305-EED-001

EOSDIS Evolution and Development (EED) Contract

Release 8.0 Segment/Design Specifications for the EED Contract

March 2011

Raytheon Company
Riverdale, Maryland

This page intentionally left blank.

Release 8.0
Segment/Design Specifications for the EED Contract

March 2011

Prepared Under Contract NNG10HP02C
CDRL Item #023

RESPONSIBLE ENGINEER

3/22/2011

Evelyn Nakamura, Chief Engineer — Date
EOSDIS Evolution and Development (EED) Contract

SUBMITTED BY

03/22/11

Christopher Durbin, Task Manager — Date
EOSDIS Evolution and Development (EED) Contract

Raytheon Company
Riverdale, Maryland

This page intentionally left blank.

Preface

This document is a formal contract deliverable. It requires Government review and approval within 45 business days. Changes to this document will be made by document change notice (DCN) or by complete revision.

Any questions or proposed changes can be addressed to:

Data Management Office
The EED Contract Office
Raytheon Company
5700 Rivertech Court
Riverdale, MD 20737

Revision History

Document Number	Status/Issue	Publication Date	CCR Number
305-EED-001	Original	March 2011	11-0023

This page intentionally left blank.

Abstract

The Release 8.0 Segment/Design Specification is an overview description of the EED Project. The functionality of the ECS software is described at the Subsystem, Computer Software Configuration Item (CSCI), Computer Software Component (CSC), and Process levels. Architecture and context diagrams illustrate the process interconnections within the ECS CSCIs and the external connections to other CSCIs, subsystems, and specified segment interfaces. Interface event description tables describe the data, messages, notifications, or status information that occurs at each level of functionality within the ECS. A basic description of the Commercial Off The Shelf (COTS) software and hardware used in ECS is included.

The high-level design in this document is the level of information derived from requirement sources, and used by the development team to complete the ECS design implementation for a software system at an 8.0 state of maturity.

Keywords: Release 8.0, Overview, SDPS, CSMS, Design, Detailed Design, Subsystem, Architecture, Software, Hardware, Object Oriented, Security, Gateway, Reports, User Interface and GUI.

This page intentionally left blank.

Contents

1. Introduction

1.1 Purpose and Scope 1-1

1.2 Document Organization 1-1

2. Related Documentation

2.1 Parent Documents 2-1

2.2 Applicable Documents 2-1

2.2.1 Other Related Documents and Documentation 2-1

2.3 ECS Tool Descriptions 2-2

2.3.1 Rational Rose 2-2

2.3.2 ClearCase Baseline Manager Configuration Management Tool 2-3

3. System Description

3.1 Mission and Release 8.0 Objectives 3-1

3.1.1 Release 8.0 Capabilities 3-1

3.2 Release 8.0 Architecture Overview 3-8

3.2.1 Release 8.0 Context Description 3-12

3.2.2 Release 8.0 Architecture 3-14

4. Subsystem Description

4.1 Data Server Subsystem Overview 4-11

4.1.1 Archive Inventory Management Software Description 4-17

4.1.2 DSS Error Handling and Processing 4-45

4.1.3 DSS Data Stores 4-46

4.2 DPL Ingest Subsystem Overview 4-48

4.2.1 DPL Ingest Computer Software Configuration Item Description 4-50

4.3 Client Subsystem Overview 4-61
4.3.1 Tools Description 4-61
4.4 Data Management Subsystem Overview 4-62
4.4.1 ECHO WSDL Order Component Software Description 4-65
4.4.2 Data Management Subsystem Hardware 4-71
4.5 Order Manager Subsystem Overview 4-73
4.5.1 Order Manager Subsystem Software Description 4-76
4.6 Communications Subsystem Overview 4-84
4.6.1 The Distributed Computing Configuration Item Software Description 4-86
4.6.2 The Distributed Computing Configuration Item Context 4-102
4.6.3 Distributed Computing Configuration Item Architecture 4-104
4.6.4 Distributed Computing Configuration Item Process Descriptions 4-105
4.6.5 Distributed Computing Configuration Item Process Interface Descriptions 4-105
4.6.6 Distributed Computing Configuration Item Data Stores 4-105
4.6.7 Communications Subsystem Hardware CI Description 4-105
4.7 Internetworking Subsystem (ISS) Overview 4-106
4.7.1 Internetworking Subsystem Description 4-107
4.7.2 Network COTS Hardware 4-109
4.8 EED General Process Failure Recovery Concepts 4-111
4.8.1 Client-Server Rebinding 4-111
4.8.2 Sybase Reconnecting 4-111
4.8.3 Request Identification 4-112
4.8.4 Request Responsibility 4-112
4.8.5 Queues 4-113
4.8.6 Request Responses 4-113
4.8.7 Duplicate Request Detection 4-117
4.8.8 Server Crash and Restart 4-118
4.8.9 Client Crash and Restart 4-120
4.9 Spatial Subscription Server (SSS) Subsystem Overview 4-122
4.9.1 Spatial Subscription Server Architecture 4-123
4.10 Data Pool Subsystem Overview 4-130
4.10.1 Data Pool Subsystem Context 4-133
4.10.2 Data Pool Hardware Context 4-135

4.10.3 Data Pool Insert CSCI Functional Overview 4-136
4.10.4 WebAccess CSCI Functional Overview 4-144
4.10.5 Data Stores 4-156
4.11 Bulk Metadata Generation Tool Subsystem Overview 4-157
4.11.1 BMGT Subsystem Context 4-157
4.11.2 BMGT/ECHO Interface 4-160
4.11.3 ECS Events and BMGT products 4-160
4.11.4 BMGT Architecture 4-163
4.11.5 Use of COTS in the BMGT Subsystem 4-166
4.11.6 BMGT Subsystem Software Description 4-167

List of Figures

Figure 3.2-1. Example Hierarchical Software Diagram 3-11
Figure 3.2-2. Release 8.0 Context Diagram 3-13
Figure 3.2-3. Subsystem Architecture Diagram 3-15
Figure 4.1-1. Data Server Subsystem Context Diagram 4-12
Figure 4.1-2. AIM CSCI Context Diagram (DPLIngest) 4-21
Figure 4.1-3. AIM Interfaces with DAAC Operators (ESDT Maintenance GUI and QA Update utility) 4-26
Figure 4.1-4. AIM Interfaces with DAAC Operators (XML Replacement Utility) 4-32
Figure 4.1-5. AIM Interfaces with DAAC Operators (Granule Deletion Utilities) 4-34
Figure 4.1-6. AIM Interfaces with DAAC Operators (Archive Check Utilities) 4-37
Figure 4.1-7. AIM Interfaces with BMGT 4-41
Figure 4.1-8. AIM Context Diagram (OMS and DPL) 4-43
Figure 4.2-1. DPL Ingest Subsystem Context Diagram 4-48
Figure 4.2-2. DPL Ingest CSCI Context Diagram 4-51
Figure 4.4-1. Data Management Subsystem Context Diagram 4-63
Figure 4.4-2. ECHO WSDL Order Component CSCI Context Diagram 4-66
Figure 4.4-3. ECHO WSDL Order Component CSCI Architecture Diagram 4-68
Figure 4.5-1. Order Manager Subsystem Context Diagram 4-74

Figure 4.5-2. Order Manager Server CSCI Context Diagram 4-77
Figure 4.5-3. Order Manager Server CSCI Architecture Diagram 4-79
Figure 4.6-1. Virtual Terminal Context Diagram 4-89
Figure 4.6-2. Virtual Terminal Architecture Diagram 4-90
Figure 4.6-3. Cryptographic Management Interface Context Diagram 4-92
Figure 4.6-4. Cryptographic Management Interface Architecture Diagram 4-93
Figure 4.6-5. Domains Hierarchy Diagram 4-95
Figure 4.6-6. DNS Domains of the EED Project Diagram 4-96
Figure 4.6-7. ECS Topology Domains Diagram 4-96
Figure 4.6-8. Domain Name Server Context Diagram 4-97
Figure 4.6-9. Distributed Computing Configuration Item (DCCI) CSCI Context Diagram 4-103
Figure 4.7-1. DAAC Networks: Generic Architecture Diagram 4-107
Figure 4.9-1. Spatial Subscription Server Context Diagram 4-122
Figure 4.9-2. Spatial Subscription Server Architecture Diagram 4-124
Figure 4.10-1. Data Pool Subsystem Context Diagram 4-133
Figure 4.10-2. Data Pool Hardware Context 4-136
Figure 4.10-3. Data Pool Insert CSCI Architecture Diagram – Registration 4-137
Figure 4.10-4. Data Pool Insert CSCI Architecture Diagram – Publication 4-137
Figure 4.10-5. WebAccess CSCI Architecture Diagram 4-145
Figure 4.11-1. BMGT Subsystem High Level Context Diagram 4-158
Figure 4.11-2. BMGT Architecture Diagram 4-163
Figure 4.11-3. Automatic Preprocessor Database Sequence 4-170
Figure 4.11-4. Manual Export Process Database Sequence 4-172

List of Tables

Table 4-1. Memory Management Table 4-6
Table 4.1-1. Data Server Subsystem Interface Events 4-14
Table 4.1-2. AIM Software Components 4-18

Table 4.1-3. AIM Interfaces with DPLIngest 4-22
Table 4.1-4. AIM Interfaces with DAAC Operators (ESDT GUI, QA Update) 4-27
Table 4.1-5. AIM Interfaces with DAAC Operators (XML Replacement Utility) 4-32
Table 4.1-6. AIM Interfaces with DAAC Operators (Granule Deletion) 4-35
Table 4.1-7. AIM Interfaces with DAAC Operators (Archive Check Utilities) 4-38
Table 4.1-8. AIM Interfaces with BMGT 4-41
Table 4.1-9. AIM Interfaces with OMS and DPL 4-43
Table 4.1-10. AIM CSCI Data Stores 4-46
Table 4.2-1. DPL Ingest Subsystem Interface Events 4-49
Table 4.2-2. DPL Ingest CSCI Interface Events 4-52
Table 4.2-3. DPL Ingest CSCI Processes 4-54
Table 4.2-4. DPL Ingest CSCI Process Interface Events 4-55
Table 4.2-5. DPL Ingest CSCI Data Stores 4-60
Table 4.3-1. Client Subsystem Interface Events 4-61
Table 4.3-2. SSI&T Tool Events 4-61
Table 4.4-1. Data Management Subsystem Interface Events 4-64
Table 4.4-2. ECHO WSDL Order Component CSCI Interface Events 4-67
Table 4.4-3. EWOC CSCI Processes 4-69
Table 4.4-4. EWOC CSCI Process Interface Events 4-69
Table 4.4-5. ECHO WSDL Order Component CSCI Data Stores 4-71
Table 4.5-1. Order Manager Subsystem Interface Events 4-74
Table 4.5-2. Order Manager Server CSCI Interface Events 4-77
Table 4.5-3. OMSRV CSCI Process 4-79
Table 4.5-4. Order Manager Server CSCI Process Interface Events 4-80
Table 4.5-5. CSCI Data Stores 4-83
Table 4.6-1. Communications Subsystem (CSS) Interface Events 4-84
Table 4.6-2. Virtual Terminal Interface Events 4-89
Table 4.6-3. Virtual Terminal Processes 4-90

Table 4.6-4. Virtual Terminal Process Interface Events 4-91
Table 4.6-5. Cryptographic Management Interface Events 4-92
Table 4.6-6. Cryptographic Management Interface Processes 4-93
Table 4.6-7. Cryptographic Management Interface Process Interface Events 4-94
Table 4.6-8. Cryptographic Management Interface Data Stores 4-94
Table 4.6-9. Domain Name Server Process 4-97
Table 4.6-10. Domain Name Server Process Interface Events 4-98
Table 4.6-11. Domain Name Server Data Stores 4-98
Table 4.6-12. Infrastructure Libraries 4-98
Table 4.6-13. Infrastructure Libraries Group Interfaces 4-100
Table 4.6-14. Distributed Computing Configuration Item (DCCI) CSCI Interface Events ... 4-103
Table 4.6-15. CSMS CSCI to CSS CSC Mappings 4-104
Table 4.7-1. Internetworking Subsystem Baseline Documentation List 4-109
Table 4.7-2. Networking Hardware for EED Networks 4-109
Table 4.8-1. Request Responses 4-115
Table 4.8-2. Fault Handling Policies 4-116
Table 4.8-3. Server Response versus Restart Temperature 4-119
Table 4.8-4. Server Response for Request Re-submission 4-120
Table 4.8-5. Server Responses to Client Failures 4-121
Table 4.9-1. Subscription Server Interface Events 4-123
Table 4.9-2. Spatial Subscription Server Processes 4-125
Table 4.9-3. Spatial Subscription Server Process Interface Events 4-126
Table 4.9-4. Spatial Subscription Server Data Stores 4-128
Table 4.10-1. Data Pool Subsystem Interface Events 4-133
Table 4.10-2. Use Cases for Data Pool Insert 4-138
Table 4.10-3. Data Pool Insert CSCI Process Description 4-139
Table 4.10-4. Data Pool ECS Insert CSCI Process Interface Events 4-140
Table 4.10-5. WebAccess CSCI Process Description 4-146

Table 4.10-6. WebAccess CSCI Process Interface Events 4-147
Table 4.10-7. Data Pool Data Stores 4-156
Table 4.11-1. BMGT Subsystem High Level Interface Events 4-159
Table 4.11-2. BMGT Metadata Product File Types 4-160
Table 4.11-3. BMGT/ECHO Interface Control File Types 4-160
Table 4.11-4. ECS Event to BMGT Product Mapping 4-161
Table 4.11-5. BMGT Processes 4-164
Table 4.11-6. Data Store 4-181

Abbreviations and Acronyms

This page intentionally left blank.

1. Introduction

1.1 Purpose and Scope

The purpose of the Segment/Design Specification for the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS) is to provide an overview of the hardware and software subsystems of the project. This document describes the high-level design of each ECS software subsystem implemented to satisfy the allocated and derived functional and performance requirements. This document also provides basic descriptions of the Commercial Off The Shelf (COTS) hardware and software used in the ECS. This document contains:

- Functional overviews of each Computer Software Configuration Item (CSCI)
- Context diagrams of each CSCI
- Interface event descriptions based on the context diagrams
- Process architecture diagrams
- Interface event description tables based on the process architecture diagrams
- CSCI data stores (databases as they relate to the process architecture diagrams)
- CSCI functions allocated to processes. For data servers, this includes descriptions of the functionality offered to clients via the server interfaces. For Graphical User Interface (GUI) applications, it describes the functionality provided to the GUI users
- Specific limitations of the capabilities provided
- Abbreviations and Acronyms

1.2 Document Organization

The remainder of this document is organized as follows:

- Section 2: Related Documentation
- Section 3: System Description
- Section 4: Subsystem Description
- Section 5: Limitations of Current Implementation
- Abbreviations and Acronyms

This page intentionally left blank.

2. Related Documentation

2.1 Parent Documents

The parent documents are the documents from which the scope and content of this Design Specification are derived. These documents are listed below.

423-46-01	EED F&PRS
423-46-03	EED Task 201 Statement of Work

2.2 Applicable Documents

Refer to the 900 Series documentation found on the EED Baseline Information System (EBIS) website: http://cmdm.hitc.com/baseline/.

