestamp in this table. This allows the gemIngest program to resume processing with the next largest timestamp. There is one row in this table for each meta-data table processed by gemIngest.

The columns of table ingestProcess are shown in Table 40 on page 96.

TABLE 40. Columns of table ingestProcess

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
data-baseName	The name of the meta-data database containing the table.	sysname	TRUE	TRUE	FALSE
tableName	The meta-data table processed by the gsaIngest program.	sysname	TRUE	TRUE	FALSE
lastTimes-tamp		timestamp	FALSE	FALSE	TRUE

3.1.2 Table: received

The received table is used to track the availability of data in the archive. The current state of each data superset is recorded in this table. Both simple datasets, and associated data supersets will be recorded in this table.

The columns of table received are shown in Table 41 on page 96.

TABLE 41. Columns of table received

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
dataSuperset-Name	The name of the data superset.	char(32)	TRUE	TRUE	FALSE
complete	This flag indicates the types of archive media which have complete copies of the raw data for the data superset. See the text for a description of the values for this field.	binary(16)	TRUE	FALSE	FALSE
incomplete	This bit field indicates the types of media which contain at least some of the raw data for the data superset, but which do not contain a complete set of the data. See the text for a description of this field.	binary(16)	TRUE	FALSE	FALSE
received	This flag indicates if the data for the data superset is available from the archive. See the text for possible values of this flag.	char(1)	FALSE	FALSE	TRUE
science	This flag indicates if the data superset has been designated a science data superset. See the text for possible values for this flag.	char(1)	FALSE	FALSE	TRUE

The complete and incomplete fields indicate the status of each of the data supersets on each type of archive media. These fields are used to control the migration from one media type to another, allowing software to easily determine which datasets need to be migrated to a media type. The complete field indicates which media types contain a complete copy of the raw data. The incomplete field indicates which media types contain at least some of the data for the data superset, but which do not contain a complete copy of the data (this is only possible in the situation where the raw data is stored in more than one file, as is the case for associated data supersets). Each bit in the bit fields represents a media type.

The current definitions of the bits of interest to the GSA are:

0x0001 — DVD.

0x0002 — CD-ROM.

0x0004 — Magnetic disk.

Note that all zeros in the complete field does not necessarily indicate that the archive center does not have a complete copy of the data. All of the files may be available, but spread over more than one type of media. The received flag indicates if all of the data is available.

The values of the received flag can be "Y" (yes) if all data for the data superset is available, "N" (no) if none of the data for the data superset is available, or "I" (incomplete) if some files for a data superset are available and some are not.

Possible values for the science flag are "Y" (yes) for science datasets, " " (not checked) for datasets which have not been evaluated. The values for datasets which have been designated as not science will be determined when the gemScience program is designed.

3.2 Received table access

The received table will be accessed by several other modules. To simplify access to the table, the **gemRec** library will be created. This library will provide functions to:

- Select rows from the table based on various search criteria:
  - the value of the science field
  - the value of the received field
  - a range of dataset names
- Insert new rows into the table.
- Remove rows from the table.
- Update rows in the table.

4. Data ingest software

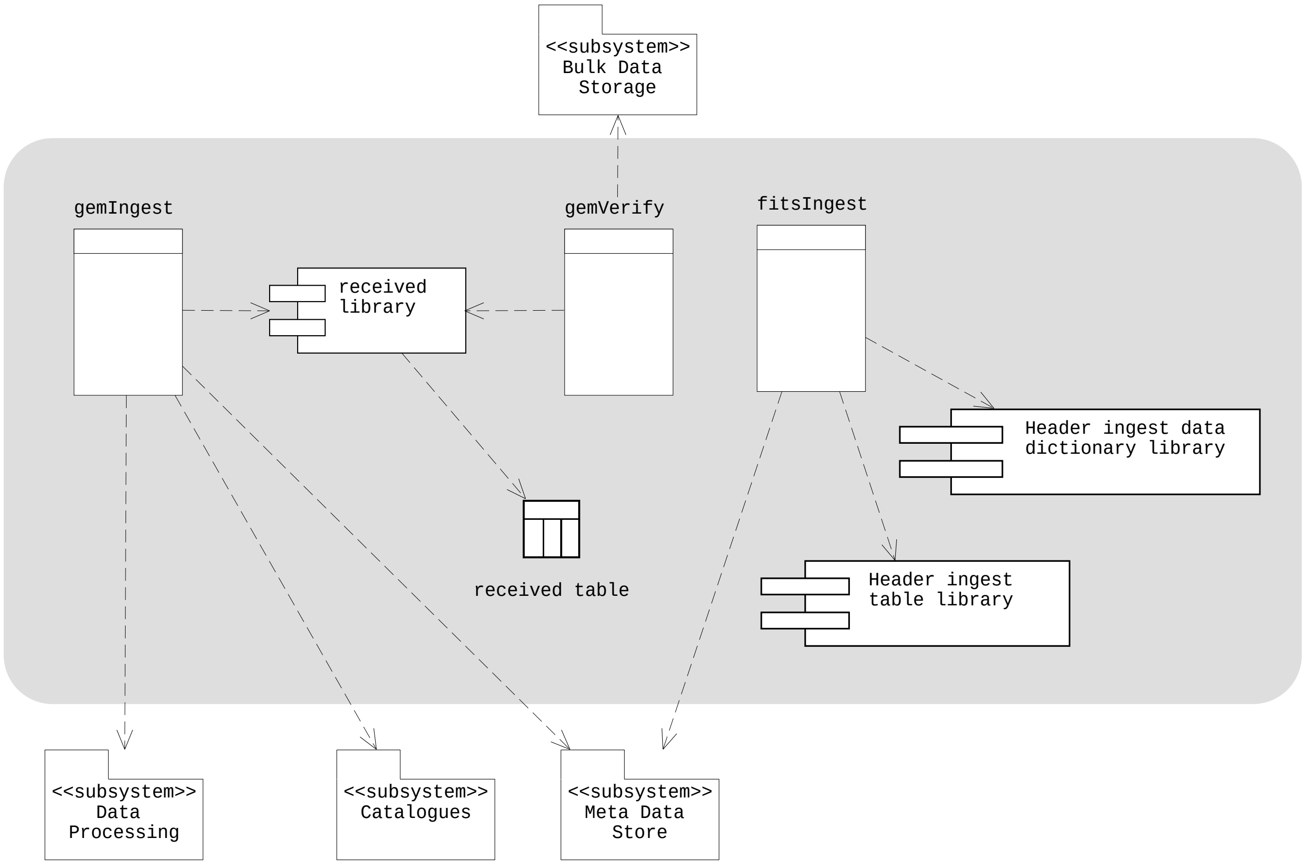
The data ingest software is responsible for monitoring both insertions and updates to the GSA meta-data database described in [3], and the bulk data received from Gemini. New and updated data will be incorporated into the GSA. The meta-data stores will be populated by a modified Gemini DHS Data Servers running at the two Gemini telescopes. The modifications required for Data Server to perform meta-data store population task are described in [14]. A component diagram for the data ingest subsystem is shown in Figure 17 on page 98. The shaded area of the diagram indicates the components which are a part of the Data Ingest subsystem.

The evaluation of the science state stored in the received table is done by the gemScience program described in Chapter 4, section 4.2 on page 71.

4.1 Monitor meta-data store

A program named **gemIngest** will be created to examine the GSA meta-data tables described in [3]. gemIngest will be run daily to copy any new data from the Gemini meta-data stores into the GSA catalogues. This program will perform the following tasks:

FIGURE 17. Data Ingest subsystem overview


<<subsystem>> Bulk Data Storage
gemIngest
gemVerify
fitsIngest
received library
Header ingest data dictionary library
received table
Header ingest table library
<<subsystem>> Data Processing
<<subsystem>> Catalogues
<<subsystem>> Meta Data Store

- Populate the received table with any new datasets found in the meta-data stores from both Gemini North and Gemini South.
- Populate the dataSuperset catalogues described in Chapter 4 from the Gemini North and Gemini South meta-data stores.
- Copy the observing programs from the Gemini North and Gemini South meta-data stores to the GSA catalogues observing program table. This will also involve extracting various parameters from the observing programs, and insert them into the GSA observing program tables as searchable fields.
- Merge the Gemini North and Gemini South observing logs into a single observing log.
- Merge the Gemini North and Gemini South environmental data tables into a single table.
- If necessary, merge the Gemini North and Gemini South publications tables into a single table. However since it seems more efficient to only have one group monitoring publications for both telescopes, it seems likely that there will only be one publications table.
- Queue the first level data processing for each dataset for execution. The first level data processing will include:
  - Examine external object catalogues (SIMBAD and NED) to determine which objects should be in the field of view, and inserting these objects into the object catalogue tables.

  - A Data Processing Discovery Agent task which will identify any further data processing appropriate for the dataset. The Data Processing Discovery Agents are described in Chapter 5, section 4.3 on page 89.

  The initial level of data processing will scheduled for high batch priority execution. The initial data processing will not require any access to proprietary data, and so is not dependant on the release date of the raw data.

- Detect any modifications to existing data in the Gemini meta-data stores, and begin the propagation of the changes through the GSA catalogues. This will involve re-copying the modified meta-data to the catalogues.
- A change to a release date which causes a public dataset to become proprietary will cause a high priority data processing task to be started. This task will examine the GSA system and determine which archive data products should be removed from the system (previews, image descriptors derived from the data, sources, object parameters derived from the data, etc.). Initially, this task will send a message to the archive operators. The message will include a script which an operator can execute to perform any required deletions from the archive tables. If making public data proprietary becomes a common occurrence, and when confidence is gained that the deletions are correct, then this process may be automated. *(Changes to release date should normally occur before becomes public, and so making public data proprietary should be considered error recovery and not part of the normal way of operating.)*

For efficiency, this program will use the timestamps in the Gemini meta-data stores to determine which data needs to be examined. By default, the program will process only new rows in each of the meta-data store tables in timestamp order. Each time the program completes processing one of the tables, the largest timestamp processed will be recorded in the ingestProcess table. The next time the program executes, it can resume processing at the next newest row in the tables.

The program will have command-line options to enable the following functions:

- Process all datasets, not just the new datasets.
- Process all datasets in a given range of datasets.
- Only process datasets from a specified meta-data database (either Gemini North or Gemini South).
- Process data from the dataset meta-data store tables, but not the other tables in the meta-data stores unless explicitly requested by other command line options.
- Process environmental data from the meta-data stores, but not the other tables in the meta-data stores unless explicitly requested by other command line options.
- Process publication information from the meta-data stores, but not the other tables in the meta-data stores unless explicitly requested by other command line options.
- Process the electronic observing logs from the meta-data stores, but not the other tables in the meta-data stores unless explicitly requested by other command line options.
- Process the Observing program tables from the meta-data stores, but not the other tables in the meta-data stores unless explicitly requested by other command line options.

The program will have a configuration file which contains the following data:

- The database name and tables names of the meta-data stores described in [3]. This should include information describing separate Gemini North and Gemini South meta-data stores, and should also allow for meta-data stores generated by the GSA using the **fitsIngest** program described in Section 4.3 on page 100.

- A list of data processing tasks to start for each new dataset.
- The names of the GSA catalogue tables to received the meta-data.

4.2 Verification of data

A program named **gemVerify** will be created to verify the contents of raw data archive against the catalogues. The program will verify the list of data files received from Gemini against the content of the received table, and update the complete, incomplete, and received fields appropriately. The program will use the ad library and tables described in Chapter 8, section 5.2 on page 134 to determine which files are available for a data superset. By default the program will process any files on archive media where the verify_date field (as recorded in the mfs table) is null. For removable media, this program will be run when new media are created or received from Gemini. For magnetic disk media, this program will be run on an as needed basis after the magnetic media have been modified.

The program will have command line options to enable the following functions:

- Process all datasets in the received table.
- Process all datasets in a given range.
- Process all media volumes in the archive.
- Process all media volumes in a given range.

4.3 Keyword recovery via fitsIngest

The **fitsIngest** program will allow meta-data to be read from FITS files, instead of from the Gemini meta-data store. The ability to read meta-data from FITS files will allow datasets collected before the creation of the meta-data store to be incorporated into the GSA, and will allow disaster recovery if the meta-data store is lost. Program fitsIngest reads files from the archive, and populates meta-data tables in the same way as is done by the DHS Data Server described in Section 4. on page 97. This program already exists as part of the GSA prototype, described in [11], and so very little effort will be needed to continue its existence.

4.4 Header ingestion data dictionary library

The **hid** library was developed as a part of the GSA prototype described in [11]. The **hid** library is responsible for maintaining a data dictionary which will be used when parsing multi extension FITS files. The library will be used by the **fitsIngest** program at the archive centre, and will be incorporated into the Gemini DHS Data Server to be used when parsing the FITS headers at Gemini. The **hid** library provides functionality to verify the correctness and completeness of FITS headers, as described in the data dictionary.

4.5 Header Ingestion table library

The **hit** library was developed as part of the GSA prototype described in [11]. The **hit** library is responsible for maintaining the meta-data tables of raw header data. The **fitsIngest** program will use the **hit** library when reading data from the meta-data tables, and the library will be incorporated into the Gemini DHS Data Server to be used to parse the FITS headers at Gemini.