2.2.1 Other Related Documents and Documentation

311-EED-001	Release 8.0 Ingest Subsystem (INS) Database Design and Schema Specifications for the EED Contract
311-EED-002	Release 8.0 Order Manager Subsystem (OMS) Database Design and Schema Specifications for the EED Contract
311-EED-003	Release 8.0 Spatial Subscription Server Subsystem (SSS) Database Design and Schema Specifications for the EED Contract
311-EED-004	Release 8.0 Data Pool Subsystem (DPL) Database Design and Schema Specifications for the EED Contract
311-EED-005	Release 8.0 Archive Inventory Management (AIM) Database Design and Schema Specifications for the EED Contract
611-EED-001	Release 8.0 Mission Operations Procedures for the EED Contract
625-EED-001	Release 8.0 Training Material Volume 1: Course Outlines
625-EED-002	Release 8.0 Training Material Volume 2: Problem Management
625-EED-003	Release 8.0 Training Material Volume 3: Ingest
625-EED-004	Release 8.0 Training Material Volume 4: Data Distribution
625-EED-005	Release 8.0 Training Material Volume 5: Archive Processing
423-42-06	Interface Control Definition for the EOS Data Gateway (EDG): Messages and Development Data Dictionary V0 and ASTER/ECS Message Passing Protocol Specification

RFC 793	Transmission Control Protocol
RFC 768	User Datagram Protocol
RFC 791	Internet Protocol
RFC 1597	Address Allocation for Private Internet
	WWW page is http://cmdm.east.hitc.com
423-41-57-6	ICD between ECS and SIPS, Volume 6 MODIS (MODAPS
423-41-57-7	ICD between ECS and SIPS, Volume 7 AMSR-E
423-41-57-9	ICD between ECS and SIPS, Volume 9 MTMGW
423-41-57-10	ICD between ECS and SIPS, Volume 10, TES Data Flows
423-41-57-11	ICD between ECS and SIPS, Volume 11, ICESat Data Flows
423-41-58	ICD between the ECS and LP DAAC
423-ICD-EDOS/EGS	ICD between EDOS and EGS

2.3 ECS Tool Descriptions

2.3.1 Rational Rose

The Rational Rose tool provides support for object-oriented analysis and design. In particular, the Rose tool provides support for controlled-iterative or component-based development. The Rose tool is used on the EED Project to document the object-oriented elements of the design using class diagrams, use-case diagrams, interaction diagrams, component diagrams, and object diagrams. The Unified Modeling Language (UML) is the methodology used on the EED Project for all design activities (although the Rose tool also supports the Booch ’93 Methodology or the Object Modeling Technique (OMT) as well).

The Rose tool can also be used to reverse engineer code developed that lacks supporting documentation to get as-built object diagrams.

Before using the Rational Rose tool, see “Rational Rose 98, Using Rose” for important tool usage and reference information. In addition, the following references can be obtained and used:

(1) “Unified Method for Object-Oriented Development,” by Grady Booch and Jim Rumbaugh (version 1.1, Rational Software Corporation) for an introduction to the respective method’s notation, semantics, and process for object-oriented analysis and design.

(2) the second edition of “Object-Oriented Analysis and Design with Applications” by Grady Booch, (Benjamin/Cummings, 1994)

(3) “Object-Oriented Modeling and Design” by James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy and William Lorensen, (Prentice-Hall, 1991)

(4) "UML Distilled: Applying the Standard Object Modeling Language" by Martin Fowler with Kendall Scott, Foreword by Grady Booch, Ivar Jacobson, and James Rumbaugh (Addison Wesley Longman, Inc., 1997)

2.3.2 ClearCase Baseline Manager Configuration Management Tool

ClearCase Baseline Manager (CBLM) consists of the ECS baseline data and a Graphical User Interface (GUI) used to control the ECS baseline.

The data comes from two sources:

1) Existing Release Notes (914-TDA-xxx) Machines Impacted data
2) Newly approved CCRs

Control Item Identifiers (CIDs) consist of an 8-digit integer with a "b" prefix (e.g., b00083456). Each COTS S/W product has its own CID. Because CIDs are mapped to ECS hosts, it was decided to represent information within ClearCase as elements. For the ClearCase CID elements, the comma separated variable (CSV) format was chosen, as this format is easily ported into and from other COTS S/W products, specifically Microsoft Access and Excel.

The ClearCase configuration specification chosen was the simplest, or the default configuration specification. A view, CM_MASTER, was created with the default configuration specification to manage the data records. The CID records (checked in ClearCase elements) are located in the /ecs/cm VOB at /ecs/cm/CIDs. This directory currently contains the 256 records that correlate to XRP-II's COTS S/W CIDs.

Another important data construct within CBLM is the notion of the Machines Impacted file, and a CCR identified construct, which maps CIDs to hosts. Each Configuration Change Request (CCR) affecting the baseline contains information about 1 or more CIDs. Also, the CCR contains information regarding the hosts receiving the COTS S/W (CID). So the CCR has a construct that in its simplest form is one "CID_MAP" file, and one Machines Impacted (MI) file. The "CID_MAP" file is a simple lookup table. In this case, there is only one entry. The entry contains first a valid CID, followed by one or more blanks, then the name of the "MI" file. In this case, the MI filename is "MI." The MI file, contains a list of valid ECS hosts having the COTS S/W identified within the CID. So a CCR (07-1234) to place a COTS S/W (e.g., Acrobat Reader), onto host e4eil01, would have an MI file containing one host, e4eil01, and one CID_MAP. If the Acrobat Reader software is CID b00081234, the CID_MAP file would contain:

"b00081234 MI"

And the MI file contains:

"e4eil01"

The CCR would be found at:

/ecs/cm/CM/2007CCRs/1234/, a directory

Under this directory is found the two files, “CID_MAP” and “MI.” Note that there is always only one CID_MAP file for each CCR, but that the CID_MAP may contain more than one entry. The simplest example of this is when a COTS S/W product needs to be mapped to SGI, Sun, and Linux hosts. There would be three MI files, “MI_SGIs”, “MI_Suns”, and “MI_Linux” for example. The CID_MAP would contain three entries, one mapping the SGI hosts to the SGI CID, one mapping the Sun hosts to the Sun CID, and one mapping the Linux hosts to the Linux CID.

As approved CCRs are required to change the CBLM data state, the effective date is then defined as the CCR approval date. This is the date the change becomes valid. The next construct, named the “Sequencer”, is the table providing the history of change. The last approved CCR is at the end of the table. As new CCRs get approved, they simply get concatenated to the end of the list in time order. The Sequencer is an executable script.

The last construct is the “dartboard.” Conceptually, the “dartboard” is a directory within ClearCase, at /ecs/cm/BLM/dartboard/. All ECS hosts are listed as files in the /dartboard/ directory. In conclusion, then data constructs are:

CIDs

CCR directories

CID_MAPs and MIs under the CCR directories

Sequencer

Dartboard

The way these pieces all work together is now briefly discussed.

When a CCR is approved that affects the baseline, a CCR is checked out. The /ecs/cm/CIDs/ ClearCase directory is checked out. The new CID is created and populated with the information present on the CCR form. The new CID number then has a ClearCase element created, and the first version becomes this new CID. The /ecs/cm/CIDs/ directory is then checked back in. Next, the MI file must be prepared. Within the CCR directory, two new files are “made” (cleartool mkelem –eltype text_file –nc CID_MAP MI). The hosts, which are to get the COTS S/W, are entered into the MI file, then the file is checked in as the first version. Next, the CID_MAP file is created, mapping the new CID number to the MI file. The CID_MAP file is checked in, then the CCR file is checked back in. This work gets the CCR information locked into ClearCase.

Next, the Sequencer file is edited to show the new CCR number at the end. This action allows the CCR’s MI and CID_MAP files to overlay onto the ClearCase baseline. This is accomplished by echoing the contents of the CID (in file /ecs/cm/CIDs/b00083123) onto each of the hosts specified with the /ecs/cm/CM/2007CCRs/1234/MI file. This data is written to the hosts files with the dartboard, located at /ecs/cm/BLM/dartboard.

Once the data has been applied to the dartboard, subsequent scripts then produce the output reports. In conjunction with the current hosts list, the scripts obtain all of the valid hosts of the site, and basically reformat the data within the dartboard files into reports. Information is added

to the reports, including the CCR number, related Release Notes documentation, and the CCR approval date.

The reports are written to the /ecs/cm/BLM/reports directory. Subsequent “expect” scripts then scp those reports to specific locations on the EBIS server, c4cbl02, and then the reports are replicated to each of the 5 remote sites (SMC and 4 DAACs).

The languages used in this tool are “sh”, “csh”, “expect”, and C. Also, “.grp” files are used to represent the ClearCase GUIs. These files are text files that are dynamically generated at the time that the GUI is launched. Code has been reused from two sources, the DeliveryTool, which is used to prepare and send data to the sites, and the replication scripts, which are used to replicate data from the Landover EBIS server c4cbl03, to the protected (SMC, DAACs) servers.

This page intentionally left blank.

3. System Description

3.1 Mission and Release 8.0 Objectives

The Mission of the National Aeronautics and Space Administration's (NASA) Earth Science Enterprise (ESE) is to develop a scientific understanding of the total Earth System and its response to natural or human-induced changes to the global environment to enable improved prediction capability for climate, weather and natural hazards. The vantage point of space provides information about Earth's land, atmosphere, ice, oceans and biota that is obtained in no other way. Programs of the enterprise study the interactions among these components to advance the new discipline of Earth System Science, with a near-term emphasis on global climate change. The research results contribute to the development of sound environmental policy and economic investment decisions.

The Earth Observing System Data and Information System (EOSDIS) Core System (ECS) has been designated as the ground system to collect, archive, produce higher-level data products and distribute data for the Earth System Science mission.

3.1.1 Release 8.0 Capabilities

The ECS capabilities have been developed in increments called formal releases. Release 8.0, which is managed by Configuration Management, is a formal release. It is a collection of new and updated capabilities provided to the users of the system and is described here to show the progress of system enhancements. The ECS collects and stores, processes, archives and distributes scientific data from six different platforms (satellites). In the following sub-sections, the platforms and instruments from which scientific data is collected are identified, the type of data ingested and archived is presented, search and order capabilities for scientific data, how data is distributed and processed, system architecture and operation, system security and Distributed Active Archive Center (DAAC) and external system support are described. Other capabilities provided by Release 8.0 include processing the data obtained, distributing raw or processed data as requested, quality assurance of processed data, supporting communication networks, and systems monitoring via interfaces with the ECS operations staff.

Release 8.0 includes the following unique capabilities and modifications:

- FTP Pooling Service

This capability allows ECS components to reuse FTP connections rather than create a new connection for every file transfer operation. The capability was initially delivered as a patch to the 7.23 release and DPL Ingest was modified to make use of the service. The benefit is reduced overhead in file transfers and reduced load on remote FTP servers. For Release 8.0 OMS and BMGT were modified to take advantage of the new service. An additional benefit is that instead of maintaining three FTP libraries for the three subsystems, there is only one library to maintain.

- Operation IceBridge support

In addition to the metadata generation support provided on IceBridge Task 5, ECS has been modified for Release 8.0 to handle several new IceBridge specific requirements. IceBridge data are collected by aircraft-mounted instruments and have different characteristics than the satellite data that have been archived in ECS. ECS components have been modified to support spatial coverages with exclusion zones, provide a capability to store and search for data based on data acquisition campaign, and support multiple browse formats including JPEG and PDF. As a result of these changes to ECS, support for new IceBridge datasets can be provided more rapidly. It also positions ECS to be a good solution for the archival and distribution of flight data in future missions.

- System Monitoring

Enhancements were delivered with Release 8.0 to automate the creation of alerts that applied to all of the DAACs. Further enhancements were made to track configuration file changes, monitor SQS connectivity, monitor the MySQL database, enhance the server control and business processing monitoring pages, and automatically verify all configuration mode by mode. These changes allow the DAAC operators to more effectively monitor ECS while saving a considerable amount of configuration time.

- BMGT operator GUI improvements

Enhancements included providing an auto-refresh feature, an alerting mechanism for incorrect collection configuration, and a way to configure the maximum number of packages to display. These improvements allow DAAC operators to more effectively monitor the BMGT system.

- EMS enhancements

Changes were made to the ECS system to provide more detailed information on QA, PH, and metadata file accesses.

- MSS to OMS database consolidation

Prior to release 8.0, order information was tracked in both the MSS and OMS databases. By migrating all of the relevant order information from MSS to OMS, the Order Manager system has been simplified significantly. In addition there is one less database to be maintained by DAAC database administrators and some database space that can be recovered.

- AIM/DPL inventory reconciliation improvements

In preparation for the next SDPS release (8.1) in which redundancy in the AIM and Data Pool databases will be reconciled, a new utility has been created and several existing utilities have been modified to identify and repair any discrepancies in the AIM and Data Pool inventory. The new AIM/DPL Checksum Comparer keeps track of any checksum discrepancies between the two databases and automatically determines the correct way to reconcile the discrepancy. Ingest has been modified to ensure all files have a checksum created if the provider does not provide one. The archive checksum verification utility, and Data Pool checksum server have been modified to allow an operator to change the checksum type. The inventory validation utility has been modified to track discrepancies in file information between Data Pool and AIM. All of these enhancements will allow the DAACs to ensure the integrity of their archives as well as meet the prerequisites to the 8.1 transition well ahead of time.

Other enhancements include a new option to repair only XML files rather than restoring all of the files for a granule, and the ability to disposition Order Manager requests in bulk. These changes resulted in simplifying operations at the DAACs. In addition EWOC has been modified to use the ECHO 10 API which will allow ECHO to retire the ECHO 9 API.

3.1.1.1 ECS Support of Instruments by Platform

- The Meteor 3 platform supported the Stratospheric Aerosols and Gas Experiment III (SAGE III) instrument
- The ACRIMSAT platform supports the ACRIM III experiment
- The Terra (AM-1) platform supports the Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER), Multi-Angle Imaging SpectroRadiometer (MISR), Moderate Resolution Imaging SpectroRadiometer (MODIS) and Measurements of Pollution in the Troposphere (MOPITT) instruments
- The Aqua (PM-1) platform supports the Moderate Resolution Imaging SpectroRadiometer (MODIS) and Advanced Microwave Scanning Radiometer (AMSR-E) instruments
- The Ice, Cloud and Land Elevation satellite (ICESat) platform supported the Geoscience Laser Altimeter System (GLAS) instrument
- The Aura platform supports the Tropospheric Emission Spectrometer (TES) instrument
- Various aircraft support the set of Operation IceBridge instruments including the Airborne Topographic Mapper (ATM), the Digital Mapping System (DMS), and the Land, Vegetation, and Ice Sensor (LVIS)

3.1.1.2 Ingest and Archive Capabilities

The following data is ingested and archived in the ECS from the various instruments described in Section 3.1.1.1:

- Ingest of science and engineering data from the EOS Data and Operations System (EDOS)
- Ingest of Product Generation Executable (PGE) software from Science Computing Facilities (SCFs) either electronically or via media tape
- Ingest of ASTER Level 1A/1B data
- Ingest of FDS (formerly FDD) orbit data
- Ingest of SAGE III MOC Level 0 data
- Ingest of SAGE III SCF higher-level products via the SIPS interface
- Ingest of Data Assimilation System (DAS) HDF-EOS data via standard polling with DR
- Ingest of MODIS higher-level products via the SIPS Interface
- Ingest of MOPITT SCF Level 0 data via the SIPS interface
- Ingest of SDPS resident data across a mode in the same DAAC or across DAACs
- Ingest of ACRIM Level 0 and higher-level data from the ACRIM SCF via the SIPS interface
- Ingest of higher-level AMSR data products from the AMSR SCF
- Ingest of the Operation IceBridge data at the NSIDC DAAC
- Archive of ICESat GLAS Level 1, Level 2 and Level 3 and ancillary data at the NSIDC DAAC
- Archive of TES Level 1, Level 2 and Level 3 data
- Archive of products previously processed and archived

3.1.1.3 Search and Order Capabilities

The ECS provides the following capabilities for search and ordering of data from the archive:

- Handling of orders from ECHO via EWOC
- CLS provides an user interface to query Order Status
- Handling of variations on search areas and product-specific spatial representations via WIST and Data Pool Webaccess
- The SSS provides an operator the interface to place standing orders (subscriptions) based on an ECS event and manage subscription status
- The Data Pool provides an operator the interface to manage insert processes, queues, collection groups and collection themes for ECS and non-ECS collections

3.1.1.4 Data Distribution Capabilities

The ECS provides the following Data Distribution capabilities for users:

- Support distributing science data products via FTP Push or Pull, and SCP Push.