5. Hardware Requirements

The data ingest system should not require any significant hardware resources. There will be some database load incurred in maintaining the catalogues, however this will not be heavy enough to significantly impact the performance of the database server.

The disk space needed to store the data ingest tables will also be minimal, at between 5 and 8 Megabytes per year.

Gemini Science Archive

Chapter 7 Data Retrieval Subsystem

1. Introduction

This chapter describes the conceptual design of the GSA Data Retrieval Subsystem.

The GSA Data Retrieval Subsystem performs the following tasks:

- Accepts user Data requests from the GSA user interfaces.
- Persistently stores user data requests in a queue. This is used as input to the data processing system, and serves as a permanent record of archive activity.
- Submits data processing requests to the GSA Data Processing system to retrieve and/or process user data requests.
- Organizes FTP requests into an appropriate FTP disk area.
- Submits media creation requests to the Bulk Data Storage subsystem to create removable media when requested by a user.
- Notifies users that requests are complete.

Figure 18 on page 104 shows a graphical overview of the components of the Data Retrieval subsystem as they relate to each other and the other subsystems of the GSA. See Chapter 1, section 9.1.2 on page 13 for a description of component diagrams. The shaded area denotes the components which are part of the GSA Data Retrieval subsystem. Note that data processing subsystem is used for retrieving data from archive media, and the Bulk Data Storage subsystem is used for creating user media.

The components of the GSA Data Retrieval subsystem are described in detail in the following sections:

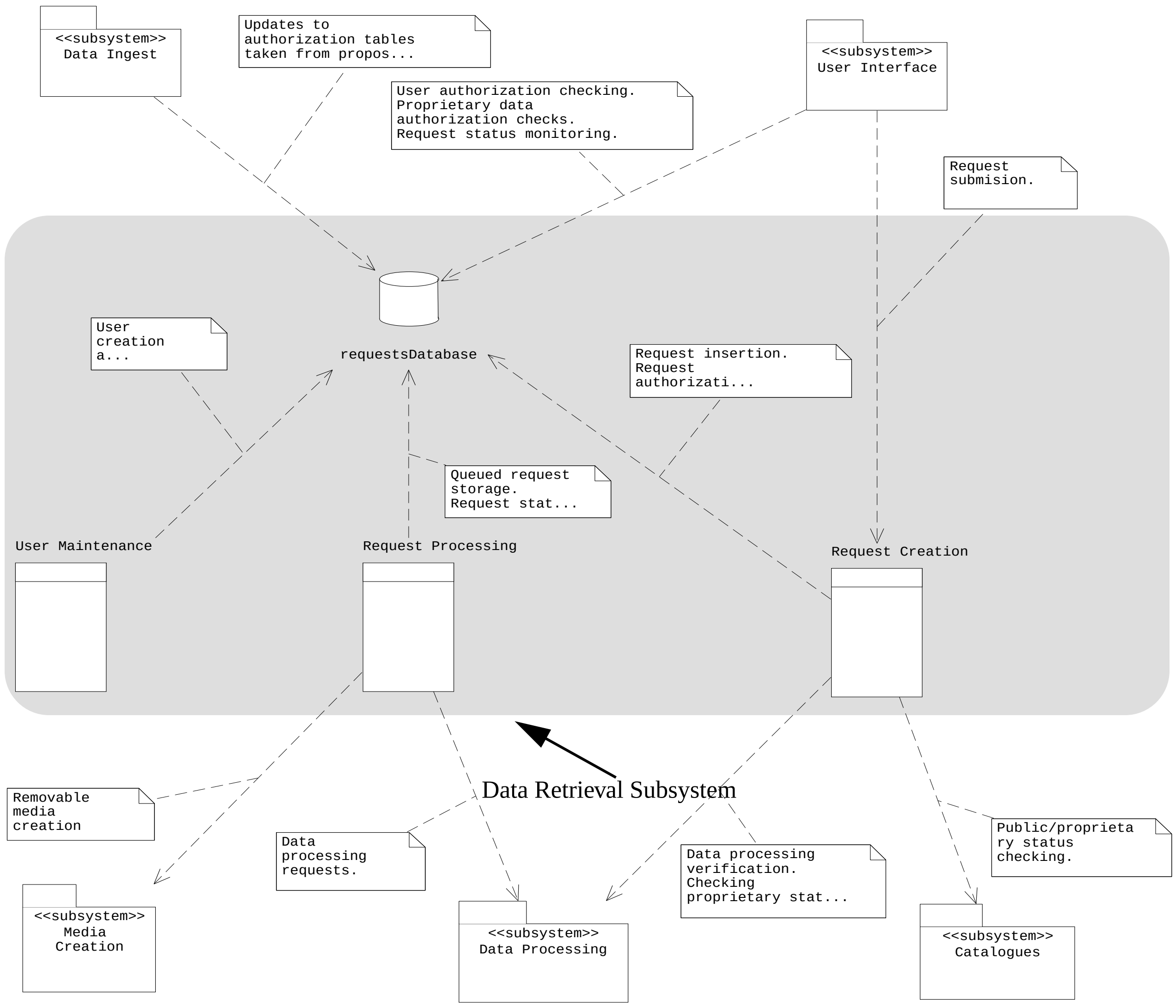
- the requests database is described in Section 3. on page 106
- the user maintenance component is described in Section 3.2.1 on page 116
- the requests creation component is described in Section 3.2.2 on page 118
- the request processing component is described in Section 3.2.3 on page 120

The other subsystems of the GSA are described in other chapters in this document.

2. Requirements on the Data Retrieval Subsystem

Table 42 on page 104 lists the GSA software requirements which apply to the GSA Data retrieval System. Note that these requirements are requirements which “have some impact” on the Data Retrieval subsystem, but the Data Retrieval subsystem is not necessarily the only subsystem associated with the requirements in the table. A note is included with each software requirement,

FIGURE 18. Data Retrieval System overview


<<subsystem>> Data Ingest
Updates to authorization tables taken from propos...
User authorization checking. Proprietary data authorization checks. Request status monitoring.
<<subsystem>> User Interface
Request submision.
User creation a...
requestsDatabase
Request insertion. Request authorizati...
Queued request storage. Request stat...
User Maintenance
Request Processing
Request Creation
Data Retrieval Subsystem
Removable media creation
Data processing requests.
Data processing verification. Checking proprietary stat...
Public/proprietary status checking.
<<subsystem>> Media Creation
<<subsystem>> Data Processing
<<subsystem>> Catalogues

which describes how the requirement affects the Data Retrieval Subsystem. See [6] for a complete description of these requirements.

TABLE 42. User interface requirements

SR1.2 The GSA shall respect proprietary periods assigned by Gemini. Retrieval system will prevent users from retrieving proprietary data without proper authorization.
SR1.3.1 Data processing shall respect Gemini policy on proprietary periods. See SR1.2.

TABLE 42. User interface requirements

SR1.3.2 Data processing shall proceed promptly. The retrieval system shall submit requests for user data processing at an appropriate priority level. The retrieval system may have to modify priority in order to prevent large requests from blocking small requests.
SR1.5 Collect information needed to track of publications based on archive data. Data requested by users will be recorded, with enough information to identify all data supersets included in the data request. Users will be asked to identify the GSA as a source of data in any publications based on GSA data, and will be asked to include information in the publication which allows the specific GSA requests to be identified.
SR3.4 GSA users shall be able to select data for retrieval. Retrieval system will retrieve data.
SR3.4.1 Users shall retrieve data via Internet, EXABYTE, or CD-ROM. The retrieval system supports these delivery media.
SR3.4.2 Only authorized users shall be able to retrieve proprietary data. See SR1.2.
SR3.4.3 GSA users should be able to choose to have data processed. The retrieval system will accept data processing information with data requests, and will be able to submit appropriate data processing requests to the GSA data processing system.
SR3.4.4 Data processing shall not use proprietary data as input. See SR1.2.
SR3.4.5 Processing recipes shall be documented. The retrieval system will collect data processing documentation and include it with the user request (this could be done by the data processing system).
SR3.4.6 Authorized Gemini staff can retrieve proprietary data. The retrieval system will accept authorization information with a request, will verify that the information is correct, and will pass the information on to the data processing system if necessary. Authorized Gemini staff have access to all GSA data.
SR3.4.7 GSA users can retrieve calibration data. The retrieval system should have a simple “retrieve all calibration files” option for requests. The retrieval system will have to access the DP calibration database to determine which data supersets need to be added to the request.
SR3.4.8 Retrieve environmental data. The retrieval system should have a simple “retrieve all environmental data” option for requests. The retrieval system will have to access the catalogues to determine which environmental data files have to be added to the request, or create the environmental data files.
SR3.4.9 It shall be possible to retrieve science programs. The retrieval system should have a simple “retrieve science programs” option for requests. The retrieval system will have to access the catalogues to determine which science program files must be added to the request.
SR3.4.10 It shall be possible to retrieve proposal information. The retrieval system should have a simple “retrieve proposals” option for requests. The retrieval system will have to access the catalogues to determine which proposal files must be added to the request.
SR3.4.11 It shall be possible to retrieve the electronic observing log. The retrieval system should have a simple “retrieve proposals” option for requests. The retrieval system will have to access the catalogues to determine which proposal files must be added to the request.

TABLE 42. User interface requirements

SR3.4.12 It should be possible for users to monitor the progress of data requests. The retrieval system should keep status and progress information describing each active request. The information should be available to users through the user interface. State should include information supplied by the data processing system.
SR3.4.13 Batch mode requests. The Data Retrieval subsystem should accept requests through either a GUI or a batch mode interface.
SR3.4.14 Document user data requests. The retrieval system should accept documentation from the user interface, and include the documentation with the data returned for a request.
SR3.4.15 Retrieve associated data supersets. The retrieval system should accept requests for data supersets, and will use the GSA catalogues to determine which datasets should be returned for the data superset. If requested, the retrieval system will initiate any data processing requested to create data products for the associated data superset.
SR3.4.16 Provide data processing recipes for associations. The retrieval system will be able to trigger data processing for associated data superset.
SR4.3 Maximum and average requirements for data retrieval by users. The retrieval system will be able so support the retrieval data rates.
SR4.6 Data for Internet retrieval should be available promptly. Internet data requests will have priority over physical media data request. Data processing for Internet requests will have priority over other data processing tasks. Large data requests (including Internet requests) will not block small data requests.

3. Data Retrieval Infrastructure

This section describes the infrastructure needed to support the data retrieval subsystem.

3.1 Data Retrieval Database

The requests database is similar to the existing CADC request database. This database acts as a queue of un-processed user data requests, is a permanent record of past user requests, and stores information about the users of the CADC archives. All of the tables in the Request Processing subsystem are intended to be shared tables, used by all archives hosted by the archive centre.

The tables can be divided into the following functional areas:

- Tables which store user information ("users" table (Section 3.1.6 on page 115) and "authorization" table (Section 3.1.1 on page 107)).
- Tables which store information describing requests submitted by users (the "requests" table (Section 3.1.5 on page 114), the "requestProcessing" table (Section 3.1.4 on page 113), and the "requestDatasets" table (Section 3.1.2 on page 110)).
- Tables shared with other GSA Subsystems (label Table).
- Static information (requestOptions Table (Section 3.1.3 on page 111)).

In addition, the data retrieval system uses tables that are not considered part of the data retrieval system. These include the recipe instance and dataset tables from the Data Processing subsystem.

The relationships between the retrieval tables are shown in Figure 19 on page 108.

3.1.1 Table: authorization

This table contains information which allows the archive to determine if a user is permitted to access proprietary data.

The authorization table is maintained by the user maintenance program, and the authorization table may also be populated or modified by the Data Ingest subsystem (as new proposals are received, authorizations entries would be added to provide access to data collected for a proposal to users able to supply the correct "proposal" password). In order to protect proprietary data, the content of this table must be protected from unauthorized access.

There will be one row in this table for each Gemini or GSA staff member authorized to access proprietary data, and one row for each executed proposal.

For the GSA, this table would be used in two different ways:

1. Authorized Gemini staff will have entries in the authorization table with their GSA archive user names. The password, tableName, columnName, and value columns would all be NULL. This will give these staff members access to all proprietary data in the Gemini archive. (User authentification will be done by the Gemini User interface.)
2. Each Gemini proposal will have an entry in the authorization table. The userId field will be NULL, the password field will contain the proposal password, the databaseName, tableName and columnName fields will identify the column where proposal identifier is stored in the Gemini catalogues, and the value field will be the proposal id.

This table will allow the request creation task to look up a proposal Id and proposal password provided by the user, to determine if a user is authorized to access any given dataset.

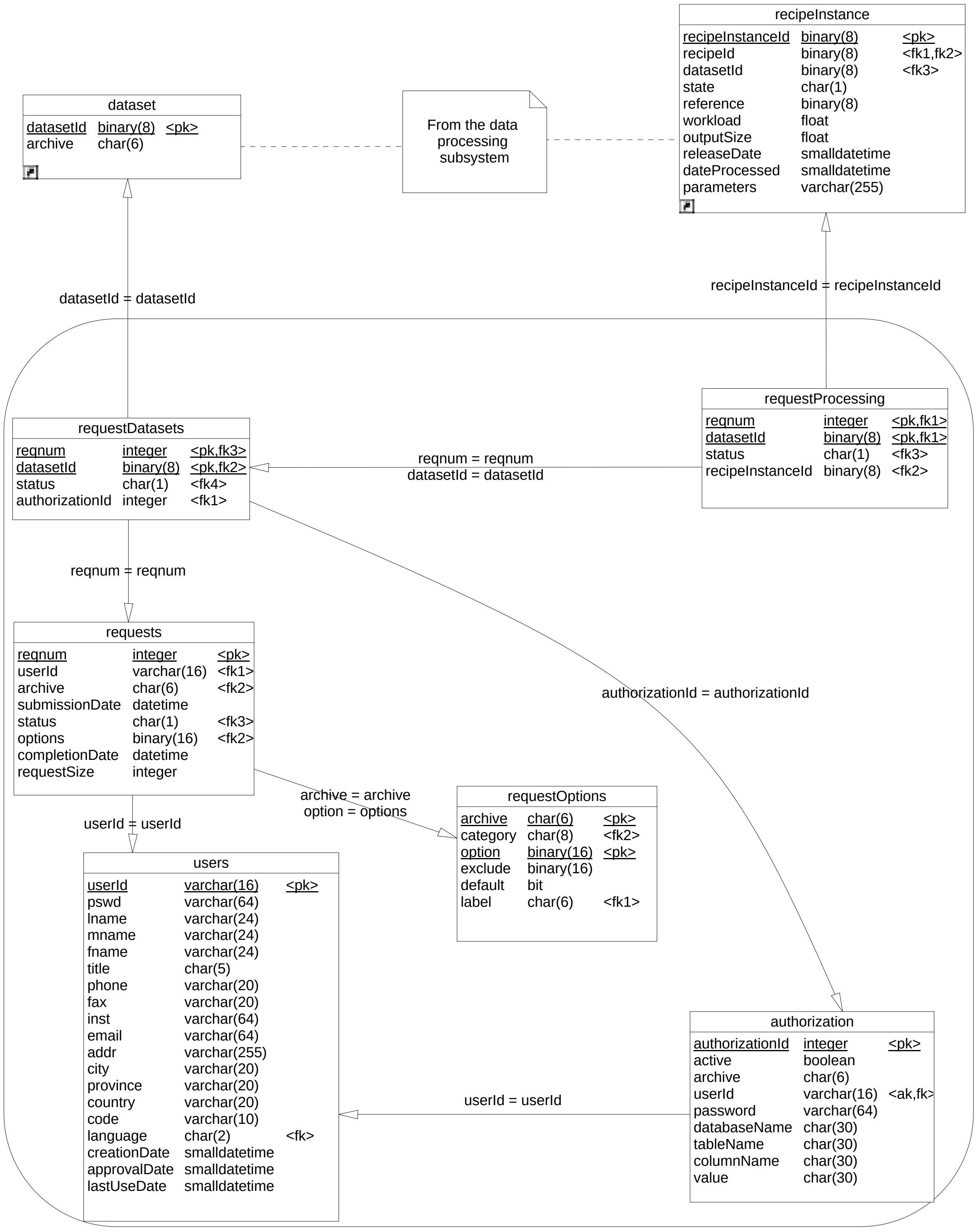
The algorithm for this use of this table would be something like this:

- A user attempts to retrieve proprietary data.
- The user interface notifies the user that there is prorietary data in the request, and gives the user the option of either eliminating the proprietary data from the request, or attempting to provide authorization to access the data.
- If the user chooses to provide authorization, the system looks up the userId in the authorization table, and checks to see if the conditions in the databaseName, tableName, columName, and value fields would permit access to all of the proprietary data.
- If there is still proprietary data, the system checks the list of user supplied authorizations to see if they would provide access to the proprietary data. In the case of Gemini data, the user supplied authorizations would consist of proposal id and password pairs. The proposal Ids and passwords may be supplied from a user interface configuration file (this will be dealt with in the user interface chapter).

The columns of table authorization are shown in Table 43 on page 109.

The active field allows authorizations to be deactivated, without losing the information required to maintain a history of proprietary data requests. Rows should never by removed from the authorization table, they should be deactivated when they are no longer required. (This could be

FIGURE 19. Data retrieval table relationships



dataset

datasetId	binary(8)	<pk>
archive	char(6)	

recipeInstance

recipeInstanceId	binary(8)	<pk>
recipeId	binary(8)	<fk1,fk2>
datasetId	binary(8)	<fk3>
state	char(1)	
reference	binary(8)	
workload	float	
outputSize	float	
releaseDate	smalldatetime	
dateProcessed	smalldatetime	
parameters	varchar(255)	

requestDatasets

reqnum	integer	<pk,fk3>
datasetId	binary(8)	<pk,fk2>
status	char(1)	<fk4>
authorizationId	integer	<fk1>

requestProcessing

reqnum	integer	<pk,fk1>
datasetId	binary(8)	<pk,fk1>
status	char(1)	<fk3>
recipeInstanceId	binary(8)	<fk2>

requests

reqnum	integer	<pk>
userId	varchar(16)	<fk1>
archive	char(6)	<fk2>
submissionDate	datetime	
status	char(1)	<fk3>
options	binary(16)	<fk2>
completionDate	datetime	
requestSize	integer	

requestOptions

archive	char(6)	<pk>
category	char(8)	<fk2>
option	binary(16)	<pk>
exclude	binary(16)	
default	bit	
label	char(6)	<fk1>

users

userId	varchar(16)	<pk>
pswd	varchar(64)	
lname	varchar(24)	
mname	varchar(24)	
fname	varchar(24)	
title	char(5)	
phone	varchar(20)	
fax	varchar(20)	
inst	varchar(64)	
email	varchar(64)	
addr	varchar(255)	
city	varchar(20)	
province	varchar(20)	
country	varchar(20)	
code	varchar(10)	
language	char(2)	<fk>
creationDate	smalldatetime	
approvalDate	smalldatetime	
lastUseDate	smalldatetime	

authorization

authorizationId	integer	<pk>
active	boolean	
archive	char(6)	
userId	varchar(16)	<ak,fk>
password	varchar(64)	
databaseName	char(30)	
tableName	char(30)	
columnName	char(30)	
value	char(30)	

TABLE 43. Columns of table authorization

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
authorizatio-nId	A unique identifier for this for in the authorization table.	integer	TRUE	TRUE	FALSE
active	Indicates if the authorization is an active entry.	boolean	TRUE	FALSE	FALSE
archive	The archive to which the authorization applies. For the GSA, this value should always be "GEMINI".	char(6)	TRUE	FALSE	FALSE
userId	The user id associated with this authorization.	varchar(16)	FALSE	FALSE	TRUE
password	The password used to authenticate access to proprietary data.	varchar(64)	FALSE	FALSE	FALSE
data-baseName	With the tableName and columnName fields, this iden-tifies the attribute which delimits the scope of the authorization.	char(30)	FALSE	FALSE	FALSE
tableName	With the databaseName and columnName fields, this identifies the attribute which delimits the scope of the authorization.	char(30)	FALSE	FALSE	FALSE
columnName	With the databaseName and tableName fields, this iden-tifies the attribute which delimits the scope of the authorization.	char(30)	FALSE	FALSE	FALSE
value	This is the value that the scope attribute must have for this authorization to apply.	char(30)	FALSE	FALSE	FALSE

relaxed to allow authorizations which have never been used to be deleted after the data has become public.)

The authorization id is generated automatically when a new row is inserted into the table, and remains unchanged indefinitely.

Any user that can provide the password will be permitted to access the data protected by the authorization. If the password is NULL, then the userId field must be non-NULL, and the authorization applies to the identified user.

The user id must either correspond to a user id in the users table Section 3.1.6 on page 115, or be NULL. A non-NULL value indicates the user is permitted to access the proprietary data identified by the tableName, columnName, and value fields. A NULL value in the user field indicates that any user that can provide the correct password may use this authorization.

The databaseName, tableName, columnName and value fields are used to limit the scope of an authorization (to all datasets for a specific proposal for example). The databaseName, tablename, and columnName fields identify an attribute of the dataset, and the value field indicates the value which the attribute must have. If all four of these fields are NULL, then this authorization entry applies to all data supersets in the archive.

The indexes of table authorization are shown in Table 44 on page 110.

TABLE 44. Indexes of table authorization

Name	Comment	Unique	Cluster
authorizationId	This index allows authorizations to be located by authorization id.	FALSE	FALSE
userId	This index would allow all authorizations associated with a user to be found, and would allow non-user specific authorizations to be found, based on the scope qualifier (eg. "find all authorizations where userId is NULL, and proposal id is 'x').	FALSE	FALSE

3.1.2 Table: requestDatasets

The request dataset table stores information about the data supersets associated with user data requests.

This table is populated only by the request creation task. This is to prevent unauthorized access to proprietary data, the content of this table must be protected from unauthorized modification. This table is populated by the request creation task.

Data should remain in this table permanently as a record of the use of the GSA.

The columns of table requestDatasets are shown in Table 45 on page 110.

TABLE 45. Columns of table requestDatasets

Name	Comment	Data Type	Mandatory	Primary	Foreign Key
reqnum	The request number associated with this dataset.	integer	TRUE	TRUE	TRUE
datasetId	The data superset id associated with this dataset. The dataset table in the data processing database is used to map dataset id to dataset name.	binary(8)	TRUE	TRUE	TRUE
status	The status of this dataset. See the text for a description of the status values.	char(1)	TRUE	FALSE	TRUE
authorizationId	The authorization which allowed the user to request this data superset. The value is NULL if the data is public.	integer	FALSE	FALSE	TRUE

The status column describes the current state of a requested data superset. Valid values for the "status" column are:

"R" — The dataset is ready for processing.

"C" — The processing for the dataset is complete.

"E" — An error occurred while retrieving the dataset.

"W" — The dataset has been written to the delivery media.

The indexes of table requestDatasets are shown in Table 46 on page 111.

TABLE 46. Indexes of table requestDatasets

Name	Comment	Unique	Cluster
reqnum	This index allows datasets to be identified by request number and dataset.	FALSE	FALSE

3.1.3 Table: requestOptions

The request options table describes each of the retrieval options available for data requests. The table contains all information required by a user interface to allow the user to select a valid set of request options.

The columns of table requestOptions are shown in Table 47 on page 111.

TABLE 47. Columns of table requestOptions

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
archive	The archive to which the option applies. A value of NULL indicates the option applies to all archives.	char(6)	TRUE	TRUE	FALSE
category	This value categorizes this option. The value would be used by a user interface to organize the options. See the text for a list of categories forseen for the GSA.	char(8)	FALSE	FALSE	TRUE
option	The option's bit pattern. Only one bit should be set in this value. See the text for a description of the options forseen for the GSA.	binary(16)	TRUE	TRUE	FALSE
exclude	This bit pattern identifies any other options which are incompatable with this option. The user interface can use this value to identify invalid option configurations. The value is the logical or of all invalid options. See the text for further details.	binary(16)	TRUE	FALSE	FALSE
default	Indicates if this option is on by default.	bit	TRUE	FALSE	FALSE
label	Used to join with the label table in order to get appropriate labels for this option. The value used here is arbitrary, but should be some mnemonic description of the option.	char(6)	TRUE	FALSE	TRUE

The valid categories in the request options table are:

DATA — The options in this category select the type of data to be retrieved for the requested data.

MEDIA — The options in this category allow selection of the delivery media for the requested data.

COMPRESS — The options in this category describe how the delivered data should be compressed.

New categories will be added as needed.

The exclude column allows each option to identify a set of other options which are incompatable. One example of the use of this column would be for the options in the media category. If only one delivery media is supported for each request, then each of the media options is incompatable with all of the other delivery media options. A user interface could use this field to determine which options must be un-set when a user selects a delivery media option. In order to simplify maintenance of this table, the exclude bit pattern may include the options own bit pattern (it will be ignored).

The request options describe any user requested options for the data request. The option bits can be "ored" together, allowing more than one option to be applied to each request. The valid options planned for Gemini data requests are:

0x00000000000000000000000000000001 — Data is to be delivered on EXABYTE tape (MEDIA category).

0x00000000000000000000000000000004 — Data is to be delivered via the internet (MEDIA category).

0x00000000000000000000000000000008 — Data is to be delivered on CD-ROM (MEDIA category).

0x00000000000000000000000000000010 — Data is to be delivered on DVD (MEDIA category).

0x00000000000000000000000000000100 — Compress the data using gzip (COMPRESS category).

0x00000000000000000000000000000200 — Compress the data using unix compress (COMPRESS category).

0x00000000000000000000000000000400 — Compress the data using hcompress (COMPRESS category).

0x00000000000000000000000000000800 — Compress all data files into a single file using zip (COMPRESS category).

0x00000000000000000000000000001000 — Create a tar file containing all requested data (COMPRESS category).

0x00000000000000100000000000000000 — Deliver raw, unprocessed data for the requested data supersets (in addition to any data processing results) (DATA category).

0x00000000000000200000000000000000 — Deliver all calibration data files appropriate for the requested data supersets (DATA category).