3.1.1.5 Data Processing Capabilities

The ECS provides the following capabilities for user/operator data processing options:

- Support the archive of products previously produced and archived
- Provide capability for operator deletion of granules
- Support Quality Assurance (QA) processing of science data products
- Automated support for on-demand requests for ASTER processing
- Provide capability to associate the ASTER browse granule for the L1A product with ASTER L1B products

3.1.1.6 System Operation and Architecture

The ECS provides the following capabilities to support the system operations and processing architecture used to provide data and services for users:

- Provide capability for operator deletion of granules, their associated metadata and browse files
- Provide the associated communications network interfaces with the SCFs
- Support managing the startup and shutdown of system network components, database and archive administration, system data and file back-up and restores, system performance tuning and resource usage monitoring, and other routine operator duties
- Operations support to update certain ESDT attributes without requiring the deletion of the data collection
- Provide the capability for editing of ECS core attribute values
- Support the consolidation of trouble tickets using TestTrack Pro
- Provide fault recovery for mode management
- Provide the capability for startup and shutdown of an entire mode
- Provide the capability for the deletion of science data from the archive
- Provide the capability for the installation of ESDTs to insert and acquire archived data
- Provide for the storage of event information into the AIM database
- Provide the capability for the monitoring of the usage of memory
- Provide COTS packages to allow operations to generate customized reports from ECS databases
- Provide a single configuration registry database to replace the numerous ECS application configuration files
- Provide for the insertion of ECS and non-ECS granules into the Data Pool

- Export ECS metadata and browse data to ECHO via BMGT
- Provide delete, publish, restore, validation, consistency and integrity checking tools

3.1.1.7 Security

The ECS provides the following capabilities for system security:

- Encryption of passwords in ECS databases
- SDP Toolkit support for thread safe concurrent processing by the science software
- Secure Transfer of data files from Data Providers upon request
- System data and file backups and restores

3.1.1.8 DAAC/External System Support

ECS Release 8.0 will be distributed to three site locations including:

1. The DAAC at the Langley Research Center (LaRC),
2. The Land Processing DAAC (LP DAAC), and
3. The DAAC at the National Snow and Ice Center (NSIDC)

The ECS Release 8.0 communications network includes the National Aeronautics and Space Administration (NASA) and the NASA Integrated Services Network (NISN). These portions of the network are physically located at the DAAC sites. The communications network connects ECS to data providers at the EDOS, NOAA Affiliated Data Center (ADC), and the EOSDIS Version 0 system.

The data users for Release 8.0 are the science user community connected to the ECHO, Data Pool Webaccess, the three DAACs, the SCFs, and the MODAPS.

1. LaRC Support:
    - ECS Release 8.0 provides a communications network and data/information management support for MISR instrument data including the receipt of MISR level 0 data and the LaRC archive and distribution of levels 1, 2 and 3 data and data products
    - ECS Release 8.0 provides a communications network and data/information management support for MOPITT instrument data including the receipt of MOPITT level 0 data, the LaRC archive, and distribution of levels 1, 2 and 3 data
    - ECS Release 8.0 provides a communications network and data/information management support for TES instrument data including the receipt of TES level 0 data and the LaRC archive, and distribution of levels 1, 2 and 3 data
    - LaRC DAAC capabilities include:

- Ingest of MISR, TES, and MOPITT Level 0 and related ancillary data
- Archival, and distribution of the higher-level products for MISR
- Receipt of higher-level MOPITT products from the MOPITT SCF, via the SIPS interface, for archival and distribution
- Receipt of SAGE III products from the SCF, via the SIPS interface, for archival and distribution
- Receipt of ACRIM products (Level 0 and Level 2 data) from the SCF, via the SIPS interface, for archival and distribution
- Receipt of TES Levels 1-3 data including algorithm and associated software packages, metadata, production histories, ancillary data and Quality Assessment (QA) data for archival and distribution

2. LP DAAC Support:

- ECS Release 8.0 provides a communications network and data/information management support for ASTER instrument data including the receipt of ASTER level 1A data electronically at LP DAAC from Japan, and distribution of higher level ASTER products by LP DAAC
- LP DAAC capabilities include:
  - Ingest of ASTER Level 1A/1B, with ancillary data needed for production
  - Archival and distribution of ASTER products
  - Receipt of higher level MODIS land products from MODAPS, via the SIPS interface, for archival and distribution

3. NSIDC Support:

- AMSR-E instrument data including the receipt of level 0 data from EDOS at ECS, and the NSIDC archive and distribution of levels 1, 2 and 3 data. The Level 1A data is received from the NSIDC V0 DAAC while the level 2 and 3 data is received from the AMSR-E SCF via the SIPS interface
- AMSR-ADEOS II Level 1A data is received from the NSIDC DAAC and archived and distributed using ECS.
- ECS Release 8.0 supports the ingest of ICESat GLAS level 1, level 2, level 3 and ancillary input data for archive and distribution at the NSIDC DAAC using the standard SIPS interface.
- NSIDC DAAC capabilities include:
  - Receipt of higher-level MODIS snow and ice products from MODAPS, via the SIPS interface, for archival and distribution
  - Ingest of AMSR-E Level 0 data and related ancillary data

- Receipt of the AMSR-E and AMSR-ADEOS II higher-level products via the SIPS interface, for archival and distribution
- Archival and distribution of GLAS Level 0 data and related ancillary data
- Receipt of the GLAS higher level products from the SCF, via the SIPS interface, for archival and distribution
- Receipt of the Operation IceBridge data including ATM, DMS, and LVIS higher-level products, among others, via the SIPS interface, for archival and distribution. The product metadata are created by the ECS MetGen program.

4. SCF Support:
   - The MOPITT higher-level products are generated at the SCF and provided to the ECS via the SIPS interface
   - ECS Release 8.0 supports receiving SAGE III Level 0 data and higher level products from the SCF via the SIPS interface
   - ECS Release 8.0 supports receiving ACRIM L0 data and higher level products from the SCF via the SIPS interface
5. MODAPS Support
   - ECS Release 8.0 provides a communications network and data/information management support for MODIS instrument data including: archive and distribution of higher level data from the MODIS Data Processing System (MODAPS)

3.2 Release 8.0 Architecture Overview

The ECS Release 8.0 architecture comprises the logical items listed here. Commercial Off The Shelf (COTS) software and hardware are used, to the extent possible, to implement the ECS functionality of these logical items.

- System
- Segments
- Subsystems
- Computer software configuration items (CSCIs)
- Computer software components (CSCs)
- Processes

ECS Release 8.0 was built of the following two segments.

- CSMS – Communications and Systems Management Segment
- SDPS – Science Data Processing Segment

Each segment was in turn built of the following subsystems:

- CSMS: CSS – Communications Subsystem
  ISS – Internetworking Subsystem
- SDPS: BMGT – Bulk Metadata Generation Tool Subsystem
  CLS – Client Subsystem
  DMS – Data Management Subsystem
  DPL – Data Pool Subsystem
  DPL INGEST – Data Pool Ingest Subsystem
  DSS – Data Server Subsystem
  OMS – Order Management Subsystem
  SSS – Spatial Subscription Server Subsystem

Hierarchical Definitions

System: A stand-alone composite of hardware, facilities, material, software, services, and personnel required for operation based upon a defined set of system level requirements and designed as a related set of capabilities and procedures.

Segment: A logical and functional subset of related capabilities, implemented with COTS hardware and COTS and custom developed software to satisfy a defined subset of the system level requirements.

Subsystem: A logical subset of Segment related capabilities, implemented with COTS hardware and COTS and custom developed software to satisfy a defined subset of segment level requirements.

CSCI: A logical subset of Subsystem related capabilities, implemented with COTS and custom developed software to satisfy a defined subset of the subsystem level software requirements.

CSC: A logical subset of CSCI related capabilities, implemented with COTS and custom developed software to satisfy a defined subset of the CSCI level software requirements.

Process: A logical and functional set of software, written in a specific order and in a defined manageable size to manipulate data as part of a product-generating algorithm. A process is a separately compiled executable (i.e., binary image). A process can use infrastructure library calls, system service calls, COTS service

calls, and application programming interfaces to manipulate data to generate products.

Figure 3.2-1 is a hierarchical software diagram. The hierarchical software diagram depicts an example of the decomposition levels used in the ECS design and described in this document. The diagram is also a graphical representation of the terms just described.

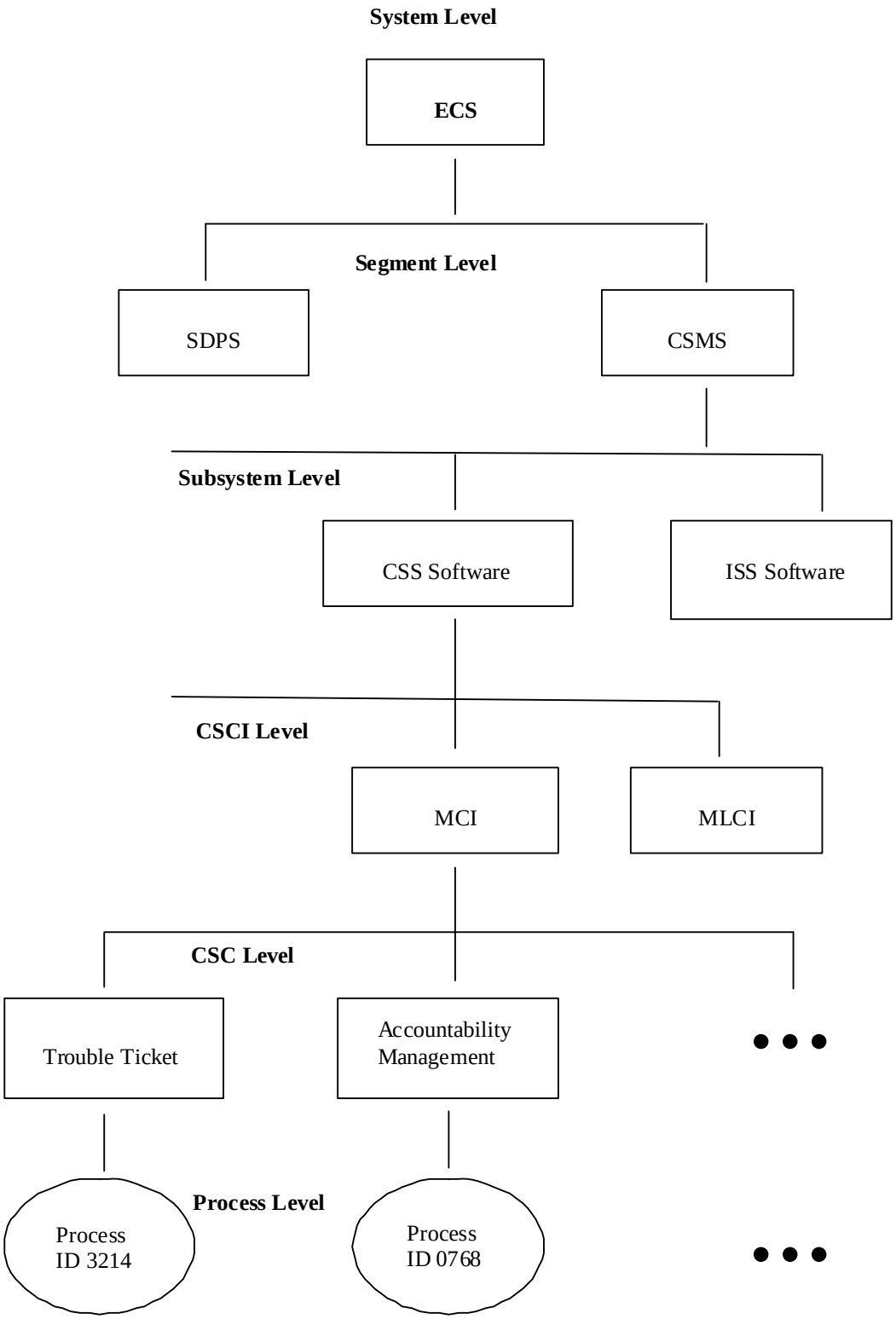

Figure 3.2-1. Example Hierarchical Software Diagram

3.2.1 Release 8.0 Context Description

ECS Release 8.0 provides the capability to collect and process satellite science data as depicted in Figure 3.2-2.

The Science Data Processing and Communications and Systems Management are the two segments of Release 8.0 described in this document. The Science Data Processing Segment (SDPS) provides science data ingest, search and access functions, data archive, and system management capabilities. The SDPS receives Level 0 science data from EDOS. The SDPS exchanges data with affiliated data centers to obtain science and other data (i.e., engineering and ancillary) required for data production. Science algorithms, provided by the Science Computing Facilities (SCFs), are archived for distribution. The Communications and Systems Management Segment (CSMS) provides the communications infrastructure for the ECS and systems management for all of the ECS hardware and software components. The CSMS provides the interconnection between users and service providers within the ECS, transfer of information between subsystems, CSCIs, CSCs, and processes of the ECS.

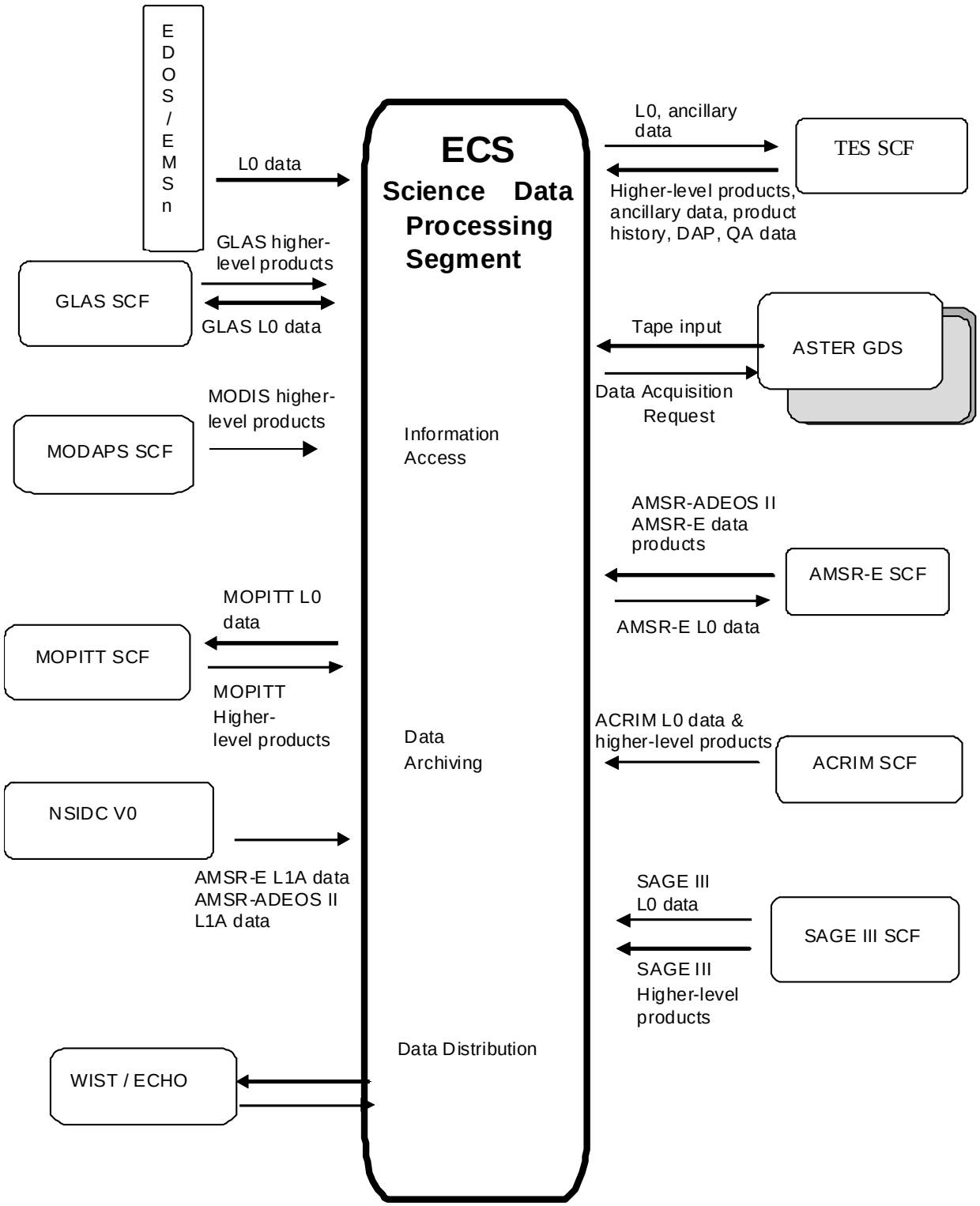

Figure 3.2-2. Release 8.0 Context Diagram

The remaining sections of this document provide an overview of Release 8.0 design and as such do not deal specifically with the configuration of components at each EOSDIS site. For more information on the site unique configurations, refer to the 920-series of General documents. Each of the segments consists of subsystems as specified in Section 3.2.