0x00000000000000400000000000000000 — Deliver all environmental data files appropriate for the requested data supersets (DATA category).

0x00000000000000800000000000000000 — Deliver all science programs appropriate for the requested data supersets (DATA category).

0x00000000000000100000000000000000 — Deliver all proposals appropriate for the requested data supersets (DATA category).

0x00000000000000200000000000000000 — Deliver all electronic observing logs for the requested data supersets (DATA category).

New options will be added as needed. These options follow the convention that the low order 8 bytes are used for options which apply to all archives, and the high order 8 bytes are used for archive specific options, however this is a convention only, and is not strictly necessary.

The indexes of table requestOptions are shown in Table 48 on page 113.

TABLE 48. Indexes of table requestOptions

Name	Comment	Unique	Cluster
option	This index allows options to be found based on archive and option id. This index will not be necessary if the option table remains small.	TRUE	FALSE

3.1.4 Table: requestProcessing

The request processing table stores information about the data processing the user has requested to be performed for each data superset in a user data request. Each row in the request processing table represents one data processing step which will be executed, and the results will be delivered to the user. Intermediate data processing required as input to the processing listed in this table is not explicitly listed in this table (for example, if a user requests a processed mosaic image, there may be several re-calibration data processing steps which would not be explicitly listed in the Request Processing table). The full "processing tree" required to produce a result will be available from the data processing catalogue.

The requestProcessing table is populated only by the request creation task. This is to prevent unauthorized acces to proprietary data. The content of this table must be protected from unauthorized modification.

The columns of table requestProcessing are shown in Table 49 on page 113.

TABLE 49. Columns of table requestProcessing

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
reqnum	The request number associated with this processing.	integer	TRUE	TRUE	TRUE
datasetId	The dataset associated with this processing.	binary(8)	TRUE	TRUE	TRUE
status	The status of the data processing. See the text for a description of the valid status values.	char(1)	TRUE	FALSE	TRUE
recipeInstan-ceId	This is a unique identifier for this recipe instance in the data processing catalogue.	binary(8)	FALSE	FALSE	TRUE

The valid values for the status column are:

" " — The processing has not yet been attempted.

"D" — The processing has been successfully completed.

"E" — The processing was attempted, but failed.

The indexes of table requestProcessing are shown in Table 50 on page 113.

TABLE 50. Indexes of table requestProcessing

Name	Comment	Unique	Cluster
reqnum	This index allows data processing to be found by request number and dataset id.	FALSE	FALSE

3.1.5 Table: requests

The requests table stores information about user data requests. There is one row in the table for each user data request.

In order to prevent unauthorized access to proprietary data, the contents of this table must be protected from unauthorized modification. This table is populated by the request creation task.

Data should remain in this table permanently, as a record of the use of the GSA.

The columns of table requests are shown in Table 51 on page 114.

TABLE 51. Columns of table requests

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
reqnum	A unique identifier for each request. This value will be generated automatically when rows are inserted into the table.	integer	TRUE	TRUE	FALSE
userId	The user who submitted the request.	varchar(16)	TRUE	FALSE	TRUE
archive	The archive associated with the data being requested (always "GEMINI" for Gemini data requests).	char(6)	TRUE	FALSE	TRUE
submission-Date	The date and time the request was submittted. This is set automatically when new rows are inserted into the table.	datetime	TRUE	FALSE	FALSE
status	The current status of the request. See the text for a description of the status values.	char(1)	TRUE	FALSE	TRUE
options	Any options associated with the request. See Section 3.1.3 on page 111 for a description of the options used by the GSA.	binary(16)	TRUE	FALSE	TRUE
completion-Date	Date and time the processing of the request is completed. This is set when the notification message is sent to the user.	datetime	FALSE	FALSE	FALSE
requestSize	The number of kilobytes delivered to the user for this request.	integer	FALSE	FALSE	FALSE

The status column describes the current state of a user data request. The status is set initially by the request creation task. Various other tasks in the data retrieval subsystem update the status value as the retrieval processing proceeds. Valid values for the request table "status" column are:

"A" — The request is queued for automatic processing.

"C" — The request was cancelled by the operator.

"D" — The request has been successfully processed.

"E" — An Error occurred while processing the request.

"M" — The request is queued for manual processing.

"P" — The request is being processed.

The indexes of table requests are shown in Table 52 on page 115.

TABLE 52. Indexes of table requests

Name	Comment	Unique	Cluster
reqnum	This index allows requests to be found by request number.	TRUE	FALSE
userid	This index allows requests to be found by user id.	FALSE	FALSE
status	This index allows requests to be found based on their current status.	FALSE	FALSE

3.1.6 Table: users

There is one row in this table for each registered user. Each user is assigned a unique user identifier, which is the primary key in the users table.

Insertions into this table are done by users requesting accounts via the WWW interface, however the new accounts will be inserted in an "inactive" state (i.e. the approvalDate value is NULL). Approval date will be set to a non-NULL value after the account has been examined and approved by authorized GSA staff, using the user maintenance interface. Inactive users will not be permitted to request data from the archives.

The columns of table users are shown in Table 53 on page 115.

TABLE 53. Columns of table users

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
userId	This is the unique user identifier.	varchar(16)	TRUE	TRUE	FALSE
pswd	Password for this user.	varchar(64)	TRUE	FALSE	FALSE
lname	User's last name.	varchar(24)	TRUE	FALSE	FALSE
mname	User's middle name.	varchar(24)	FALSE	FALSE	FALSE
fname	User's first name.	varchar(24)	FALSE	FALSE	FALSE
title	User's title.	char(5)	FALSE	FALSE	FALSE
phone	User's telepone number.	varchar(20)	FALSE	FALSE	FALSE
fax	User's fax number.	varchar(20)	FALSE	FALSE	FALSE
inst	User's home institution.	varchar(64)	TRUE	FALSE	FALSE
email	User's email address.	varchar(64)	TRUE	FALSE	FALSE
addr	User's street address.	var-char(255)	FALSE	FALSE	FALSE
city	City of the user's address.	varchar(20)	FALSE	FALSE	FALSE
province	Provice of the user's address.	varchar(20)	FALSE	FALSE	FALSE
country	Country of the users address.	varchar(20)	TRUE	FALSE	FALSE
code	Postal code of the user's address.	varchar(10)	FALSE	FALSE	FALSE
language	The language for this row in the table. Value values are "EN" - English, or "FR" - French.	char(2)	TRUE	FALSE	TRUE
creationDate	Date the account was created.	smalldate-time	TRUE	FALSE	FALSE

TABLE 53. Columns of table users

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
approvalDate	Date the account was approved. If NULL, this account has not been approved.	smalldate-time	FALSE	FALSE	FALSE
lastUseDate	This is the date and time the user last used the account.	smalldate-time	FALSE	FALSE	FALSE

The users last name will be used as a secondary key by the user maintanance program, to ensure duplicate accounts are not issued.

The password column of the users table must be protected from public access.

The userId is a unique identifier assigned to each archive user.

The indexes of table users are shown in Table 54 on page 116.

TABLE 54. Indexes of table users

Name	Comment	Unique	Cluster
users_id	This index allows user information to be found based on user id.	TRUE	FALSE

3.2 Software

This section describes the various software components of the Data Retrieval subsystem.

3.2.1 User Table Maintenance

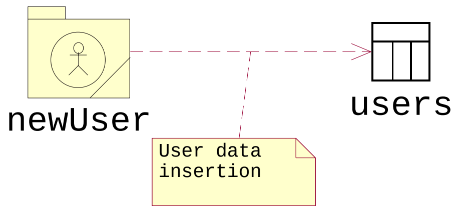
The user table maintenance is done by three tasks:

- A GUI, used by GSA staff for adding new users or modifying existing user data, and for adding or removing authorization for specific users. The GUI is described in Chapter 10, section 3.1 on page 141.
- A command line application which inserts new users into the users table in an inactive state. This application will be usable as a CGI application by a web page. This application is used by the GSA user interface when a user requests an account. This program is described in Section 3.2.1.1 on page 116.
- A command line application which inserts new authorizations into the authorization table. This application is used by the Data Ingest subsystem when adding authorization for new Gemini proposals. This program is described in Section 3.2.1.2 on page 117

3.2.1.1 User Insertion Program

This should be a command line program which inserts new users into the users table. New users should be inserted with the “approvalDate” column set to NULL. The application should have command line options which allow setting of all columns in the users table, except “creation-Date”, and “approvalDate”, which will be set automatically (the long command line should not be a problem, since this program will always be used as a helper task to a GUI. An overview of the User Insertion program is shown in Figure 20 on page 117.

FIGURE 20. User Insertion Overview



The command line for this application would be:

newUser -userId *newUserId* -password *password* -lastname *lastName* [-firstName *firstName*] [-middleName *middleName*] -email *email* [-title *title*] -telephone *telephone* [-fax *fax*] [-institution *institution*] [-address *address*] [-city *city*] [-province *province*] -country *country* [-postalCode *postalCode*] [-language *language*]

The address string will be allowed to contain newlines which will cause the address to be printed on multiple lines of a mailing label. If language is not set, english will be assumed.

Before inserting the new user, the program should verify that:

- The userId is not already in use.
- The language is one of the supported languages.
- Verification of the email address would be nice, but may not be possible.

If any of these verifications fail, an error will be returned by the program, and the user will not be inserted into the users table.

Note: This application could be combined with the GUI described in Chapter 10, section 3.1 on page 141, with the GUI relegated to calling this application.

3.2.1.2 Authorization Insertion Program

This is a command line program which inserts authorization entries into the user authorization table. The application will be used by the Data Ingest subsystem to add an authorization for each Gemini proposal. The proposal authorizations will allow GSA user to have password protected access to proprietary data. An overview of the Authorization Insertion program is shown in Figure 21 on page 118.

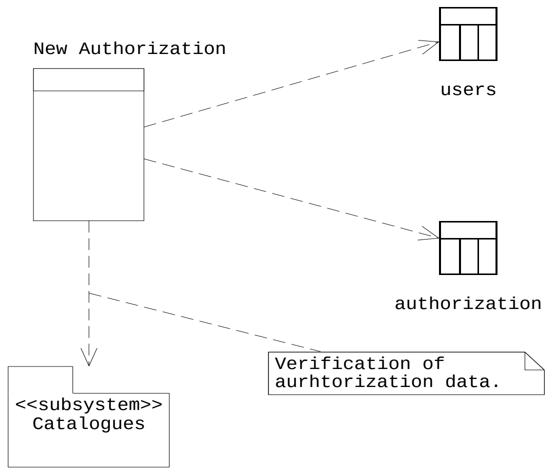
The command line for this application would be:

newAuthorization -archive *archive* [-userId *userId*] [-password *password*] [-databaseName *databaseName* -tableName *tableName* -columnName *columnName* -value *value*]

Before inserting new authorization the program should verify that:

- The userId, if given matches a user id in the users table.
- The archive matches a known archive.

FIGURE 21. Authorization Insertion Overview



- If *password*, *databaseName*, *tableName*, and *columnName* are given, ensure that the database, table, and column exist. Note that a check for the existence of data in the *tableName* which matches the authorization should not be done, since insertion of the authorization may be done before insertion of any data for a proposal.

If any of these checks fail, then an error will be returned, and no authorization will be inserted into the User Authorizations table.

To prevent access to proprietary data, this program must be protected from use by unauthorized users.

Note: This application could be combined with the GUI described in Chapter 10, section 3.1 on page 141, with the GUI relegated to calling this application.

3.2.2 Request Creation

User data requests are created by a command line application which takes as input:

- User id.
- Authorization information.
- A list of request options.
- A list of datasets with associated data processing.