3.2.2 Release 8.0 Architecture

3.2.2.1 Subsystem Architecture

The ECS SDPS subsystems are depicted in Figure 3.2-3. A subsystem consists of the Commercial Off The Shelf (COTS) and/or ECS developed software and the COTS hardware needed for its execution. The SDPS subsystems can be grouped into a 'Push' or 'Pull' category of functionality with the exception of DSS. As shown in the subsystem architecture diagram, the information search and data retrieval makes up the 'Pull' side of the ECS architecture/design and consists of the CLS, DMS, OMS, SSS, DPL and also uses the DSS functionality described on the 'Push' side of the ECS architecture. Data capture (ingest of data), storage management, planning and data processing of satellite or previously archived data from other sites make up the 'Push' side of the ECS architecture/design and consists of the DSS, DPL, DPL INGEST, and OMS. This document describes the software and hardware components of each subsystem. However, since the hardware configurations differ between the sites, the hardware descriptions in this document are at a generic level. Specific hardware and network configurations for each site are documented in the 920 and 921 series technical documents.

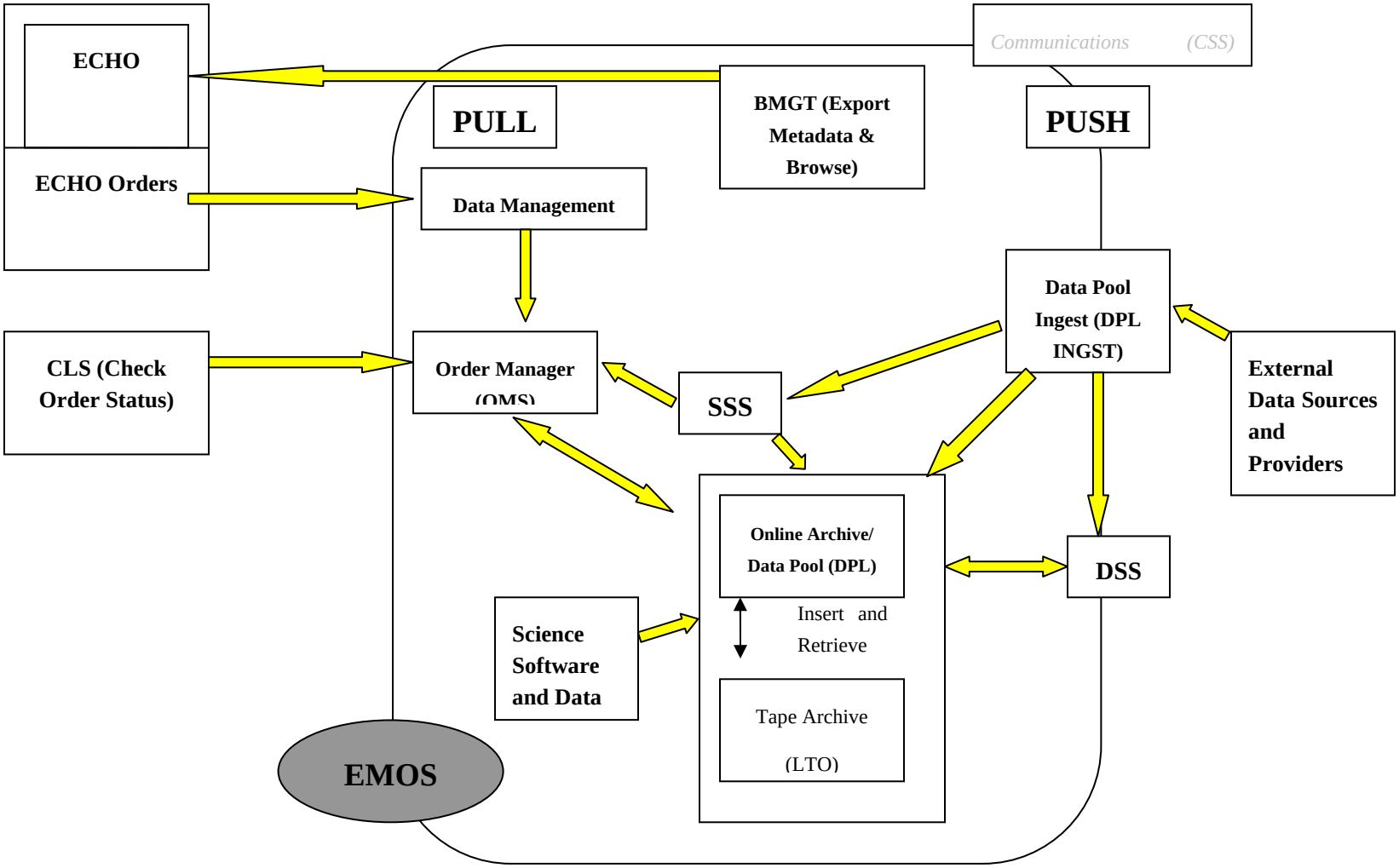

Figure 3.2-3. Subsystem Architecture Diagram

The ECS SDPS architecture/design consists of:

- BMGT exporting inventory status information to ECHO.
- CLS providing a user interface to check order status.
- DMS providing support for data retrieval across all ECS sites.
- DPL supporting the search, order, and distribution of selected granules with associated metadata and browse granules (if available).
- DPL INGEST service will handle the SIPS ingest interface, S4P, cross-DAAC ingest, EDOS ingest, ASTER Ingest and Polling without Delivery Record specifically for EMOS.
- DSS with the functions needed to manage the inventory of archived data.
- OMS managing all orders received from the DMS EWOC (i.e., from WIST and ECHO), the Machine-to-Machine Gateway, and the Spatial Subscription Server.
- SSS supporting the creation, viewing and updating of subscriptions and the creation, viewing and deletion of bundling orders (specification of distribution packages and criteria for package completion).

CSMS – The following subsystems are the CSMS subsystems, which interact with and support the SDPS to complete the ECS architecture.

1. The CSS with:
    - Control Center System (CCS) Middleware provides a common Name Server, which packages the common portions of the communication mechanisms into global objects to be used by all subsystems. The Name Server provides a set of standard CCS Proxy/Server classes, which encapsulates all of the common code for middleware communications (e.g., portals, couplers, RWCollectables, etc.)
    - Libraries with common software mechanisms for application error handling, aspects of recovering client/server communications; Universal References to distributed objects and interfaces to e-mail, file transfer and network file copy capabilities
2. The ISS with:
    - Networking hardware devices (e.g., routers, switches, hubs, cabling, etc.) and their respective embedded software. For more information on site unique configurations, refer to the 920-series of General documents

4. Subsystem Description

Design Description Organization and Approach

This section presents a subsystem-by-subsystem overview description of the “as-built” ECS. The current high-level design information is provided for the Hardware Configuration Items (HWCI), Computer Software Configuration Items (CSCI), and Computer Software Components (CSC) for each subsystem and is being delivered to the DAACs in drop increments.

The CSMS subsystem descriptions include:

- Subsystem functional overviews with a subsystem context diagram and a table of interface event descriptions
- CSCI descriptions with a context diagram and a table with interface event descriptions
- Architecture Diagrams, Process Descriptions, and Process Interface Event Tables. The Architecture Diagrams show the processes of the CSCI/CSC and how these processes connect with other CSCIs and CSCs of the same subsystem and the interfaces with other subsystems and external entities such as Operations, External Data Providers and Users.
- Data Store descriptions for each CSCI in the CSMS subsystem. The Data Stores are identified with the software name and shown in the architecture diagrams either as single data stores or as a group of data stores with a generic name such as “Data Stores” or “database”
- Hardware descriptions of the subsystem hardware items and the fail-over strategy

The convention used for Context and Architecture diagrams includes using circular shapes to show the subject subsystems, CSCIs, CSCs, or processes (with name in bold), elliptical shapes to show associated CSCIs, CSCs, or processes within a given subsystem and squares or rectangles to show external subsystems, CSCIs, CSCs, and processes. Data stores are shown using the data store or database name with a pair of horizontal lines one above and one below the name, or as a cylinder with the name below. An interface event is data, a message (which includes a notification or status); a command, request or status code passed between subsystems, CSCIs, CSCs, or processes. The convention used to identify events is a straight line between two objects labeled with a phrase beginning with an action-oriented word to best describe the event. The arrow on the event line indicates an origination point and to where the event is directed. A direct response to an event is not always shown on the diagram because sometimes there is no response (e.g., for an insert or delete request) and other times the response comes from another part of the ECS. Interface events are identified in the interface event or process interface tables starting with the interface event at the top or middle of the diagram and going clock-wise around the diagram. The external interface subsystem is identified in the interface event description and is in bold to assist with the location of the interface events on the diagram. If there are two items in bold, there are two different interfaces (Subsystems, CSCIs, or CSCs) requesting the same interface event. These conventions are consistent with other ECS documentation. The convention for

naming the ECS processes is Ec <subsystem abbreviation> meaningful name. The *Ec* identifies the process as an ECS developed process versus a Commercial Off The Shelf (COTS) product. The *subsystem abbreviations* are listed subsystem-by-subsystem.

- Bm for BMGT
- Cl for CLS
- Cs for CSS
- Dl for DPL (also Data Pool Ingest)
- Dm for DMS
- Ds for DSS
- Ms for MSS
- Nb for SSS
- Om for OMS

The *meaningful name* identifies the process and its functionality within the subsystem, CSCI, or CSC. An example is EcDsAmIiu, which identifies an ECS-developed DSS process called the AIM Inventory Insert Utility. Some names within an architecture diagram do not follow this convention because the names are COTS product names. All COTS product names are kept for simplicity and to adhere to licensing and trademark agreements. The remaining names that do not follow the naming convention are imbedded throughout the system and would require time to replace and cause operational disruptions. These names will be cleaned up during the final maintenance stages of the contract if directed by the customer.

Object-oriented modeling and design

Object-oriented modeling and design is a new way of thinking about problems using models organized around real-world concepts. The fundamental construct is the object, which combines both data structure and behavior in a single entity. Objected-oriented models are useful for understanding problems, communicating with application experts, modeling enterprises, preparing documentation and designing programs and databases.[1]

Superficially the term "object-oriented" means that we organize software as a collection of discrete objects that incorporate both data structure and behavior. This is in contrast to conventional programming in which data structure and behavior are only loosely connected. There is some dispute about exactly what characteristics are required by an object-oriented approach, but generally include four aspects: identity, classification, polymorphism and inheritance.[11] *Identity* means that data is quantized into discrete, distinguishable entities called *objects*. A paragraph in my document, a window on my workstation and a white queen in a chess game are examples of objects. Objects can be concrete, such as a file, or conceptual, such as a *scheduling policy* in a multi-processing operating system. Each object has its own inherent

[1] Object-oriented Modeling and design, James Rumbaugh et all, copyright 1991 by Prentice-Hall, Inc. ISBN 0-13-629841-9

identity. In other words, two objects are distinct even if all their attribute values (such as name and size) are identical.[1]

In the real world an object simply exists, but within a programming language each object has a unique *handle* by which it can be uniquely referenced. The handle may be implemented in various ways, such as an address, array index or unique value of an attribute. Object references are uniform and independent of the contents of the objects, permitting mixed collections of objects to be created, such as a file system directory that contains both files and sub-directories.[1]

Classification means that objects with the same data structure (attributes) and behavior (operations) are grouped into a *class*. Paragraph, Window, and ChessPiece are examples of classes. A *class* is an abstraction that describes properties important to an application and ignores the rest. Any choice of classes is arbitrary and depends on the application.[1]

Each class describes a possibly infinite set of individual objects. Each object is said to be an instance of its class. Each instance of the class has its own value for each attribute but shares the attribute names and operations with other instances of the class. An object contains an implicit reference to its own class: it "knows what kind of a thing it is."[1]

Polymorphism means that the same operation may behave differently on different classes. The *move* operation, for example, may behave differently on the *Window* and *ChessPiece* classes. An *operation* is an action or transformation that an object performs or is subject to. *Right justify, display* and *move* are examples of operations. A specific implementation of an operation by a certain class is called a *method*. Because an object-oriented operator is polymorphic, it may have more than one method implementing it.[1]

In the real world, an operation is simply an abstraction of analogous behavior across different kinds of objects. Each object "knows how" to perform its own operations. In an object-oriented programming language, however, the language automatically selects the correct method to implement an operation based on the name of the operation and the class of the object being operated on. The user of an operation need not be aware of how many methods exist to implement a given polymorphic operation. New classes can be added without changing existing code, provided methods are provided for each applicable operation on the new classes.[1]

Inheritance is the sharing of attributes and operations among classes based on a hierarchical relationship. A class can be defined broadly and then refined into successively finer *subclasses.* Each sub-class incorporates, or *inherits* all the properties of its *super-class* and adds its own unique properties. The properties of the superclass need not be repeated. For example, *ScrollingWindow* and *FixedWindow* are subclasses of *Window*. Both subclasses inherit the properties of *Window*, such as a visible region on the screen.[1]

The ECS is a large, complex data storage and retrieval system used to store and retrieve large amounts of science and science-related data. The system was designed using an object oriented design approach. With so many objects and the sizes of some of them, it is necessary to have

---

[1] Object-oriented Modeling and design, James Rumbaugh et all, copyright 1991 by Prentice-Hall, Inc. ISBN 0-13-629841-9

some insight into the amount of memory being utilized within the ECS. The information about to be presented is a brief look at the memory management of the "key" (top ten utilized) objects within the ECS subsystems.

In this object oriented system design, objects are created and used via classes throughout the system to help perform the functions and meet the needs of the system. The objects for the ECS are very numerous, sometimes very large and cannot be provided in their entirety at this time. However, presented in the table below are the "key" objects for this system and how they are created, passed and deleted within the ECS.

Introduction to memory management approaches and memory usage table

Good memory management in some applications is important and requires significant planning and development time. Many important ECS applications are large, long running, multi-threaded, heavy memory users and therefore are prime candidates for improved memory management.[1]

Improper memory management can result in memory leaks, fast memory usage growth or large application footprints and random crashes. ECS servers are periodically purified for memory leaks and there is a history of progress in this area. Similar work should be expected to continue as development and maintenance continues.

Long running server like applications that are free from memory leaks can nonetheless have significant memory and Central Processing Unit (CPU) usage performance degradation. A common culprit is heap fragmentation. The repeated allocation and deallocation of memory (such as with the new and delete operators of C++) can result in a large number of unusable free blocks of memory. They are free blocks but are interspersed with non-free blocks. They become unusable since they are not contiguous (fragmented) and as time goes by, it becomes harder and harder for the OS to service requests for more memory. Such situations even lead to crashes of other, non-offending applications running in the same box.

There are strategies, tools and software to avoid both memory leaks and fragmentation. This includes but is not limited to:

- Periodic application of purification software (already an ECS practice)
- Software design which uses dynamic memory as little as possible, such as automatic storage or COTS data structures
- Class-level memory management to allocate large chunks of memory instead of one class instance at a time ("Effective C++" by Scott Meyers and "Advanced C++" by James Coplien address this technique)
- Non-class level memory pools and
- COTS heap manager

---

[1] Object-oriented Modeling and design, James Rumbaugh et all, copyright 1991 by Prentice-Hall, Inc. ISBN 0-13-629841-9

Table 4-1 below is provided in case further memory management improvements are desired. Given operator or field input of seemingly inefficient memory or CPU usage, this table can be used to help target specific ECS subsystems, servers and frameworks or classes for improvement. It can be decided to apply some of the approaches at one level (e.g., on one guinea pig server or class) or perhaps experiment with changing the entire ECS C++ system with the use of a COTS heap manager. In any case, a great deal of planning and manpower is required.

Table 4-1. Memory Management Table (1 of 5)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
DPLINGEST	EcDlInPollingService EcDlInProcessingService EcDlInNotificationService	DpCoAlert	Used to describe problematic conditions in DPL INGEST that require attention by the operator.	EcDlInPollingService EcDlInProcessingService EcDlInNotificationService	not passed	EcDlInPollingService EcDlInProcessingService EcDlInNotificationService	One per condition	Alerts are used to convey descriptions of problems to database and are maintained within the service for the lifetime of the problem. Deleted when alert is cleared by operator.
	EcDlInPollingService EcDlInProcessingService EcDlInNotificationService	DpCoMessage	Used to communicate a resource addition, modification, subtraction, suspension or resumption from the operator to the service.	EcDlInPollingService EcDlInProcessingService EcDlInNotificationService	not passed	EcDlInPollingService EcDlInProcessingService EcDlInNotificationService	One per operator action	Messages are used to communicate changes in resource state and are maintained within the service until the change has been made to the services representation of that resource.

Table 4-1. Memory Management Table (2 of 5)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDlInPollingService EcDlInProcessingService EcDlInNotificationService	DpCoResource and children.	Used to describe a resource used by service. For example, ftp hosts, providers.	EcDlInPollingService EcDlInProcessingService EcDlInNotificationService	not passed	EcDlInPollingService EcDlInProcessingService EcDlInNotificationService	One per resource.	Created on startup or when a resource is added to the system. Deleted when a resource is removed from the system or the service is shutdown.
	EcDlInPollingService	DpInPoller	Used to obtain PDR files from a specific polling location	EcDlInPollingService	not passed	EcDlInPollingService	One per polling location	Created on startup. Deleted when a polling location is removed or the service is shutdown
	EcDlInPollingService	DpInResourceCheckTimer	Used to periodically poll for changes in ingest resource properties.	EcDlInPollingService	not passed	EcDlInPollingService	One	Created on startup. Deleted on shutdown.
	EcDlInPollingService	DpInPollingDatabase	Used to handle all interaction with ingest database	EcDlInPollingService	not passed	EcDlInPollingService	One	Created on startup. Deleted on shutdown.
	EcDlInProcessingService	DpInPDRParser	Used to parse a PDR file	EcDlInProcessingService	not passed	EcDlInProcessingService	One per PDR	Created when request is activated. Deleted when parsing has completed.

Table 4-1. Memory Management Table (3 of 5)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDlInProcessingService	DpInPDR	Used to represent the properties of a request during processing.	EcDlInProcessingService	not passed	EcDlInProcessingService	One per request activated	Created when request is activated. Deleted when request processing is complete.
	EcDlInProcessingService	DpInGranule	Used to represent the properties of a granule during processing.	EcDlInProcessingService	not passed	EcDlInProcessingService	One per granule in request	Created when request is activated. Deleted when request processing is complete.
	EcDlInProcessingService	DpInFile	Used to represent the properties of a file during processing.	EcDlInProcessingService	not passed	EcDlInProcessingService	One per file in a granule	Created when granule is activated. Deleted when granule processing is complete.
	EcDlInProcessingService	DpInProcessingDBInterface	Used to handle all interaction with ingest database	EcDlInProcessingService	not passed	EcDlInProcessingService	One	Created on startup. Deleted on shutdown.
	EcDlInNotificiationService	EcDfAutoDispatcher	Email notification queue	EcDlInNotificiationService	not passed	EcDlInNotificiationService	One	Created on startup. Deleted on shutdown.
	EcDlInNotificiationService	EcDfAutoDispatcher	File transfer notification queue	EcDlInNotificiationService	not passed	EcDlInNotificiationService	One	Created on startup. Deleted on shutdown.
	EcDlInNotificationService	DpInServerMessagesTimer	Used to periodically poll for changes in ingest resource properties.	EcDlInNotificationService	not passed	EcDlInNotificationService	One	Created on startup. Deleted on shutdown.
	EcDlInNotificationService	DpInNotifyPopulateTimer	Used to periodically add new notification actions from ingest database	EcDlInNotificationService	not passed	EcDlInNotificationService	One	Created on startup. Deleted on shutdown.

Table 4-1. Memory Management Table (4 of 5)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
	EcDlInNotificationService	DpInNotifyRemoveCompletedActionsTimer	Used to remove completed notification actions from the ingest database	EcDlInNotificationService	not passed	EcDlInNotificationService	One	Created on startup. Deleted on shutdown.
	EcDlInNotificationService	DpInNotifyEmailAction	Used to perform the notification of ingest via email	EcDlInNotificationService	not passed	EcDlInNotificationService	One per email notification	Created when new email notification is retrieved from database. Deleted when action has been completed.
	EcDlInNotificationService	DpInNotifyFileTransferAction	Used to perform the notification of ingest via file transfer	EcDlInNotificationService	not passed	EcDlInNotificationService	One per file notification	Created when new file transfer notification is retrieved from database. Deleted when action has been completed.
	EcDlInNotificationService	DpInNotifyDatabase	Used to handle all interaction with ingest database	EcDlInPollingService	not passed	EcDlInPollingService	One	Created on startup. Deleted on shutdown.
	EcDlInOdlToXml	OdlToXmlTranslator	Used to store the ringpoint section of the PDR file in xml format	EcDlInOdlToXml	not passed	EcDlInOdlToXml	One	Created on startup. Deleted on shutdown.
	EcDlInOdlToXml	OdlToXmlTranslator	Used to store the PDR file	EcDlInOdlToXml	not passed	EcDlInOdlToXml	One	Created on startup. Deleted on shutdown.

Table 4-1. Memory Management Table (5 of 5)

Subsystem Name	Executable Name (M)	Key Classes	Description (M)	Where Created? (Executable/ process name) (M)	Passed To (Executable/ process name)	Where Deleted? (process name) (M)	Number of Instances (Example – 1 per granule)	Comments/Remarks (Items of special interest. Example - Size per instantiation, never "deleted", etc.)
OMS	EcOmOrderManager	OmSrClientDb OmSrDbInterface	Handles connection and queries to the database server.	EcOmOrderManager	Not passed	EcOmOrderManager	One instance	The memory is deallocated when the server comes down.
	EcOmOrderManager	OmSrDispatchQueue	Keeps track of requests for processing.	EcOmOrderManager	Not passed	EcOmOrderManager	Four instances	The memory is deallocated when the server comes down.
	EcOmOrderManager	OmServer	Main encapsulating class.	EcOmOrderManager	Not passed	EcOmOrderManager	One instance	The memory is deallocated when the server comes down.
	EcOmOrderManager	OmSrDistributionRequest	Stores information related to distribution requests.	EcOmOrderManager	Not passed	EcOmOrderManager	One instance per request	The memory is deallocated when the server comes down, or when present request is terminated in any way.
CLS	Not Applicable							
SSS	Not Applicable							
Toolkit	Not Applicable							

4.1 Data Server Subsystem Overview

The Data Server Subsystem (DSS) subsystem consists of one CSCI called the "Archive Inventory Management" (AIM) CSCI. All other CSCIs were eliminated in previous ECS releases. The AIM CSCI provides the following services:

- Services for adding new ESDTs
- Services for validation of granule metadata during Ingest
- Recording and tracking the location of files within the Archives
- Recording the addition of new Volume Groups within the Archives
- Supplying events to BMGT
- Providing database functionality for BMGT processing data
- Providing a Universal Reference (UR) for each granule ingested
- Managing the removal of granules from the system
- Managing QA metadata for science granules
- Creation and storage of Metadata Control Files (MCF).

Data Server Subsystem Context

Figure 4.1-1 shows context diagrams for the DSS subsystem. These diagrams illustrate the interaction of DSS with other ECS subsystems. DPLIngest was separated out into its own diagram because all the subsystems could not fit into one diagram.

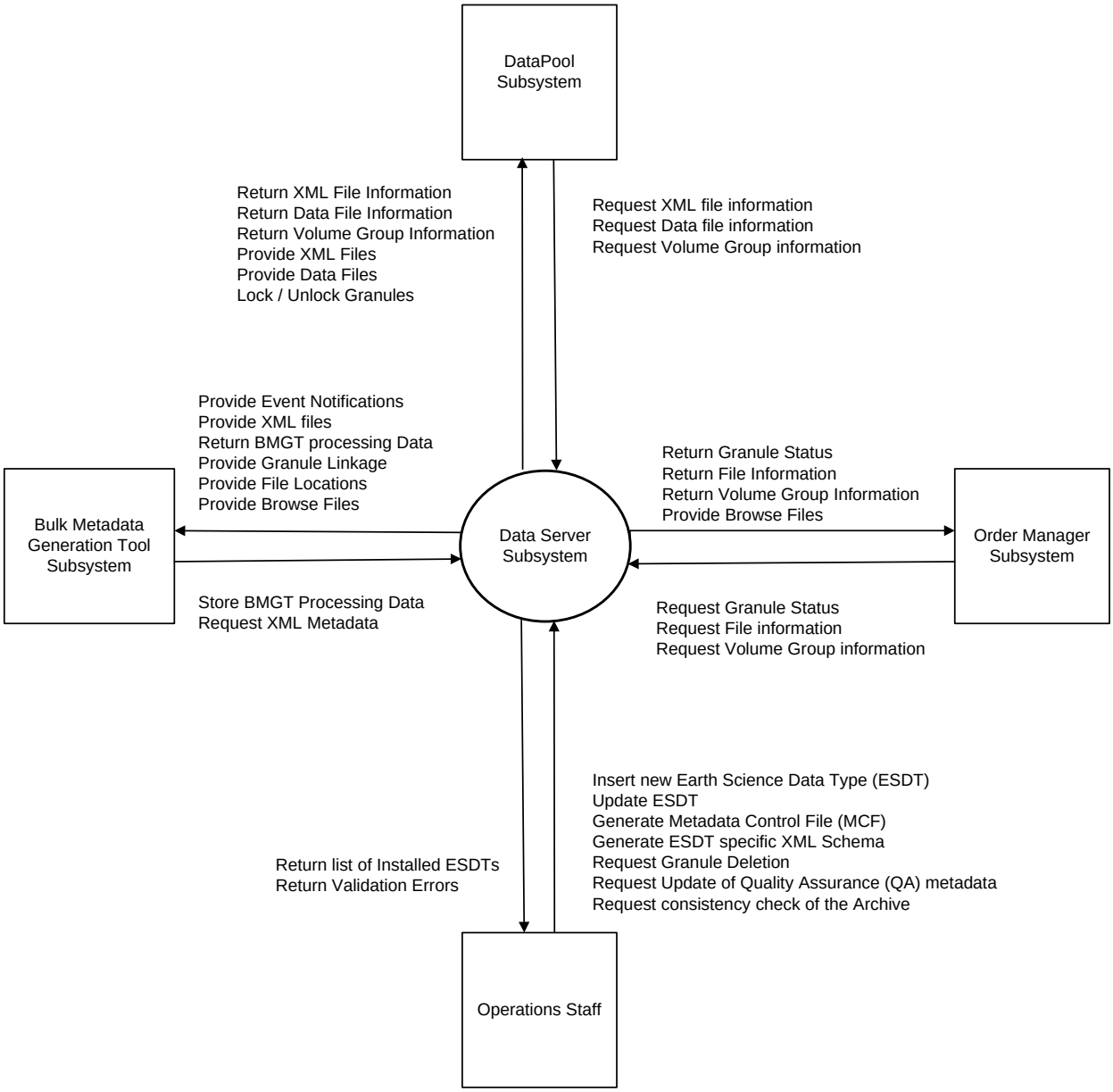

Figure 4.1-1. Data Server Subsystem Context Diagram (1 of 2)

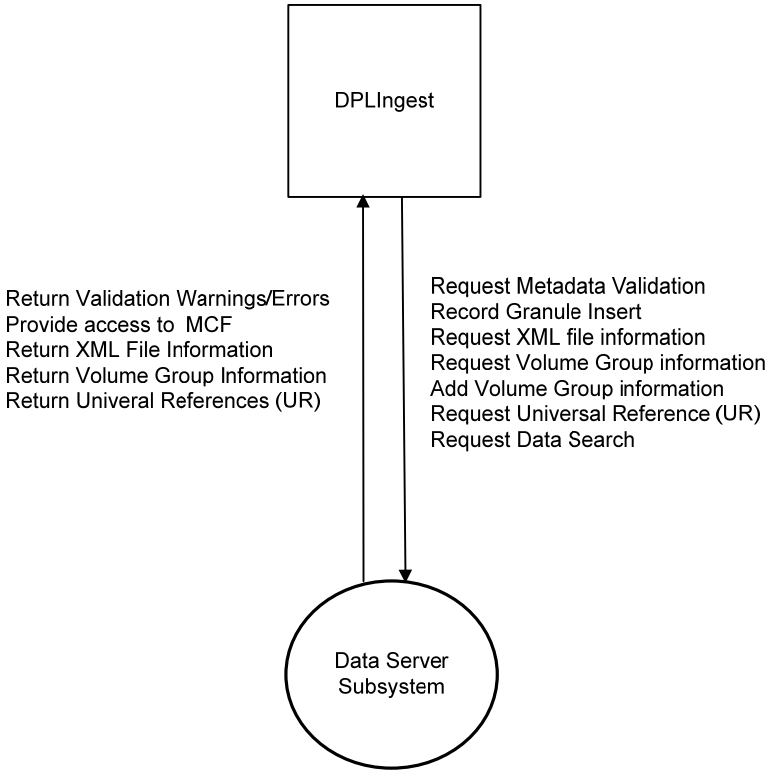

Figure 4.1-1. Data Server Subsystem Context Diagram (2 of 2)

Table 4.1-1 provides a description for each of the interface events shown in the Data Server Subsystem context diagrams. If the interface is shared between multiple subsystems the interface event will only occur one time in the table and all subsystems that share the interface will be listed in the description.

Table 4.1-1. Data Server Subsystem Interface Events (1 of 3)

Event	Interface Event Description
Request XML file information	When the **DPL Ingest** component ingests granules such as Browse, Processing History, or QA, it requests the location of the XML file for the associated science granule from the **DSS Inventory Database** and updates the linkage information within that XML file. **The DataPool Action Driver (DPAD)** component requests XML file information when staging granules into the DataPool.
Request Data File	The **DPAD** component requests data file information from the **DSS Inventory Database** when staging granules into the DataPool.
Request Volume Group Information	The **DPLIngest** component requests volume group information from the **DSS AIM Inventory Database** to determine where to store the files being ingested. The **DPAD** component requests volume group information when staging granules into the DataPool. Additionally the **OMS** and **BMGT** subsystems requests browse granule location information.
Return XML File Information	The **DSS Inventory Database** provides XML file name and location information to the **DataPool and DPLIngest**.
Return Data File information	The **DSS Inventory Database** provides data file name, size, checksum, and location information to the **DataPool**.
Return Volume Group Information	The **DSS Inventory Database** provides a list of directories in the Archive as well as information to determine where each granule is archived to **DataPool, OMS,** and **DPLIngest**.
Provide XML Files	The **DataPool** and **BMGT** subsystems read XML files directly from the **AIM XML Archive.**
Provide Data Files	The **DataPool** reads Science and Browse granule data files directly form the **AIM Granule Archive.**
Lock / Unlock Granules	The **AIM XRU** and **QA Update utility** use the **DataPool database** to lock and unlock granules while they are being processed so that they cannot be accessed by other ECS components.
Request Granule Status	The **OMS** subsystem requests the information from the **DSS Inventory Database** to determine if a granule is “marked for deletion”, or “hidden” and thus can’t be ordered.
Request File Information	The **OMS** requests file name, size, and checksum information from the **DSS Inventory Database** while processing orders.
Return Granule Status	The **AIM Inventory Database** returns information to allow **OMS** to determine if a granule can be ordered.
Return File Information	The **AIM Inventory database** provides information such as file size, checksum, internal and distribution file names to the **OMS**.
Provide Browse Files	The **OMS** and **BMGT** subsystems read browse files directly from the **AIM Granule Archive**.
Insert New Earth Science Data Type	The **DSS ESDT Maintenance GUI** allows **operators** to add ESDTs to the system by providing a descriptor file for the ESDT.
Update ESDT	The **DSS ESDT Maintenance GUI** allows **operators** to modify a limited set of attributes associated with an installed ESDT by providing a replacement descriptor file for the ESDT.