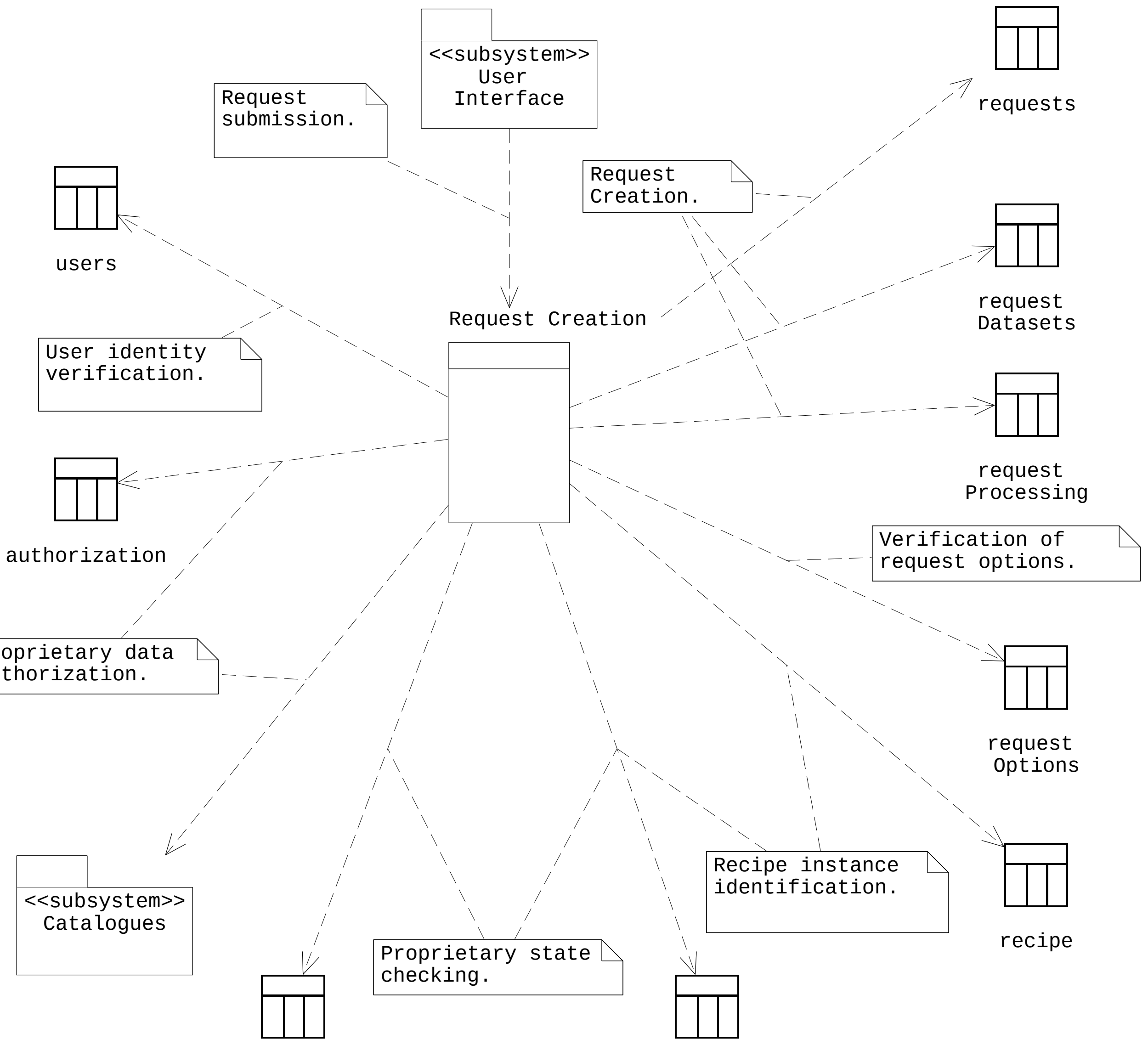
This will be a stand alone application usable by a web user interface. An overview of the request creation task is shown in Figure 22 on page 119.

The command line for this application would be:

newRequest -userId *userId* -password *password* -archive *archive* [-options *options*]

The options are described in the description of the requestOptions table. The options are encoded on the command line as hexadecimal strings. Multiple options can be specified by calculating the logical “or” of the bitmaps representing the selected options.

FIGURE 22. Request Creation Overview



The remainder of the data would be passed as standard input to the program. Specifically, there will be one line in the standard input for each dataset to be retrieved. Each line will contain data-set name and data processing information. The format of each dataset information line would be:

dataset *dataSupersetId* [processing *recipeId* ...]

recipeId is the recipeId value from the data processing recipe table.

In addition, any proprietary data authorization information supplied by the user will be included in the standard input to the program. The format of the authorization information would be:

authorize *authorizationId password* ...

The request creation program must verify the following:

- The user id and password are valid.
- The requested datasets exist in the archive.
- The user is authorized to retrieve any proprietary data in the request. Users will be authorized to retrieve proprietary data if they are priveleged users, or if the necessary proposal passwords are provided in the request authorization information.
- All requested data is available in the archive.
- That any requested data processing is valid, and does not use data the user is not authorized to have. The program will access the Data Processing system to get a list of all datasets used as input to a data processing recipe instance, and must verify that the user is authorized to access the data.
- The request's options are valid and self consistent.

The program will insert one row in the requests table for each data request, one row in the requests data superset table for each data superset requested, and one row in the request processing table for each data processing result requested. The program will use the data processing system to map recipe identifiers into recipe instance identifiers.

3.2.3 Request Processing

The request processing program is a program which runs continuously, is responsible for monitoring the retrieval tables for new data requests, and generating the data processing jobs which will cause new requests to be processed. An overview of the request processing task is shown in Figure 23 on page 121.

The request processing system creates and submits the following data processing requests:

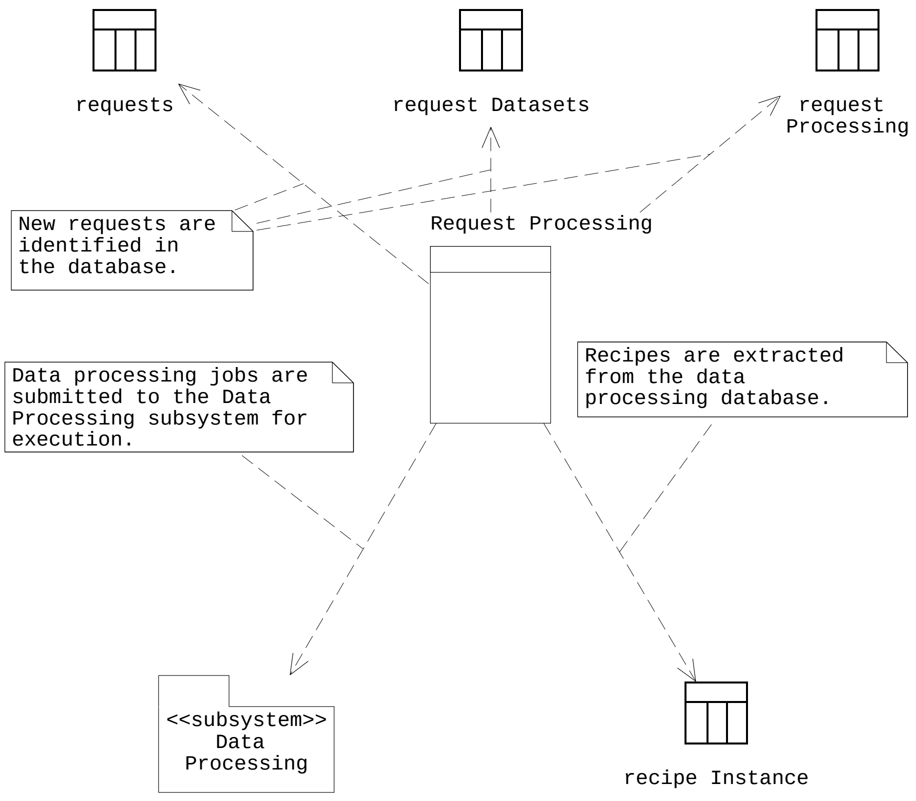
1. Each data processing recipe instance requested by the user will be queued for execution.
2. A wrapper recipe instance will be created for each user requested recipe. The wrapper recipe takes the output from the user requested recipe instance, does any transformations required (e.g. compression), and forwards the data to the bulk storage system for creation of delivery media. The wrapper recipe instance will also retrieve any raw data required for the request. The wrapper recipe instance will perform the appropriate updates to the status in the data superset table.
3. A request recipe is created which waits for the completion of the other recipes, and then sends a notification message to the user indicating that the recipe is complete. The request recipe would also update the request table to indicate that the request is complete.

3.2.3.1 User Media Creation

Data processing recipes generated by the retrieval system will forward data to the Media Creation subsystem. The Media Creation subsystem will then either use the Gemini DHS Storage Server to write data to removable media, or stage data in an FTP directory, as appropriate.

Removable media creation creates some difficult problems with scheduling and disk space utilization. If disk space is a limited resource, then processing of removable media requests should not proceed until there is an operator present to write the removable media, and release disk space used to store the data. The disadvantage of this is that the possibly lengthy processing of data can not proceed until an operator is available.

FIGURE 23. Request Processing Overview


requests
request Datasets
request Processing
New requests are identified in the database.
Request Processing
Data processing jobs are submitted to the Data Processing subsystem for execution.
Recipes are extracted from the data processing database.
<<subsystem>> Data Processing
recipe Instance

Given the low cost of disk space, it is more cost effective to simply ensure that there is ample disk space available. The disk space usage should be monitored to ensure available space remains adequate. There are other solutions to the problem, however they involve automated monitoring of staging disk space and data processing progress, reserving staging space, and other similar techniques. Implementing these more complex solutions would cost more than purchasing the large staging area.

3.2.3.2 Scheduling

The un-processed requests stored in the tables described in Section 3. on page 106 are considered queued for processing. The request processing program contains a scheduling algorithm which selects the order in which data requests will be processed. The goals of the data retrieval scheduling system are:

- Quick response to most data requests destined for FTP retrieval.
- Large data requests do not block small data requests destined for FTP retrieval.
- More than one small data request can be processed simultaneously.
- Requests are processed automatically, except in the case of very large data requests.

The GSA data processing subsystem will support priority levels, and so it should be possible to achieve these goals by assigning priorities to data requests. User data requests could be categorized by amount of data used/returned, the amount of CPU time required, and the delivery media type. Estimates for data and CPU requirements can be made before processing starts. A simple three tier system could be used, categorizing requests as “small”, “medium”, or “large”. All pending data requests could then be submitted to the data processing system, with smaller requests to be delivered via FTP given higher priority. The data processing systems priority and queuing mechanisms will result in the scheduling goals being met. All data requests would be submitted to the data processing system and processed in a first in first out order, within their priority level.

The categorization should be controlled by the contents of a configuration file, which define the boundaries between “small” and “medium”, and “medium” and “large” requests in both CPU time, and data size. An example of the configuration file content would be:

```
sizeLimit 100 Mbytes 500 MBytes
timeLimit 10Min 60Min
```

An automated data retrieval system introduces some complicated disk management issues. Most of these issues most disappear in a disk-space-rich environment. Given the low cost of disk space, it is more cost effective to simply ensure that there is ample disk space available for both FTP storage, and for the staging area for data to be written to removable media.

3.2.3.3 User Interface Interaction With the Requests Database

The user interface will need access to request status information in order to allow users to monitor the progress of data requests, as described in Chapter 3, section 2.1.18 on page 42. The required information can be collected by having data processing recipe instances update the status values in the **requestProcessing** table as each step in the data processing is completed. This may be a somewhat coarse indication of progress when a request consists of a large amount of processing done on a small number of datasets, but it should be adequate in the general case of a moderate amount of processing done on a moderate number of datasets.

4. Resource Requirements

The following sections describe the resource required to operate the GSA data retrieval system.

4.1 Media Writers

At least one writer will be required for each supported delivery medium (CDROM, DVD, and EXABYTE). Past experience with archives at the CADC has shown that users prefer FTP for retrieving data, and so there should only be occasional need for creating removable media for user requests. If this continues to be the case for Gemini, the media writers can be shared with other subsystems (e.g. the bulk data storage system).

If the assumption is that there will only be occasional removable media requests is wrong, it may be necessary to have media writers dedicated to user requests, or it may be necessary to purchase an automated “stacker” system for creating removable media.

4.2 Disk space

The following are the major disk space requirements for the GSA data retrieval system. The relatively low cost for disk space means that many complicated disk rationing problems that arise in an archive system can be effectively solved by simply ensuring that disk space is never in short supply.

4.2.1 Database Storage

The storage requirement for the retrieval databases will be very modest. Five hundred Mbytes of database space should be adequate for several years of operations.

4.2.2 Temporary Work Space

Each of the processing nodes (see Section 4.3 on page 123) will need enough disk space to allow any of the processing steps required by the GSA to be performed on any processing node. Fifty Gbytes of disk space per processing node should be adequate.

4.2.3 FTP Storage Area

All data requests to be retrieved via FTP will have to be stored on disk until the archive user has retrieved the data. The amount of disk space required for this will depend on the maximum number of user requests currently in the FTP area, and the size of the requests. A typical scenario may have 10 user data requests, each containing 10 GMOS images at about 60 Mbytes each. The total space required for this scenario would be about 6 Gbytes. In order to eliminate any possible contention for FTP space, 50 Gbytes should be allocated to the FTP storage area.

An automated system will be set up to periodically remove old requests from the FTP storage area. (The CADC currently advises users that requests will be automatically deleted after one week.)

4.2.4 Removable media staging area.

Data requests destined for removable media (CDROM or EXABYTE) will have to be staged on magnetic disk until the GSA operator writes the data to removable media. The retrieval software must either ration this staging space in an intelligent way, or assume that ample staging space is available, and use a much simpler error handling system to detect when the staging space is full (e.g. simply block all removable media requests until there is space available). A large staging area will also allow the bulk of the processing for a large data request to be done automatically, with operator intervention required only for creation of removable media. The largest data request ever received by the CADC was for 58 Gbytes of data. A staging area of 100 Gbytes should be adequate to store the data for at least one data request of any reasonable size.

The removable media staging space could be shared with the staging space used by other removable media writing tasks (i.e. archive media creation).

4.3 CPU

The GSA will require CPU resources for processing user data requests. The exact amount of CPU power required will depend on the usage of the archive, and the complexity of the data processing being done. These are both unknowns, and so the required CPU resources can only be guessed at. Assume average 30 minutes CPU time per request, a single commodity CPU should be enough to process about 20 user data requests per day, with reasonable response times. (This issue is also covered in Chapter 5, section 5.2 on page 91.