Table 4.1-1. Data Server Subsystem Interface Events (2 of 3)

Event	Interface Event Description
Generate Metadata Control File	The **DSS ESDT Maintenance GUI** allows **operators** to create an MCF file for each ESDT added, it also contains a function to generate new MCFs for each installed ESDT.
Request Granule Deletion	The **DSS Granule Deletion utilities** provides the **operator** the ability to identify granules to be deleted, mark them for future deletion, return them from a "marked for deletion" to an active status, and physically remove (delete) them from the archives and Inventory database.
Request Update of Quality Assurance metadata	The **DSS Quality Assurance Update Utility** (QAUU) provides the **operator** the ability to modify/add metadata related to the quality of the granule to the XML file associated with the granule.
Request consistency check of the Archive	The **DSS Archive Check utility** allows the **operator** to check the files stored in the archives against the records stored in the Inventory database.
Return list of Installed ESDTS	The **DSS AIM ESDT Maintenance GUI** provides **operator** with a list of ESDTs installed within the Inventory.
Return Validation Errors	The **DSS AIM ESDT Maintenance GUI**, when inserting or updating an ESDT, reports to the **operator** any errors found when processing the Descriptor.
Store BMGT Processing Data	The **DSS Inventory Database** provides a storage location for **BMGT** processing data, including the events to be processed.
Request XML Metadata	The **DSS Inventory Database** provides the location of XML files within the DSS XML Archive.
Provide Event Notifications	The **DSS Inventory Database** records real-time events for **BMGT** to process.
Provide XML files	The **BMGT** reads XML metadata from the **DSS XML Archive** (both descriptors and granule XML files) when processing collections and granules. The descriptors are stored in ODL format and are translated to XML for BMGT.
Provide Granule Linkage	The **DSS Inventory Database** is used by **BMGT** to determine the Browse granules that are associated with the Science Granules being processed.
Provide File Locations	The **AIM Inventory Database** provides the location of XML metadata files and Browse files to the **BMGT**.
Request XML Metadata Validation	The **DPLIngest** component uses the **DSS XML Validation Utility** (XVU) to validate the XML metadata file associated with each granule ingested.
Record Granule Insert	The **DPLIngest** component uses the **DSS Inventory Insert Utility (IIU)** to record critical metadata about each granule ingested. The metadata is passed to the IIU via an XML file. Note: DPLIngest will copy the XML file to the AIM XML Archive.
Add Volume Group Information	The **DPLIngest GUI** component uses the **DSS Inventory database** to store information about new volume groups within the archive.
Request Universal Reference	The **DSS Inventory Database** provides a UR to the **DPL Ingest** component for each granule ingested.
Request Data Search	The **DPLIngest** sends a search request to the **DSS Inventory Database** for a granule corresponding to a particular ESDT short name and version, which has a particular local granule id.

Table 4.1-1. Data Server Subsystem Interface Events (3 of 3)

Event	Interface Event Description
Return Validation Warnings/Errors	The **DSS XVU** returns a list of warning and/or error messages to **DPL Ingest** if problems were found during the validation of the granule metadata.
Provide access to MCF	The **DPLIngest** component retrieves the appropriate MCF for the ESDT from the **DSS XML Archive** when ingesting a non-SIPS granule.
Return Universal Reference	The **DSS Inventory Database** provides a UR to **DPLIngest** for each granule ingested.

Data Server Subsystem Structure

The DSS is one CSCI and two HWCIs:

- The Archive Inventory Management (AIM) CSCI catalogs earth science data as logical collections. Each of these collections is referred to as an "Earth Science Data Type" (ESDT) and the members of the collection are referred to as "Granules." The AIM CSCI services requests to add new collections to the inventory, add individual granules to existing collections, update granules within a collection, check the consistency of the Inventory database and the Archive file systems, and remove individual granules or whole collections of granules.
- The XML Archive HWCI stores XML files for each science granule in the Inventory as well as descriptor files, MCFs, and XML schema files used for validating XML.
- The Granule Data Archive HWCI provides high-capacity system for the long-term storage of data files.

Detailed information on hardware/software mapping, hardware diagrams, disk partitioning, etc., can be found in 920-TDx-00x, the 921-TDx-00x, and the 922-TDx-00x series of baseline documents. These documents are located at the web site http://pete.hitc.com/baseline/index.html and click on the Technical Documents button.

Use of COTS in the Data Server Subsystem

- RogueWave's Tools.h++

  The Tools.h++ class libraries provide libraries of object strings and collections. These class libraries are statically linked and delivered with the custom code installation. This library is only used by the Archive consistency checking utility.

- Rogue Wave's Net.h++

  ToolsPro.h++ is a C++ class library, which includes the net.h++ class library, which provides an object-oriented interface to Inter-Process Communication (IPC) and network communication services. The Net.h++ framework enables developed code to be portable

to multiple operating systems and network services. This library is only used by the Archive consistency checking utility.

- Sybase Adaptive Enterprise Server (ASE)

  The Sybase ASE provides the capabilities to retrieve, query, insert, update, and delete database records.

- Boeing Autometric's Spatial Query Server

  The Spatial Query Server (SQS) provides the capability to store and search spatial metadata. It is used by the DSS AIM's IIU and XVU components for inserting and validating (respectively) spatial metadata.

- Sybase Open Client / CT_LIB

  The Sybase Open Client provides access between DSS custom code and the Sybase ASE DBMS.

Use of shareware products:

- XERCES

  A library of software for parsing and manipulating XML, it provides the XVU, IIU, and the ESDT Maintenance GUI the ability to process XML files.

- JConnect3

  The XVU, IIU, and the ESDT Maintenance GUI use JConnect3 to access the Sybase Database Server.

- Standard Java runtime environment

  The XVU, IIU, and the ESDT Maintenance GUI use the standard Java runtime environment to execute their java processes and to provide a set of supporting functionality.

- Java Server Faces

  JavaServer Faces provides a web based application framework and user interface components for the ESDT Maintenance GUI. This includes handling navigation between pages, user input, and display.

4.1.1 Archive Inventory Management Software Description

The Archive Inventory Management (AIM) CSCI is composed of several software components. Some of these components were moved to the AIM CSCI from other DSS CSCIs while others are new components used to replace SDSRV functionality. The existing components that are moved to AIM are:

- Granule Deletion utilities
  - EcDsBulkSearch

    - EcDsBulkDelete
    - EcDsBulkUndelete
    - EcDsDeletionCleanup
- Quality Assurance Update utility
- Archive Check Utility

The new software components included in the AIM CSCI are:

- ESDT Maintenance GUI
- XML Validation Utility
- Inventory Insert Utility
- XML Replacement Utility
- XML Archive Check Utility

The AIM CSCI is used by other ECS CSCIs to manage the XML metadata and storage location for each granule within the ECS inventory. Table 4.1-2 lists the components of the AIM CSCI along with a brief description of the component.

Table 4.1-2. AIM Software Components (1 of 3)

Process	Type	Hardware CI	COTS / Developed	Functionality
EcDsBulkSearch.pl	Command line utility	OMLHW	Developed	EcDsBulkSearch.pl utility provides a command line operator interface for creating a list of granules to be used for Bulk Delete or Bulk Undelete operations.
EcDsBulkDelete.pl	Command Line Utility	OMLHW	Developed	The EcDsBulkDelete.pl utility provides a command line operator interface for deleting granules (marking the granule so that it can't be accessed and making it eligible for future removal) in the Inventory database.
EcDsBulkUndelete.pl	Command line utility	OMLHW	Developed	The EcDsBulkUndelete.pl provides a command line operator interface for changing granules that were previously marked as deleted to an "active" state in the Inventory database.

Table 4.1-2. AIM Software Components (2 of 3)

Process	Type	Hardware CI	COTS / Developed	Functionality
EcDsDeletionCleanup.pl	Command line utility	OMLHW	Developed	The EcDsDeletionCleanup.pl provides a command line operator interface for removing the metadata and data files associated with granules that were previously marked for deletion. This utility removes all references to the deleted granules from the AIM CSCI.
EcAmQAUpdateUtility	Command line utility	OMLHW	Developed	The Quality Assurance Update Utility (QAUU) is a java command line utility that updates QA information in the XML Archive and possibly the XML files within the public DataPool. It processes an input file specifying the QA attributes that are being changed along with a specification for the granules to be updated.
EcDsAmArchiveCheckUtility	Command line utility	DPLHW	Developed	The Archive Check is a C++ utility that compares the records of the AIM Inventory database to files in the XML Archive and Granule Archive. It reports discrepancies for both missing files and missing database records.
ESDT Maintenance GUI	GUI	DPLHW	Developed	The ESDT Maintenance GUI is a web based java application that installs, updates, and deletes ESDTs from the system. It can also be used for informational purposes to see which ESDTs are installed in the system and to view an individual descriptor. The GUI is also responsible for generating ESDT specific XML schema files and MCFs from the descriptor and storing them in the XML Archive.
EcDsAmXvu	Utility	DPLHW	Developed	The XML Validation Utility (XVU) is a java process that parses the XML metadata file being ingested and validates it against a set of rules stored in an ESDT specific schema (located in the XML archive).
EcDsAmIiu	Utility	DPLHW	Developed	The Inventory Insert Utility (IIU) is a java process that parses the XML metadata file being ingested and inserts the essential metadata attributes of the granule into the Inventory database.

Table 4.1-2. AIM Software Components (3 of 3)

Process	Type	Hardware CI	COTS / Developed	Functionality
EcDsAmXru	Command line utility	DPLHW	Developed	The XML Replacement utility (XRU) is command line java utility that replaces XML files within the XML Archive and creates an event for the BMGT subsystem to process. The XRU validates the new candidate file prior to replacing the original file in the XML Archive.
EcDsCheckXMLArchive.pl	Command line utility	DPLHW	Developed	The XML Archive Check utility compares the files stored in the XML archive to the entries in the AIM Inventory database. This utility is optimized for processing the XML Archive, while the EcAmQAUpdateUtility is optimized for processing volume groups.
SQS	Server	DBLHW	COTS	The Spatial Query Server is a COTS product used to validate, store, and query spatial objects in the AIM Inventory database.
Sybase	Server	DBLHW	COTS	The Sybase ASE is the primary database engine for ECS. It used for storing ESDT and granule metadata in the AIM Inventory database.

4.1.1.1 AIM Interfaces with DataPool Ingest

Figure 4.1-2 shows the components of the AIM CSCI that interface with the DPLIngest CSCI. Oval shapes are used to indicate AIM processes. The diagram shows the interactions between the AIM components as well as the data stores within the AIM CSCI. Refer to the DPLIngest section of the 305 document for a complete description of DPLIngest processing.

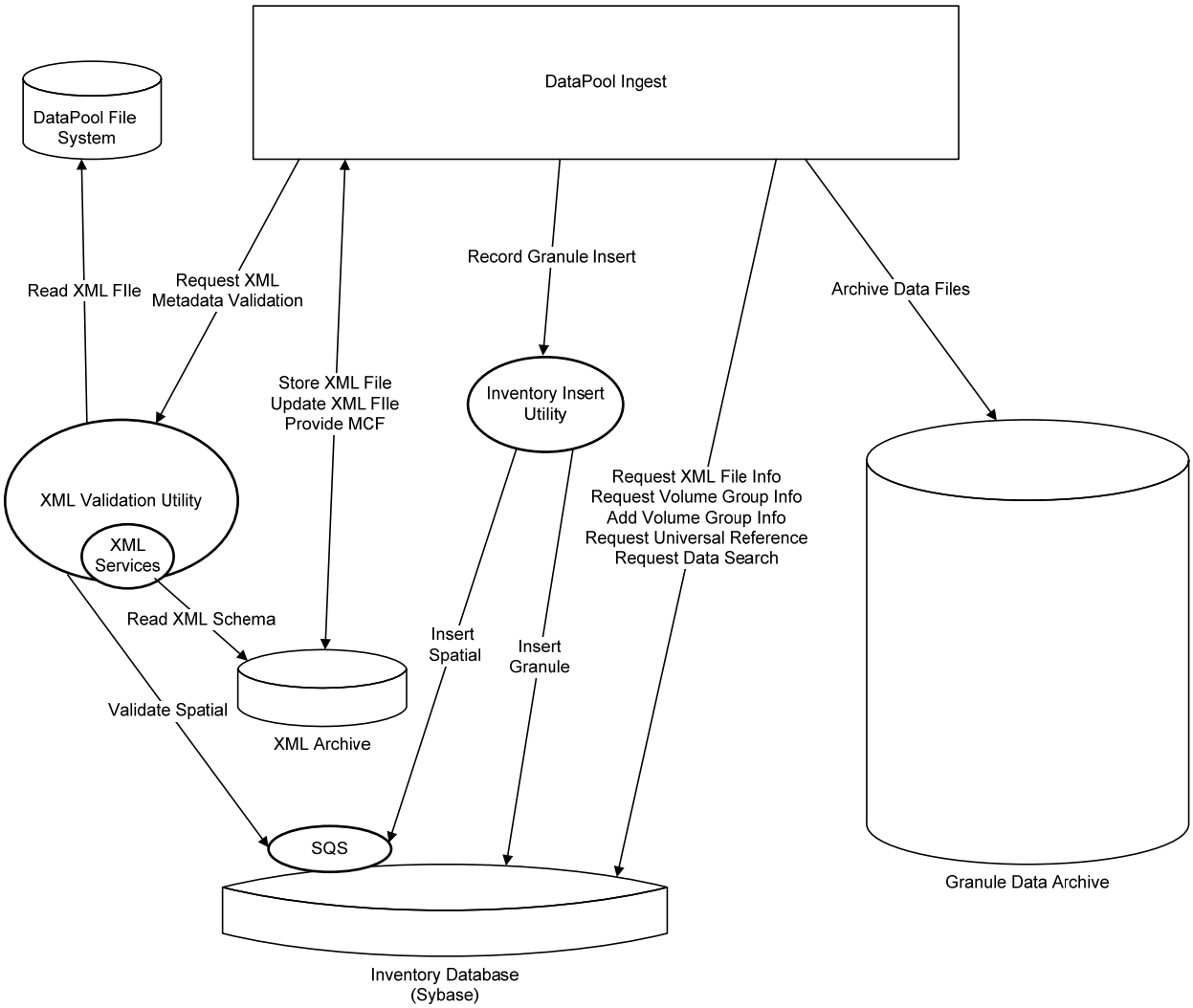

Figure 4.1-2. AIM CSCI Context Diagram (DPLIngest)

Table 4.1-3 explains each of the events/interfaces depicted above.

Table 4.1-3. AIM Interfaces with DPLIngest (1 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Request XML Metadata Validation	One per granule ingested	EcDsAmXvu	EcDlInProcessingService	Each granule ingested into ECS must pass a metadata validation step. ▪ The DPLIngest EcDlInProcessingService component passes the name and location of the XML metadata file within the DataPool hidden directory to the AIM XVU process. ▪ The XVU validates the file using the XML Services jar file and the ESDT specific schema located in the XML Archive. ▪ The XVU processes the XML "post validation information set" to determine the outcome of the XML validation. ▪ Optional elements that are invalid are removed from the file. ▪ The XVU also performs custom validation on certain elements (for example: it validates spatial metadata using SQS and the Inventory database. ▪ The XVU returns a value of 0 or success, 2 for failure, 3 to indicate the metadata passed validation but that some optional elements were removed, and finally a value of 4 indicates the request should be retried at a later time.
Read XML Schema	Once per granule ingested		EcDsAmXvu	When validating a granule XML file, the XVU reads the ESDT specific XML schema from the XML Archive. The schema is created as part of installing the ESDT.

Table 4.1-3. AIM Interfaces with DPLIngest (2 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Read XML File	Once per granule ingested		EcDsAmXvu	The XML file is read into a DOM tree within the XVU process.
Validate Spatial	Once per granule	SQS	EcDsAmXvu	The XVU uses SQS to execute a query that intersects the spatial region contained in the XML file with a known spatial object within the Inventory database. This causes SQS to validate the geometry of the spatial region in the XML file. Validation errors to the spatial region are considered fatal, causing the XVU to return a validation failure.
Store XML File	Once per science granule ingested		EcDlInProcessingService	The EcDlInProcessingService stores an XML metadata file for each science granule ingested. XML files are not archived for Browse, QA, Production History, or Delivered Algorithm Package granules.
Update XML File	Once per browse granule ingested		EcDlInProcessingService	When the EcDlInProcessingService ingests a Browse granule, it adds the Browse ID to the XML metadata file of each science granule that is referenced by the Browse granule linkage information.
Provide MCF	Once per non-SIPs granule ingested		EcDlInProcessingService	When Ingest needs an MCF to pre-process a Non-SIPS ingest request, it reads the MCF directly from the Small File Archive. The directory where the MCFs are stored is a configuration item in Ingest.
Request XML File Info		Stored Procedure	EcDlInProcessingService	Ingest reads the list of XML paths from the Inventory database. Ingest creates new paths in the XML Archive for each ESDT when it ingests the first granule for the ESDT for the current month. This new directory is recorded in the Inventory database as part of the Record Granule Insert event.