4.4 Sybase database

Very low load on database (less that 1% of a reasonable database server). The Data Retrieval subsystem should be able to easily share a database server with other archive tasks.

The disk space needed for the requests tables will be very small, probably less than 10 Mbytes per year.

4.5 Operator time

The request processing system should run unattended, except:

- Creation of removable media.
- Error handling.
- Handling of exceptional requests: i.e. very large requests or requests which require a lot of space or data processing time should be approved by an operator before they are processed.

4.6 Security

The datasets requested by users should be proprietary information, and so the content of the requests tables will be protected from unauthorized access. Users must be prevented from accessing proprietary data. The features needed to protect proprietary data are noted in the appropriate places in this design.

Created:July 18, 2000
Modified:May 21, 2002

Gemini Science Archive

Chapter 8 Bulk Data Storage Subsystem

1. Introduction

This chapter describes the conceptual design of the GSA Bulk Data Storage subsystem. The Bulk Data Storage subsystem is based on existing CADC systems, and only the required modifications will be described.

The purpose of the Bulk Data Storage Subsystem is to:

- Store the bulk data received from Gemini. This includes storage of backup copies of data to be used for recovery.
- Keep track of where the bulk data for each dataset is stored, including where the data can be found on backup media (see Section 4.1 on page 129, Section 5.2 on page 134, and Section 5.3 on page 134).
- Make bulk data available to other subsystems of the GSA (see Section 5.1 on page 133).
- Provide facilities to copy both the “live” archive and the backup media to new storage media types as technologies evolve (see Section 5.5 on page 135).

The Bulk Data Storage subsystem will be heavily based on the existing CADC AD library (Archive Directory), the Gemini DHS Storage Server, and existing practices within the CADC. Figure 24 on page 126 shows a graphical overview of the components of the Bulk Data Storage subsystem as they relate to each other and to the other subsystems of the GSA. The shaded area denotes the components which are part of the Bulk Data Storage subsystem.

2. Requirements on the Bulk Data Storage Subsystem

Table 55 on page 126 lists the software requirements which at least partially apply to the GSA Bulk Data Storage System. Each requirement is followed by a short description of how the requirement will effect the Bulk Data Storage Subsystem. See [6] for a complete description of these requirements.

3. Archive Storage Media Trade Studies

This section justifies the choices made when choosing the implementation strategy for the GSA bulk data storage subsystem.

3.1 Bulk Data Storage Media

The basic categories of storage media available for storing bulk data in an archive are:

FIGURE 24. Bulk Data Storage Subsystem overview

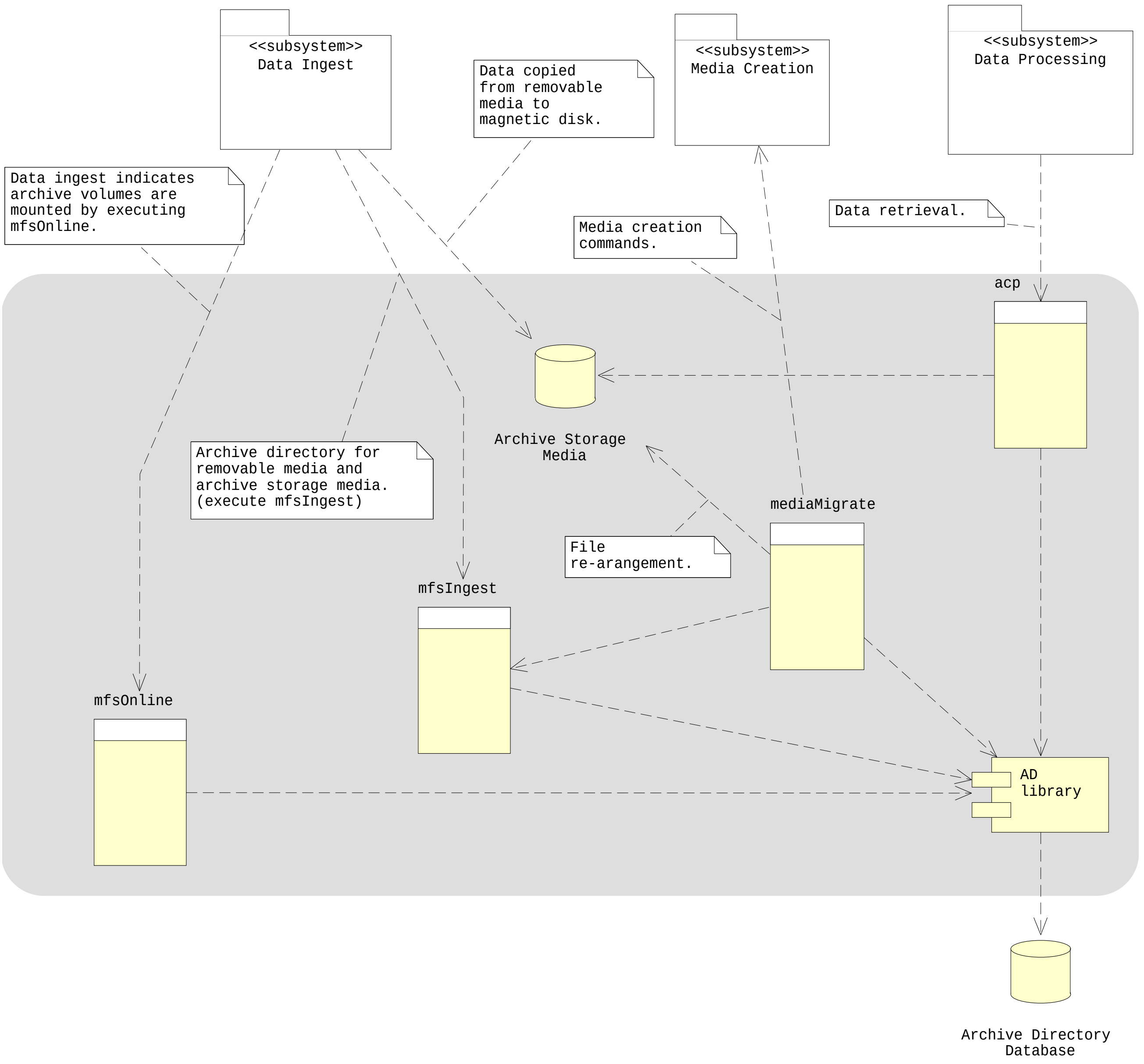

<<subsystem>>
Data Ingest
Data copied from removable media to magnetic disk.
<<subsystem>>
Media Creation
<<subsystem>>
Data Processing
Data ingest indicates archive volumes are mounted by executing mfsOnline.
Media creation commands.
Data retrieval.
acp
Archive Storage Media
Archive directory for removable media and archive storage media. (execute mfsIngest)
mediaMigrate
File re-arangement.
mfsIngest
mfsOnline
AD library
Archive Directory Database

TABLE 55. Bulk data storage subsystem requirements

SR1.1 The GSA shall archive all data. All data received from Gemini will be available from the bulk data storage system. The storage system will have sufficient capacity to store all data.
SR1.3.2 Data processing shall proceed promptly. Science and calibration data will be stored on-line, allowing the data to be accessed without manual operator intervention.

TABLE 55. Bulk data storage subsystem requirements

SR4.2 Maximum and average requirements for raw data ingest. The bulk data storage system will be able to accept new data at the required rates. This includes retrieval of data for automatic processing (preview, derived descriptors, derived objects, etc.).
SR4.3 Maximum and average requirements for data retrieval by users. The bulk data storage system will be able to supply data for user requests at the required rates.
SR4.6 Data for Internet retrieval should be available promptly. Data must be stored on-line. Data must be available reasonably quickly.
SR5.3 Data shall be electronically secure. Provide tools to create backup media as necessary. Failures and recoveries must be incorporated. Proprietary data will be secure from unauthorized electronic snooping.
SR5.7 GSA operations staff shall incorporate evolving technologies into the GSA. It will be necessary to periodically copy both the “live” archive, and backup media to new technology. The Bulk Data Subsystem will provide facilities to do the copying. There must be no interruption in service from the Bulk Data Subsystem while the copying and media upgrades take place.

off-line — Data are stored on a shelf, and are only available through operator intervention when there is a need for the data. If off-line storage is used, software must be available to prompt the operator to mount appropriate media, and to coordinate the execution of the requesting task with the availability of the data.

on-line — All of the data is stored on spinning magnetic disk.

near-line — The data is stored on a jukebox, and loaded automatically by jukebox robotics when needed. Superficially, the data appears to be on-line, however there are important differences between on-line data and near-line data:

- It takes between 10 and 30 seconds for the jukebox to put a volume into a reader and make it available for reading.
- The readers used in jukeboxes have considerably slower data transfer rates compared to magnetic disk.
- Jukeboxes have a small number of media readers (four in the jukeboxes used by the CADC). If there are fewer volumes in use than the number of readers available in the jukebox, then data throughput is limited primarily by the performance of the readers. When the number of volumes in use exceeds the number of readers, the jukebox begins “thrashing” — loading and using each volume for a short period of time before unloading the volume to load another. When a jukebox is thrashing, the performance becomes limited by the jukebox robotics. Thrashing also causes more wear-and-tear on the jukeboxes.
- If there is more than one file to be retrieved from each different volume, the order in which the files can effect performance. Reading 10 files from one volume, followed by reading 10 other files from a second volume can be significantly more efficient than reading the same 20 files in random order.

The costs associated with each of these options are listed in Table 56 on page 128. Only CD-ROM and DVD-ROM are considered for off-line storage. There are other options, however the other options are not “consumer” products, and so the costs are significantly higher. It should also be noted that the cost of on-line storage is dropping faster than near-line storage, and so on-line storage is becoming an increasingly attractive option. Note that the costs given in Table 56 are those in effect in early 2001, and are subject to change, for instance:

- The cost of blank CD-ROM media is dropping slowly.
- The cost of blank DVD-ROM media is dropping moderately quickly.
- Double sided DVD-ROM may soon be available, nearly halving the hardware cost, but using media with an unknown cost.
- Magnetic disk costs is dropping steadily.

TABLE 56. Total cost of ownership of archive storage media (USD)

Category	Need	Cost
off-line	CD-ROM	Media $0.65/GB
		Operator: To meet the availability requirement, operators would have to be available 24 hours a day.
		Software to manage data retrieval
	DVD-ROM	Media $6.5/GB
		Operator: To meet the availability requirement, operators would have to be available 24 hours a day.
		Software to manage data retrieval
near-line	CD-ROM	Media $0.65/GB
		Jukeboxes $44/GB
		Jukebox host system $14/GB
		Jukebox driver software $16/GB
		Maintenance
		Software to manage data retrieval
	DVD-ROM	Media $6.5/GB
		Jukeboxes $6/GB
		jukebox host systems $2.5/GB
		Jukebox driver software $2/GB
		Maintenance
		Software to manage data retrieval.
on-line	RAID disk array	Media $13/GB
		Host computer system $2.5/GB
		Maintenance
		No additional software to manage data retrieval

The requirement that GSA users have 24 hour a day access to data effectively eliminates the off-line media option. The cost of having an operator available at the archive site at all times is far higher that the approximately $20000/year cost of storing Gemini data on-line or near-line.

Although the costs of the near-line DVD-ROM and on-line storage listed in Table 56 are essentially the same, we have selected on-line storage for the Gemini Science archive for the following reasons:

1. The cost of on-line storage is currently about the same as the cost of near-line storage.
2. Historically, the cost of on-line storage has been dropping faster than near-line storage. If this trend continues, on-line storage will be significantly cheaper by the end of 2001.

3. The performance of on-line storage is significantly better than near-line storage (although properly managed jukeboxes would meet GSA performance requirements).
4. The near-line storage option would need software to manage data retrieval, in order to utilize the jukeboxes efficiently. This software adds complexity and cost to the GSA software development.
5. Upgrading to newer, higher density, cheaper storage options will be necessary with all of the storage options, however this upgrade path is easier and cheaper with magnetic disk media than with any of the other options.

The GSA will use off-line CD-ROM and/or DVD-ROM as backup media for the on-line archive. The choice between CD-ROM and DVD-ROM will have to take into account the following factors:

- The difference in cost between the media.
- The handling costs required to write the media.
- The handling costs required to ingest the media into the DHS system.

Gemini is currently writing CD-ROM for delivery to the GSA as archive media. Since the costs of purchasing and writing the archive media dominate the total cost of using a media type, and since these costs are born by Gemini, we expect Gemini will choose when to switch to DVD-ROM as the archive delivery media type.

4. Database Schema

This section describes the database tables used by the GSA bulk data system.

4.1 Archive Directory Database

The CADC has an Archive Database (AD) which is responsible for keeping track of where files can be found. The existing AD database is primarily geared to supporting CD-ROM, DVD-ROM, and tape archive media. The AD database does support magnetic disk media, however magnetic disks are currently only used for small, quickly changing datasets. The AD support for magnetic disks will have to be upgraded to support the use of magnetic disk as a primary archive media, and to integrate magnetic disk storage with the CD-ROM and DVD-ROM AD storage.