Table 4.1-3. AIM Interfaces with DPLIngest (3 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Request VolumeGroup Info	Once at startup of EcDlInProcessingService or after new groups are added.	Stored Procedure	EcDlInProcessingService	Ingest reads the list of open volume groups from the Inventory database at startup.
Add Volume Group	Upon request by DAAC operators	Stored Procedure	DAAC Operator / DPL Ingest GUI	The DAAC operator uses the DPL Ingest GUI to create new volume groups. The GUI inserts these into the Inventory database via stored procedure. When this happens, a message is created in the Ingest database to instruct Ingest to refresh the cached list of volume groups.
Request Universal Reference	Once for each granule Ingested	Stored Procedure	EcDlInProcessingService	Ingest accesses the Inventory database to get the next available granule ID. It then uses its configured UR prefix and the ShortName and VersionID of the granule to construct a Universal Reference.
Request Data Search	Once for each Browse granule Linkage	Stored Procedure	EcDlInProcessingService	Ingest receives Local Granule ID values in the linkage section of the Browse ingest request. It searches the Inventory database (via stored procedure) to convert these to granule ID values.
Record Granule Insert	Once for each granule Ingested	EcDsAmIiu	EcDlInProcessingService	Ingest sends a request to the AIM Inventory Insert Utility to record each granule ingested.
Insert Spatial	Once per science granule with spatial attributes	Embedded SQL	EcDsAmIiu	The IIU inserts the spatial metadata into the Inventory database via the Spatial Query Server. The spatial metadata is inserted before the non-spatial metadata. Each use separate database transactions. If the non-spatial insert fails, then the spatial is removed. In the event that the spatial fails due to a duplicate key error, the IIU assumes it is processing a retry of the metadata insert and proceeds to the non-spatial insert.

Table 4.1-3. AIM Interfaces with DPLIngest (4 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Insert Granule	Once per granule ingested	*Stored procedure*	EcDsAmIiu	The IIU records non-spatial information such as the temporal metadata, the file information, and the XML file location in the Inventory database.
Archive Data Files	Once per granule ingested		EcDlInProcessingService	Ingest copies the data files for the granule into the data archive. The location/directory is determined by executing an Inventory database stored procedure to compare the granule metadata with the DsStVolumeGroup table.

4.1.1.2 AIM CSCI interfaces with DAAC Operations Staff

There are several interfaces between the DAAC operations staff and the Archive Inventory Management (AIM) CSCI. These can't be show in one diagram so this section is broken up into several sections, each explaining a common set of DAAC operator interfaces. Each figure is followed by a table that describes the interfaces / events shown in the figure.

4.1.1.2.1 AIM ESDT Maintenance GUI and QA Update utility

Figure 4.1-3 shows the DAAC operator interfaces with the ESDT Maintenance GUI and the QA Update utility and interaction of these components with other AIM components.

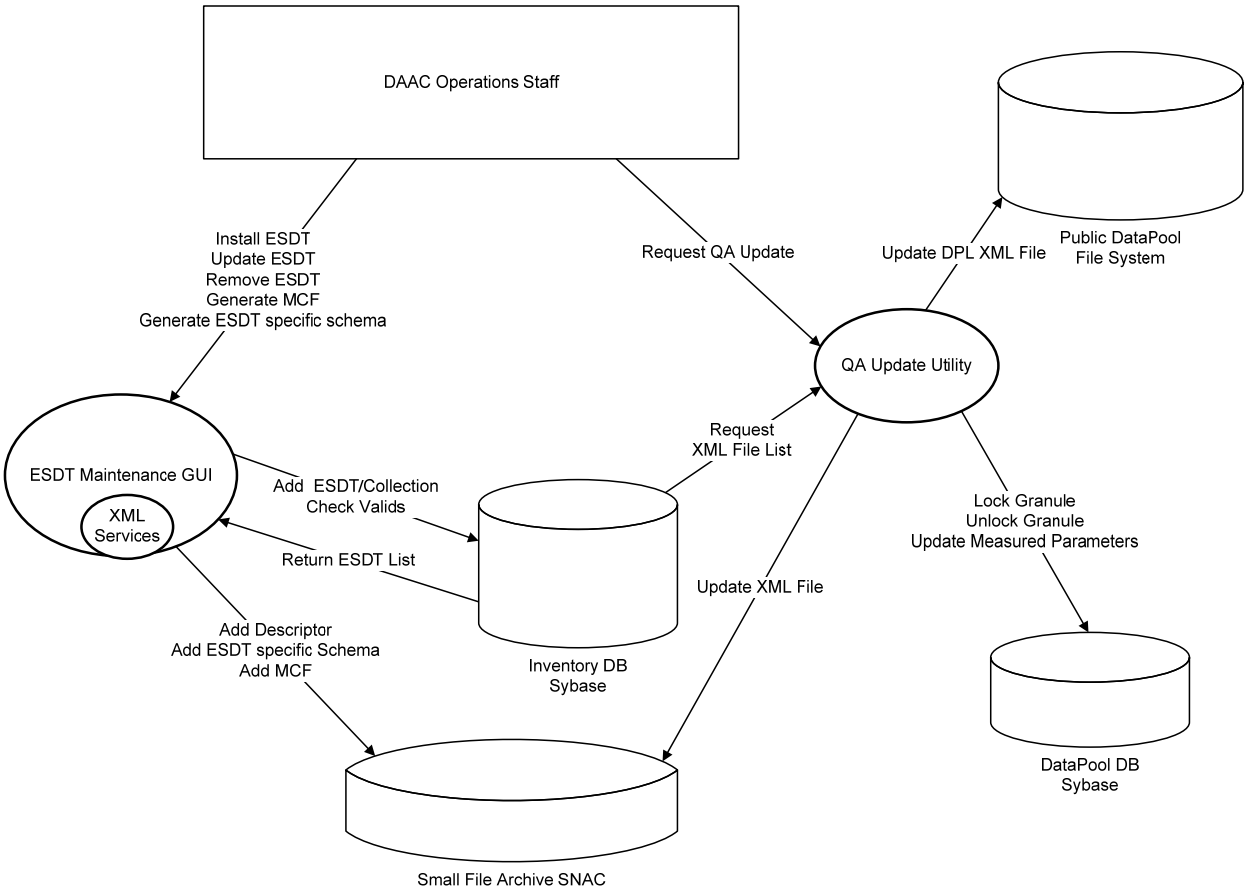

Figure 4.1-3. AIM Interfaces with DAAC Operators (ESDT Maintenance GUI and QA Update utility)

Table 4.1-4 describes each of the events depicted above.

Table 4.1-4. AIM Interfaces with DAAC Operators (ESDT GUI, QA Update) (1 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Install ESDT	Once for each new ESDT	ESDT Maintenance GUI	DAAC Operations	The DAAC operators install ESDTs using the ESDT Maintenance GUI. The GUI performs the following: • Reads the descriptor files from a configured directory. • Converts the descriptor to XML. • Records the ESDT in the Inventory database. • Validates the descriptor against an ESDT XML schema using the XML services module. • Validates certain elements of the descriptor against valids stored in the Inventory database. • Adds the Collection entry to the Inventory database. • Registers an Insert Event in the Spatial Subscription Server database. • Extracts an MCF from the descriptor and stores it in the XML archive. • Builds an ESDT specific XML schema file and stores it in the XML Archive. • Records the completion of the install in the Inventory database.

Table 4.1-4. AIM Interfaces with DAAC Operators (ESDT GUI, QA Update) (2 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Update ESDT	When ever an ESDT descriptor file changes	ESDT Maintenance GUI	DAAC Operations	When an ESDT is updated the ESDT Maintenance GUI performs the following: • Reads the new descriptor file from the configured directory. • Converts the descriptor to XML. • Records the update of the ESDT in the Inventory database. • Validates the descriptor against an ESDT XML schema using the XML services module. • Updates the Collection entry in the Inventory database. Note: changes to metadata such as spatial search type, Product Specific Attribute definitions, and other attributes that could invalidate existing metadata are not supported. • Extracts the MCF the from the descriptor and stores it in the XML archive • Builds an ESDT specific XML schema file and stores it in the XML Archive. • Records the completion of the update in the Inventory database.

Table 4.1-4. AIM Interfaces with DAAC Operators (ESDT GUI, QA Update) (3 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Remove ESDT	Upon Operator request (after all granules are removed)	ESDT Maintenance GUI	DAAC Operations	ESDTs can be removed from the system if all the granules are physically deleted. The ESDT Maintenance GUI performs the following: • Marks the ESDT as being removed in the Inventory database • Removes the MCF and XML schema from the XML archive • Removes the Insert Event from the SSS database • Removes any remaining XML metadata file directories from the XML archive • Removes the collection from the Inventory database • Removes the ESDT information from the Inventory database.
Generate MCF	Once per ESDT Install Update or upon direct request by the operator		ESDT Maintenance GUI	The ESDT Maintenance GUI extracts the MCF section from the descriptor file and stores it as a separate file in a configured MCF directory within the XML archive.
Generate ESDT specific schema	Once per ESDT Install or Update or upon direct request by the operator		ESDT Maintenance GUI	The ESDT Maintenance GUI reads the Inventory section of the descriptor file and compares it to a “common” granule XML schema. It adds the common schema elements that match the descriptor entries to the new ESDT specific XML schema file and stores it in the configured descriptor directory of the XML Archive. The rules in the descriptor can be set to customize the element definitions (mandatory/optional or specific domain list) in the ESDT specific schema.
Add ESDT / Collection	Once per ESDT Install	Stored procedure	ESDT Maintenance GUI	The ESDT Maintenance GUI executes several stored procedures to register the ESDT in the Inventory database.

Table 4.1-4. AIM Interfaces with DAAC Operators (ESDT GUI, QA Update) (4 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Check Valids	Once per ESDT Install	Stored procedure	ESDT Maintenance GUI	The ESDT Maintenance GUI checks certain values in the descriptor file (for example: Discipline, Topic, Term Keyword) against a set of valid values stored in the Inventory database.
Return ESDT List	Upon request by Operator	Stored procedure	ESDT Maintenance GUI	The ESDT Maintenance GUI retrieves the list of installed ESDTs and presents them to the Operator.
Add Descriptor	Once per ESDT Install or Update		ESDT Maintenance GUI	The ESDT Maintenance GUI copies or replaces the descriptor file to the configured descriptor directory within the XML archive.
Add ESDT specific schema	Once per ESDT Install or Update		ESDT Maintenance GUI	The ESDT Maintenance GUI stores the ESDT specific granule XML schema in the descriptor directory of the XML archive. This schema is used to validate granules during Ingest.
Add MCF	Once per ESDT Install or Update		ESDT Maintenance GUI	The ESDT Maintenance GUI extracts the Inventory section of the descriptor file and stores it in the configured MCF directory within the XML archive. This file can be accessed directly by DPL Ingest.
Request QA Update	Whenever quality information for a granule or set of granules is available	EcDsAmQauu	DAAC Operations	The DAAC operator submits a file that specifies the granules to be updated along with the new values for the Quality Flags for a Measured Parameter. The QAUU processes the file by uploading it to one of three different request tables (based upon the file format LGID, dbID, or ESDT + Temporal). The request details are copied to a new table where the information is normalized into a set of granules including dbID and Local Granule ID. The QAUU processes the list of granules to be updated in batches.

Table 4.1-4. AIM Interfaces with DAAC Operators (ESDT GUI, QA Update) (5 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Lock Granule	Executed once per batch, locking each granule within the batch	Stored Procedure	EcDsAmQauu	The QAUU coordinates with other processes that access granules in the DataPool by locking the granule in the DataPool database (inserting to the DlOMSGranules table). All the granules within a batch are locked prior to processing the batch.
Request XML File List	Once per batch of granules processed	Stored Procedure	EcDsAmQauu	The QAUU requests the list of XML files that correspond to the granules being processed.
Update XML File	Once per granule processed		EcDsAmQauu	Depending upon the header file line, the QAUU either updates the ScienceQualityFlag or the OperationalQualityFlag, or the AutomaticQuality flag for the supplied Measured Parameter for each granule. It also updates the associated QualityFlagExplanation.
Update DPL XML File	Once per public DPL granule processed		EcDsAmQauu	For each granule within the batch to process, If the granule is in the public DataPool, the QAUU copies the updated XML file from the XML Archive to the DPL file system.
Update Measured Parameters	Once per batch of granules to process	Stored Procedure	EcDsAmQauu	The QAUU updates the quality flags within the DataPool database for each supplied Measured Parameter for each of the granules in the batch. This update occurs as one large database transaction.
Unlock Granule	Once per batch of granules to process	Stored Procedure	EcDsAmQauu	When the QAUU finishes processing a batch of granules, it unlocks them in the DataPool database by removing the entries from the DlOMSGranules table.

4.1.1.2.2 AIM XML Replacement and Granule Deletion utilities

Figure 4.1-4 shows the interactions between the XML Replacement Utility and the other AIM components. There is only one use case for the XRU and it is initiated by the DAAC operator. The diagram also shows an interface to the DataPool database.

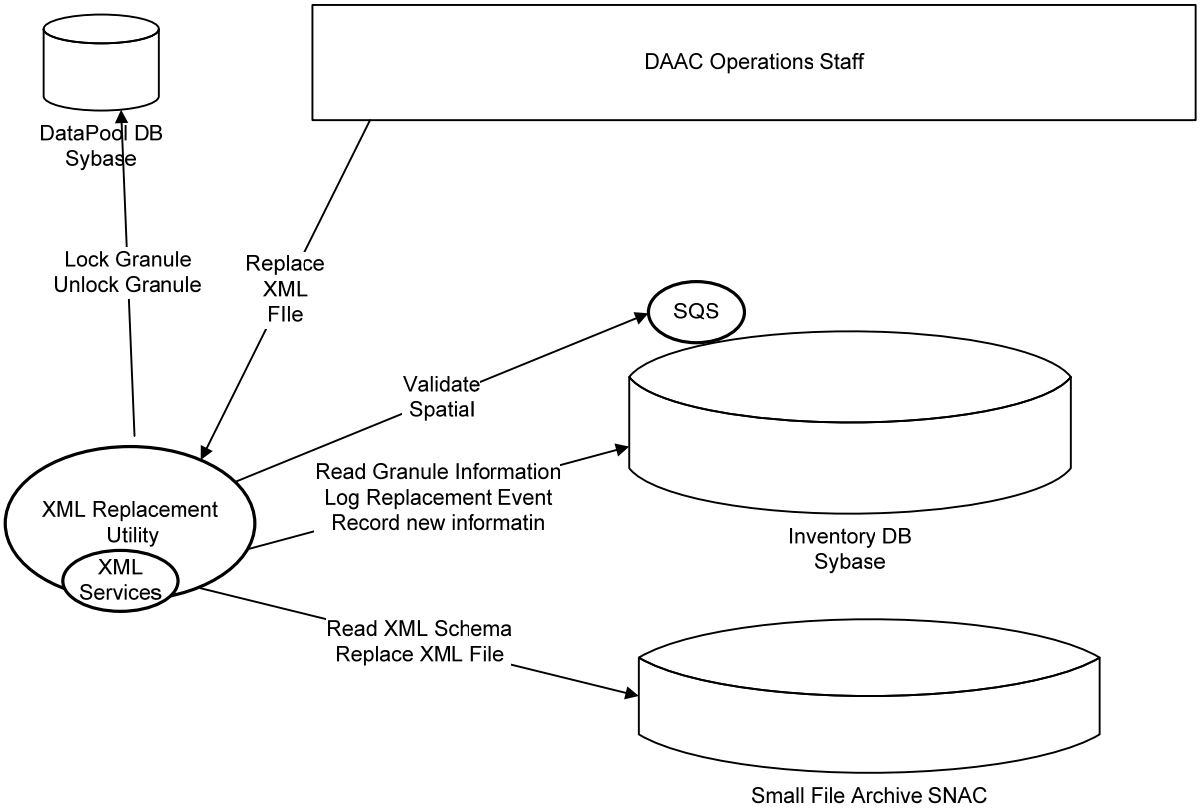

Figure 4.1-4. AIM Interfaces with DAAC Operators (XML Replacement Utility)

Table 4.1-5 describes the interfaces / events depicted above between the DAAC operators and the Granule Deletion utilities.