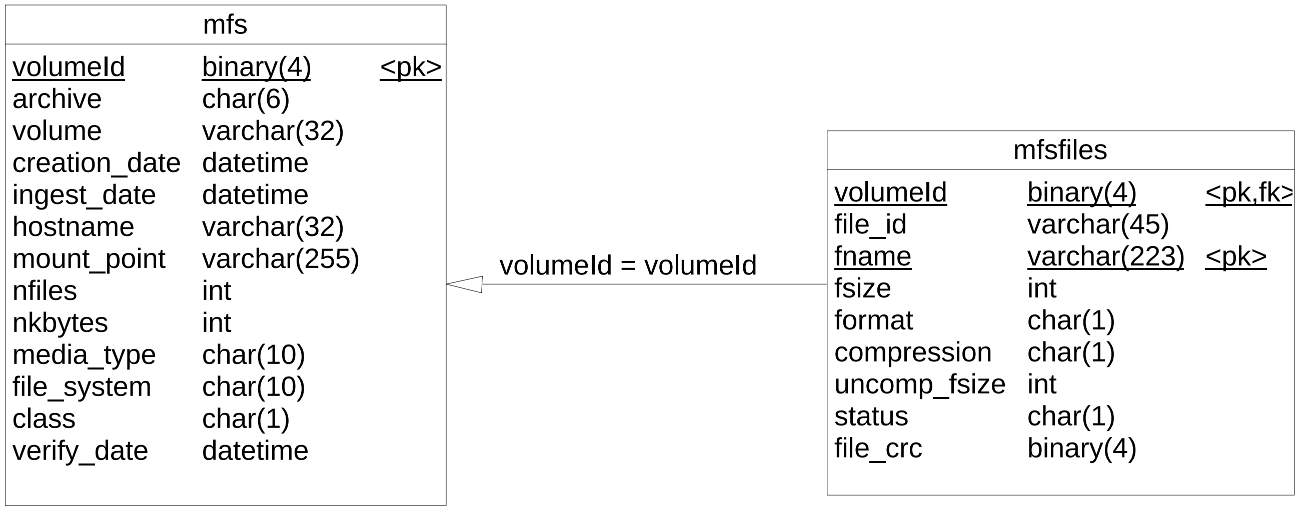
The AD database will use two tables to store information about GSA files:

- The mfs table stores information about archive volumes which are on file systems which can be "mounted". For the GSA, we will copy each of the removable media to a magnetic disk, and each directory will be treated as a separate volume by AD. The schema of the mfs table is described in Section 4.1.1 on page 129. Each of the backup optical media will also be ingested as a volume in the AD tables.
- The mfsfiles table contains information about all of the files on each of the volumes listed in the mfs table. The schema of the mfsfiles table is described in Section 4.1.2 on page 132.

4.1.1 Table: mfs

The mfs table contains information about each mountable file system volume known to the archiving system. Each volume is represented in one row in the mfs table.

FIGURE 25. Bulk data storage table relationships



The media types supported by these tables will be magnetic disk file systems, CDROMs with ISO 9660 file systems, DVDs with ISO 9660 file systems, and DVDs with UDF file systems. These tables can be extended to support any media which can be mounted as a Unix file system.

The columns of table mfs are shown in Table 57 on page 130.

TABLE 57. Columns of table mfs

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
volumeId	This is a unique identifier used to identify each volume.	binary(4)	TRUE	TRUE	FALSE
archive	The archive associated with the volume. For Gemini data, this will always be "GEMINI".	char(6)	TRUE	FALSE	FALSE
volume	The name assigned to the volume.	varchar(32)	TRUE	FALSE	FALSE
creation_date	The creation date of the volume. See the text for a description of this value.	datetime	TRUE	FALSE	FALSE
ingest_date	The date the volume was last ingested into the archive database.	datetime	TRUE	FALSE	FALSE
hostname	If the volume is mounted, this is the name of the host computer that has mounted the volume. If the volume is not mounted, this value will be NULL.	varchar(32)	FALSE	FALSE	FALSE
mount_point	The mount point of the volume on the host computer. If the volume is not mounted, this value will be NULL.	var-char(255)	FALSE	FALSE	FALSE
nfiles	The number of files on the volume.	int	TRUE	FALSE	FALSE
nkbytes	The number of kilobytes on the volume. This should be the sum of the fsize column of the mfsfiles table, for all files on this volume.	int	TRUE	FALSE	FALSE
media_type	The media type of the file system. See the text for a description of the media types.	char(10)	TRUE	FALSE	FALSE

TABLE 57. Columns of table mfs

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
file_system	The file system format. See the text for a description of the file system formats.	char(10)	TRUE	FALSE	FALSE
class	The class indicates the type of activity allowed on the file system. See the text for a description of the volume classes.	char(1)	TRUE	FALSE	FALSE
verify_date	This is the date the volume was verified against the content of the archive catalogues.	datetime	FALSE	FALSE	FALSE

The class column indicates the kind of activity expected on the volume. The supported classes are:

"S" — Static. The content should never change. This is the only class supported for DVD and CDROM, but it may also apply to magnetic disks.

"I" — Incremental. For this class, files may be added to the file system, but existing files should not be changed or deleted. This class only applies to magnetic disks.

"D" — Dynamic. For this class, new files may be added, existing files may change, but files should not be deleted. This class only applies to magnetic disks.

"F" — Free. There are no restrictions on how the data in the file system is manipulated. New files may be added, existing files may change, and existing files may be removed. This class applies to magnetic disks.

The different classes will be used to allow detection of invalid modes of operation by the archive maintenance software. For example, if an archive's primary storage is on magnetic disk, the file systems may have class static to indicate that any changes to the files should be reported. If on the other hand, a magnetic disk is being used to accumulate data to be written to optical media, then a class of "F" would be used to indicate that files will appear and disappear as new data arrives and as optical media are created.

The creation date column contains the date the volume was created. For magnetic disk file systems, this will be a best guess at when the file system was created, but because the creation dates of a directory are changeable, it can only be a guess.

The file system types supported are:

"ISO9660" — This is the file system typically used on CDROM disks, and some DVD disks.

"UDF" — This is the file system typically used on DVD disks.

"UFS" — This is the file system used for magnetic disk file systems.

Note that the system does not preclude use of the above file systems in unusual contexts, for example it is possible to copy a ISO9660 CDROM disk image to magnetic disk, and mount the disk image as an ISO9660 file system.

Known media types are:

"CD" — A CDROM disk.

"DVD" — A DVD disk.

"MD" — A magnetic disk directory.

The indexes of table mfs are shown in Table 58 on page 132.

TABLE 58. Indexes of table mfs

Name	Comment	Unique	Cluster
mfs_volume	This index allows file systems to be found by archive and volume name.	TRUE	TRUE
mfs_crc	This index allows file systems to be found based on the CRC volume id.	TRUE	FALSE

4.1.2 Table: mfsfiles

The mfsfiles table stores information about each file known to the archiving system. There will be one row in the mfsfiles table for each copy of a file. If a file appears on more than one volume, then there will be a row for each copy of the file, and the latest copy of the file will be marked as the "current" copy.

The columns of table mfsfiles are shown in Table 59 on page 132.

TABLE 59. Columns of table mfsfiles

Name	Comment	Data Type	Manda-tory	Primary	Foreign Key
volumeId	This is the unique identifier for the volume containing the file. The volumeId value is used when joining with the mfs table.	binary(4)	TRUE	TRUE	TRUE
file_id	The file ID for this file. Each distinct file in an archive will have the same file ID, regardless of where the file is stored.	varchar(45)	TRUE	FALSE	FALSE
fname	The path of the file on the volume, starting from the root of the mountable file system.	var-char(223)	TRUE	TRUE	FALSE
fsize	The size of the file as it is stored on the file system.	int	TRUE	FALSE	FALSE
format	The format of the file. See the text for a description of the formats.	char(1)	TRUE	FALSE	FALSE
compression	The type of compression used on the data. See the text for a description of the compression types.	char(1)	TRUE	FALSE	FALSE
uncomp_fsize	The size the file would have if it was not compressed.	int	TRUE	FALSE	FALSE
status	The status of the file. See the text for a description of the status values.	char(1)	TRUE	FALSE	FALSE
file_crc	A CRC (Cyclic Redundancy Check) of the file's con-tents. This value is used to ensure that the file has not been unexpectedly changed.	binary(4)	FALSE	FALSE	FALSE

Valid compression types are:

"G" — The gzip program was used to compress the file.

"U" — The UNIX compress program was used to compress the file.

"C" — The compfits program was used to compress the file.

" " — The file is not compressed, or the compression type is not known.

Other compression types may be added if they prove to be useful in the archive.

The format types used for the Gemini archive are:

"F" — The file is a FITS file.

" " — The file type is unknown.

Valid values of the status column are:

"C" — This copy of file is the "current" copy, and should be used in preference to other copies (only one copy of a file should have the "C" status.

"I" — There is a better copy of this file somewhere, which should be used in preference to this copy. The best copy of the file with be flaged with a "C" status.

"E" — This copy of the file is corrupt.

The indexes of table mfsfiles are shown in Table 60 on page 133.

TABLE 60. Indexes of table mfsfiles

Name	Comment	Unique	Cluster
mfsfiles_crc	This index allows files to be found by volume id and file name.	TRUE	TRUE
mfsfiles_file_id	This index allows files to be found based on file id and status.	FALSE	FALSE

5. Software

This section describes the software required to support bulk data retrieval.

5.1 Data Access

The data processing system would be the only GSA subsystem that should be requesting data. All data destined for user requests should be processed through the data processing system. If all data is on-line, data access can be through the UNIX rcp program, with no special data management software needed. A wrapper script called “acp” will be created which takes as input an archive name, a source file id, and a destination path. The script will locate the file in the archive, and use rcp to copy the file to the destination path. The command line for the acp program will be:

acp *archive fileid destinationPath*

This scheme would be more complicated to allow for media which is not on-line (note that if all Gemini data is on-line, support of off-line media will not be a requirement for Gemini). To properly support off-line or near-line media, requests for data files would have to be sent to the DHS

Data Server. The Data Server would have to be changed to organize data requests into an optimal order. The "optimal" order for data retrieval would balance the need to minimize disk swaps, the need to allow high priority data processing jobs to proceed before low priority jobs, and the need to allow data processing jobs to finish.

The problems associated with ignoring the limitations of the jukeboxes supporting near-line media also exist for magnetic disks, but on a completely different scale, and we feel that the problems can be ignored. When the number of simultaneously active volumes in a jukebox exceeds the number of readers in the jukebox, the time to switch "contexts" from reading one volume to reading the next has to include the time to load a new volume into a reader, or about 10 seconds. The equivalent context switching time with magnetic disks is the disk seek time, which is about 10 milliseconds. The number of available "readers" per volume available for magnetic disks is also much higher than the number of readers available in a jukebox — with current technology, there will be about 20 volumes (copies of DVDs) per magnetic disk, compared to 700 DVD volumes in a jukebox with 4 readers.

Note: There are also data processing design implications for media which are not on-line:

- *Data processing jobs would request files from a data server.*
- *Data processing jobs would have to be suspended when one or more data files are not immediately available.*
- *Blocked data jobs should not cause unused CPU cycles if other runable jobs are available.*
- *When files become available, the state of the job would have to be changed to "runable".*
- *There may be a very large number of jobs suspended waiting for files.*
- *Jobs would be "un-suspended" in batches, possibly with a mix of high and low priority jobs.*
- *After being "un-suspended", high priority jobs must proceed before low priority jobs.*
- *Sometimes, required files may not be known until after processing starts.*

5.2 AD library

The AD library is an existing CADC library, used by applications when accessing the AD tables described in Section 4.1 on page 129. The AD library will require the following changes:

- Support the use of magnetic disk volumes in the mfs tables.

5.3 mfsIngest

This is an existing CADC program used to examine the contents of an archive volume, and populate the AD tables. mfsIngest will need the following changes:

- A CRC needs to be calculated for each file on the volume, and inserted into the mfsfiles table.
- Support needs to be added for magnetic disk volumes. This includes support to ensure the rules described in the **class** column of the **mfs** table are followed.

5.4 mfsOnline

This is an existing CADC program which is used to update the AD tables to indicate that a volume is mounted or not mounted. When volumes are on-line, the archive systems assume that the data can be accessed without operator intervention. When volumes are off-line, the archive systems assume that operator intervention is required before the data on the volume can be accessed. This program should not require any changes to support the GSA.

5.5 mediaMigrate

As media types become obsolete, the cost of maintaining the obsolete media tend to increase relative to the costs of using new, higher density archive media. This cost includes the physical storage of the media, the handling required to make use of the media, and the cost of maintaining readers to access the data. Eventually, the cost of maintaining the old media will exceed the cost of migrating old data to new media. This process will apply to both the on-line magnetic disks, and to the off-line backup media.

The mediaMigrate program will help manage the migration process, and will ensure all data is copied to the new media, and that obsolete volumes are removed from the system. This software does not need to be available until the initial backup media (CD-ROM) become obsolete, or if Gemini switches to a delivery media which is not appropriate for use as a backup (hot swappable hard disks, or the internet) and so detailed design and development of this software can be deferred.

The mediaMigrate application will perform the following tasks:

- Find files that need to be migrated to a new media and send the files to the storage server.
- Identify obsolete volumes that can be removed from the Bulk Data system tables. Obsolete volumes are defined as volumes whose entire content is available on the new media, and which are identified by archive staff as being of no value. (Some media may be kept for a time as a second backup copy, but these media have value, and should remain in the system tables). Obsolete volumes will be removed from AD tables, and archive operators will be prompted to discard the volumes from physical storage.
- Add new organization to ad tables.

6. Hardware requirements

This section describes the hardware required to support the GSA bulk data storage system.

6.1 Magnetic disk storage

Currently, there are two reasonable options for supplying the magnetic disk storage space required for the GSA:

- “network appliances”, which would connect to a LAN, and provide network mountable file systems.
- “RAID” systems which connect to a host system using SCSI interfaces.

If the “RAID” option is selected, a server system will be required to support the RAID array. The CADC is currently experimenting with both of these magnetic disk storage options, and the decision on the best approach should be delayed as long as possible.

Both types of devices are currently available with .5 to 1 Terabytes storage capacity.

If the current trend of increasing magnetic disk capacities continues, then the number of storage devices needed to store the GSA data should quickly reach an upper limit. The total cost of maintaining many older, lower capacity devices will exceed the cost replacing the devices with a smaller number of new higher capacity devices.

6.2 CD-ROM/DVD-ROM Storage

CD-ROM or DVD-ROM will be used to transport data between Gemini and GSA, and will also be used as a backup for the magnetic disk primary storage. An off-site backup is not necessary, since Gemini will also have a complete copy of the data.

The GSA needs access to a DVD-ROM jukebox for the following reasons:

- Data is expected to arrive from Gemini in batches of several disks (up to 90 CDs or 12 DVDs, based on a data rate of 4 Gigabytes per day (see [7]), and two weeks between disk shipments). It will be much easier for GSA operators to copy the data to magnetic disk of the entire batch can be inserted into a jukebox.
- If one of the magnetic disk storage devices fails, it will be necessary to copy several hundred CDs and/or DVDs to a new storage device. Putting the media into a jukebox will make recovery much faster, and will provide an alternate data source for archive operations until recovery is complete
- A DVD jukebox should be used rather than a CD jukebox because:
  - DVD readers support also support reading CD-ROM volumes.
  - The cost difference is negligible.
  - Gemini may switch from CD-ROM to DVD-ROM as the archive distribution media.

The jukebox could be shared with other CADC archives.

6.3 Estimated storage needs.

Based on the numbers given in [7], the storage needs of the GSA will range between .5 and 1.5 Terabytes per year, given the initial instrument suite.

Created:January 3, 2001
Modified:May 21, 2002

Gemini Science Archive

Chapter 9 Media Creation

1. Introduction

This chapter describes the conceptual design of the GSA Media Creation Subsystem. This system is based on the Gemini DHS Storage Server.

The purpose of the Media Creation Subsystem is to:

- Create removable media for user requests.
- Create removable media for migration of archive backup media.

Figure 26 on page 138 shows a graphical overview of the components of the Media Creation Subsystem, as they relate to each other and to the other subsystems of the GSA. The shaded area denotes the components which are part of the media creation subsystem.

2. Requirements on the Media Creation Subsystem

Table 61 on page 138 list the software requirements which at least partially apply to the GSA Media Creation subsystem. Each requirement is followed by a short description of how the requirement will affect the Media Creation subsystem. See [6] for a complete description of these requirements.

3. Database Schema

This section describes the database tables used by the GSA Media Creation system.

3.1 Archive Media Database

The archive media database supports creation of removable media for both the archive and for archive users. The archive media database is described in the Gemini DHS documentation [2]. The table structure should not need modification to support the GSA.

4. Software

This section describes the software required to support bulk data retrieval.

4.1 AM library

The AM library is an existing CADC library, used by applications when accessing the AM tables described in Section 3.1 on page 137. The library should not require any changes.

FIGURE 26. Media Creation Subsystem overview

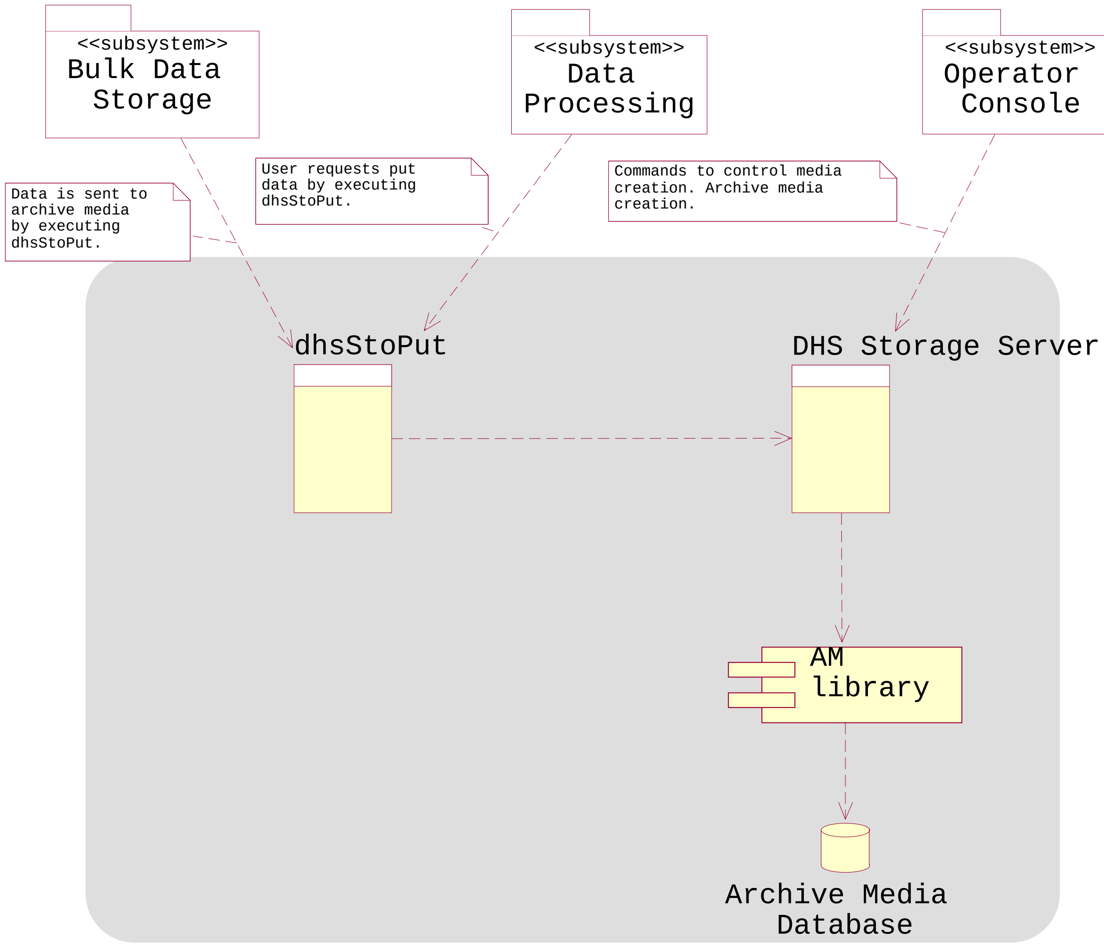

<<subsystem>>
Bulk Data Storage
<<subsystem>>
Data Processing
<<subsystem>>
Operator Console
Data is sent to archive media by executing dhsStoPut.
User requests put data by executing dhsStoPut.
Commands to control media creation. Archive media creation.
dhsStoPut
DHS Storage Server
AM library
Archive Media Database

TABLE 61. User interface requirements

SR3.4.1 Users shall retrieve data via Internet, EXABYTE, or CD-ROM. Provide facilities to allow user requests to be written to supported media.
SR5.3 Data shall be electronically secure. Provide tools to create backup media as necessary.
SR5.7 GSA operations staff shall incorporate evolving technologies into the GSA. Provide facilities to create new backup media.

4.2 DHS Storage Server

The Storage Server described in the Gemini DHS documentation [2] will be used for creating both user and archive media. No changes should be required to support the GSA, except possibly to automate the media creation task, which is needed to support an automated stacker/labeler.

4.3 dhsStoPut

The **dhsStoPut** application is an existing application, used to send data files to the DHS storage server when they are to be written to media. The **dhsStoPut** application could be used by the data processing system when creating user media, or to incorporate processed data files into the archive bulk data store. **dhsStoPut** may also be used by the **mediaMigrate** application described in Section 5.5 on page 135. **dhsStoPut** should not require any modification.

5. Hardware requirements

This section describes the hardware required to support the GSA bulk data storage system. This hardware should be shared with other archives at the archive centre, and so the hardware should be either already available, or available on a shared cost basis.

CD writers — CD writers will be necessary for creating media for delivery to archive users.

DVD writers — DVD writers will be necessary if the GSA upgrades the backup storage media to DVD. DVD may also be used as a delivery media for archive users.

tape writers — An EXABYTE tape writer will be necessary for creating media for delivery to archive users.

stacker/labeler — A stacker/labeler for writing CDs and DVDs would allow automation of the media creation process. A stacker/labeler is optional, but would be very useful if there are a large number of requests for data on CD or DVD, or during backup media migration. A stacker/labeler would reduce operator time and produce better looking media.

host computer — Creating CDs and DVDs is much less likely to fail if the writer host system is dedicated to doing the writing.

Created:July 18, 2000
Modified:August 31, 2001

Gemini Science Archive

Chapter 10

Operator Console

1. Introduction

This chapter describes the design of the GSA Operator Console. The operator console will provide GSA operators with a user-friendly interface to the GSA operations. The functions that an operator will be able to perform using the operator console are:

- Monitor GSA status, including the status of all of the other subsystems of the GSA.
- Monitor the resource usage of the GSA, including CPU usage, disk space usage, and database space usage.
- Control the creation of removable media for archive users and for archive media migration.
- Provide an interface to maintain the list of GSA users and the authorizations to retrieve proprietary data.

2. Requirements on the Operator Console

There are no software requirements directly associated with the Operator Console, since the console supports the GSA operations as a whole, without directly addressing any software requirement.

3. Operator Console Design

The functionality required from the operator console is dependant on the design of the other GSA subsystems. Because of this, most of the design of the operator console will be deferred until the detailed design of the GSA.

The Operator Console will be similar to the Gemini DHS console, and since some of the functionality required from the operator console is already available in the DHS console, parts of the DHS console may be incorporated into the GSA operator console. The functionality needed by the Operator console which is already available in the DHS console includes:

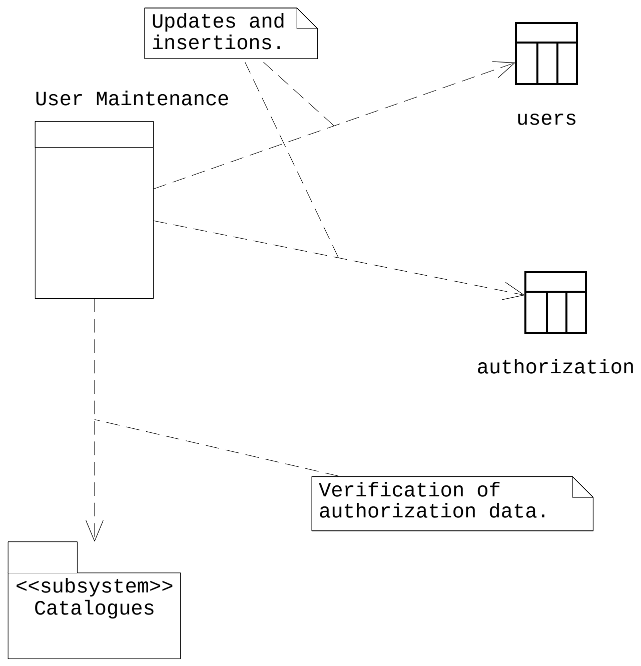
- Controlling creation of removable media for user and archive media.
- Monitoring disk and database space use.

The DHS console also has facilities for monitoring the state of the DHS subsystems which could be extended to the GSA subsystems.

3.1 User Table Maintenance GUI

An overview of the user table maintenance GUI is shown in Figure 27 on page 140. The GUI will be implemented as HTML web pages, restricted to be used only by GSA staff.

FIGURE 27. User Table Maintenance Overview



The user table maintenance GUI will have the following functions:

- Allow creation new of rows in the users table, setting all columns except creation date and approval date to values supplied through the GUI (creation date and approval date will be set to the current date and time).
- Allow viewing of user data from the users table, queried by userId, or last name.
- Allow modification of any columns of the users table, except userId, creation date, and approval date.
- Allow verification that users requesting new accounts do not already have accounts.
- Allow activation of users inserted by the command line application described in Chapter 7, section 3.2.1.1 on page 114, by setting the approval date to the current date and time.
- Allow removal of a user when an account is not approved.
- Allow insertion of rows into the user authorization table, allowing the operator to set “archive”, “userId”, “password”, “tableName”, “columnName”, and “value” columns. The interface should verify that a valid authorization row is created, as described in Section 3.2.1.2 on page 115.
- Allow deactivation of authorization table rows by setting the “active” column to false.
- Allow the password for an authorization to be changed.
- Allow viewing of current authorizations.