Table 4.1-5. AIM Interfaces with DAAC Operators (XML Replacement Utility) (1 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Replace XML File	Upon request by the Operator	*EcDsAmXru*	*DAAC Operator*	The XML Replacement utility replaces XML files in the XML archive and logs an event for BMGT to process.
Read Granule Information	Once per XML file to replace	*Stored Procedure*	*EcDsAmXru*	The XRU reads the last update time of the Granule from the Inventory database and compares it to the time in the XML file. It also reads the location of the XML file in the XML archive.

Table 4.1-5. AIM Interfaces with DAAC Operators (XML Replacement Utility) (2 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Lock Granule	Once per XML file to replace	*Stored Procedure*	*EcDsAmXru*	The XRU locks the granule in the DataPool so that it cannot be accessed by other ECS processes while it is being replaced.
Read XML Schema	Once per XML file to replace		*EcDsAmXru*	The XRU reads the XML file into a DOM tree so that the elements can be validated.
Validate Spatial	Once per XML file to replace	*Embedded SQL*	*EcDsAmXru*	The XRU validates the spatial metadata element of the XML file by executing a query against a known value in the Inventory database. This causes SQS to validate the spatial value.
Replace XML File	Once per XML file to replace		*EcDsAmXru*	If the elements of the new XML file are valid, the existing file in the XML Archive is replaced with the new file.
Record new Information	Once per XML file to replace	*Stored Procedure*	*EcDsAmXru*	The XRU records a new size, checksum, and lastUpdate time in the Inventory database for the granule.
Log Replacement Event	Once per XML file to replace	*Stored Procedure*	*EcDsAmXru*	The XRU records an event in the Inventory database to be processed by BMGT.
Unlock Granule	Once per XML file to replace	*Stored Procedure*	*EcDsAmXru*	The XRU removes the lock from the DataPool database once processing is completed.

4.1.1.3 AIM Granule Deletion utilities

Figure 4.1-5 shows the DAAC operator interfaces between the DAAC operator and the XML Replacement and Granule Deletion utilities.

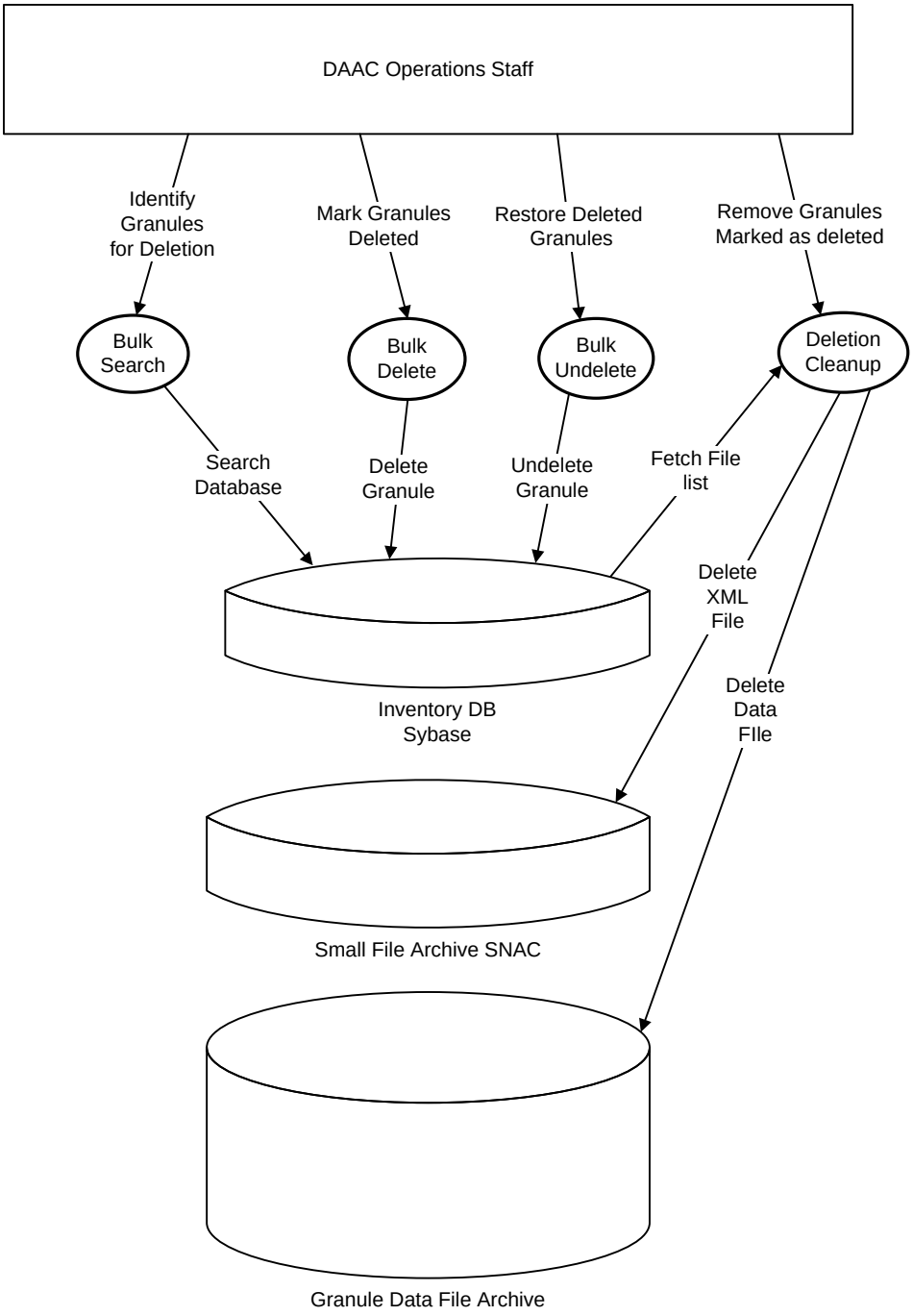

Figure 4.1-5. AIM Interfaces with DAAC Operators (Granule Deletion Utilities)

Table 4.1-6 describes the interfaces / events depicted above between the DAAC operators and the Granule Deletion utilities.

Table 4.1-6. AIM Interfaces with DAAC Operators (Granule Deletion) (1 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Identify Granules for Deletion	Upon request by the Operator	*EcDsBulkSearch.pl*	*DAAC Operator*	The Bulk Search utility searches the Inventory database based upon the supplied arguments to identify a list of granules. The list is stored in a text file that is compatible with both the Bulk Delete and the Bulk Undelete utilities.
Search Database	Once per invocation of the utility	*Dynamic SQL*	*EcDsBulkSearch.pl*	The Bulk Search utility dynamically constructs an SQL statement based upon the arguments given. These can include ShortName, VersionID, Temporal Range, insert time, Local Granule ID, and current deletion status. The output of the search is a list of Granule IDs stored in a text file.
Mark Granules Deleted	Upon request by DAAC Operator	*EcDsBulkDelete.pl*	*DAAC Operator*	The Bulk Delete utility processes a file containing a list of Granule IDs and marks each granule as either “deleted from the Archive” or marks each granule as deleted on the current date. Options exist for marking associated granules. The associated granules (Browse, QA, or PH) can be ignored or they can also be marked as deleted at the same time the science granule is deleted.
Delete Granule	Once per granule deleted	*Stored Procedure*	*EcDsBulkDelete.pl*	The Bulk Delete utility executes a stored procedure to update the granule in the Inventory database with the appropriate deletion status.
Bulk Undelete	Upon request by DAAC Operator	*EcDsBulkUndelete.pl*	*DAAC Operator*	The Bulk Undelete processes a file of Granule IDs and either reverses the Delete From Archive or the Granule Deletion status (based upon the arguments supplied). The utility also has options for processing associated granules.

Table 4.1-6. AIM Interfaces with DAAC Operators (Granule Deletion) (2 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Undelete Granule	Once per granule processed	*Stored Procedure*	*EcDsBulkUndelete.pl*	The Bulk Undelete utility executes a stored procedure to reverse the deletion status for each granule supplied in the input file. Associated granules are also processed after all science granules are individually processed. This is done in a single stored procedure based upon the time a transaction time for the undelete operation.
Remove Granules marked as deleted	Upon request by DAAC operator	*EcDsDeletionCleanup.pl*	*DAAC Operator*	The Deletion Cleanup utility examines the Inventory database for granules that are eligible for removal. The eligible granules have a delete effective date in the Inventory database (set by the Bulk Delete utility) that is less than the time argument passed into the utility. This time argument is referred to as the “lag time” for deleting granules. Once the lag time has passed, the granule is eligible for physical removal from the system.
Fetch File list	Once per invocation of the utility	*Stored Procedure*	*EcDsDeletionCleanup.pl*	The Deletion Cleanup utility uses a stored procedure in the Inventory database to expand the list of granules to delete to cover both the primary and backup volume groups and return the actual list of files.
Delete XML File	Once per file processed		*EcDsDeletionCleanup.pl*	For each file to delete, the Deletion Cleanup utility removes the file from the XML Archive.
Delete Data File	Once per file processed		*EcDsDeletionCleanup.pl*	For each data file to delete, the Deletion Cleanup utility removes the file from both the primary and backup (if applicable) volume group(s).

4.1.1.3.1 AIM Archive Check and XML Archive Check utilities

Figure 4.1-6 shows the interactions between the Archive Check and XML Archive Check utilities and other AIM components. Both utilities are initiated by the DAAC operator.

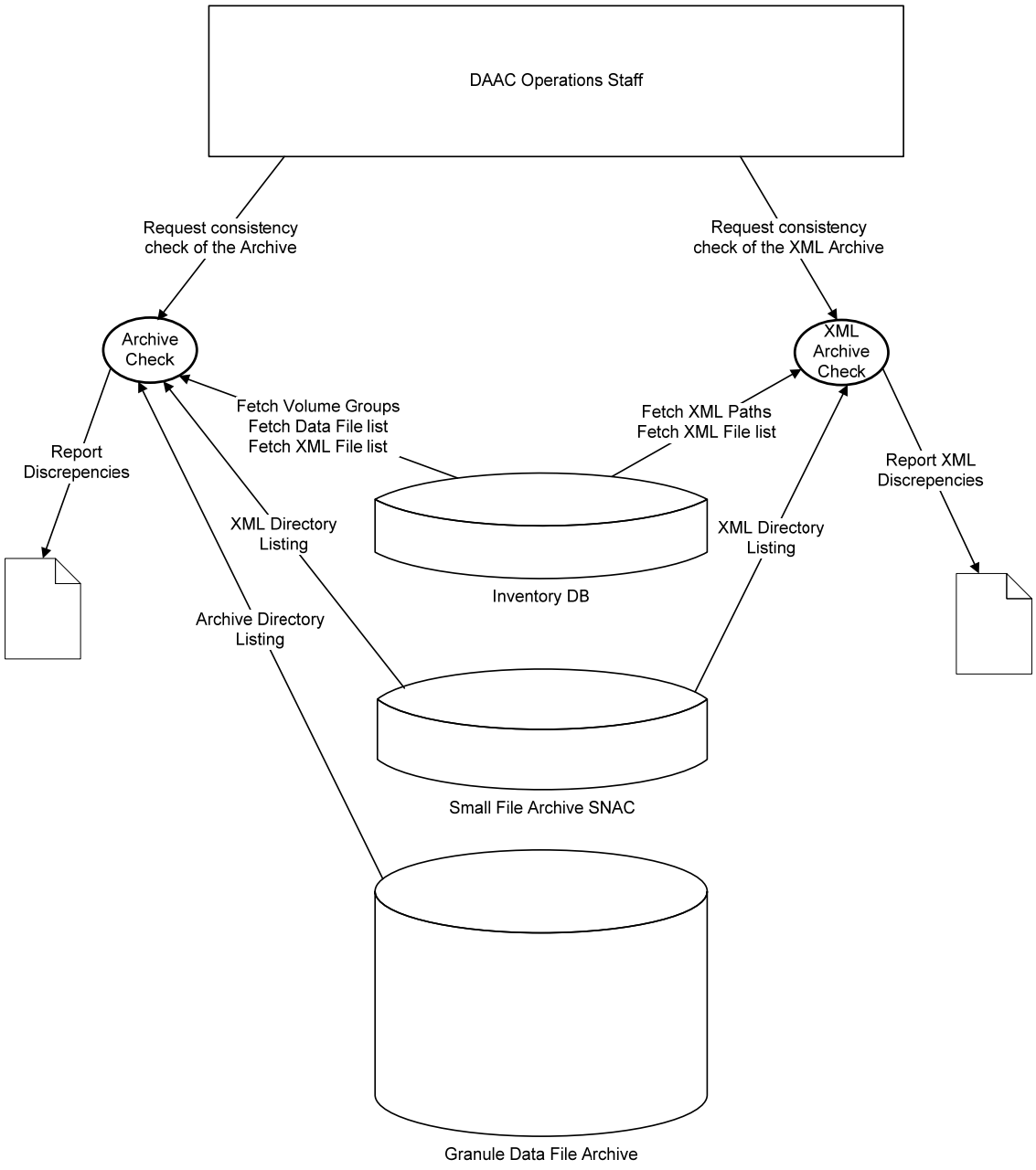

Figure 4.1-6. AIM Interfaces with DAAC Operators (Archive Check Utilities)

Table 4.1-7 describes the interfaces / events depicted above.

Table 4.1-7. AIM Interfaces with DAAC Operators (Archive Check Utilities) (1 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Request consistency check of the Archive	Upon request by Operations	*EcDsAmArchiveCheckUtility*	*DAAC Operations*	The Archive Check utility obtains a list of Volume Groups from the Inventory database and for each Volume Group, the utility compares the list of files in the Volume Group to the entries recorded in the Inventory database that correspond to granules that are mapped to the Volume Group. The utility also checks to make sure every science granule within the Volume Group has an XML file in the appropriate directory of the XML Archive. The mapping of data files to granules is based upon ShortName, VersionID, insert time into the archive, and a comparison of the acquisition time of the granule with a date within the Volume Group to indicate a forward processing or reprocessing use. This rule can be executed by any process to determine the location of the data file. XML Files are stored based upon the Year and Month of the acquisition time of the granule, or the Year and Month of the insert time, if acquisition time is not captured. The Inventory database directly records this association to the absolute directory where the XML file is stored.
Request consistency check of the XML Archive	Upon request by Operations	*EcDsAmXMLArchiveCheck*	*DAAC Operations*	The XML Archive Check utility iteratively processes each XML metadata directory and compares the contents of the directory with the contents of the Inventory database. It begins with getting a list of XML metadata directories from the database. Then loops through each one doing the consistency check.

Table 4.1-7. AIM Interfaces with DAAC Operators (Archive Check Utilities) (2 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Fetch Volume Groups	Once per invocation of the utility	*Stored Procedure*	*EcDsAmArchiveCheckUtility*	The Archive Check utility retrieves the list of Volume Groups to process from the Inventory database.
Fetch Data File list	Once per Volume Group processed	*Stored Procedure*	*EcDsAmArchiveCheckUtility*	The Archive Check utility retrieves the "internal" file name associated with each granule that is mapped to the Volume Group being processed. The mapping is based upon ESDT, insert time into the archive, and a comparison of the acquisition time of the granule with a date within the Volume Group to indicate a forward processing or reprocessing use.
Fetch XML File list	Once per volume group processed Once per XML metadata directory	*Stored Procedure*	*EcDsAmArchiveCheckUtility* *EcDsAmCheckXMLArchive*	The Archive Check utility retrieves a list of XML files associated with each granule to be processed within the Volume Group. The XML Archive Check utility iteratively retrieves the list of XML files in each XML directory.
XML Directory listing	Once per granule processed		*EcDsAmArchiveCheckUtility* *EcDsAmCheckXMLArchive*	The Archive Check utility verifies that each granule being processed has an XML file in the XML Archive. The XML Archive Check utility obtains a list of all files within the XML directory being processed. This will be compared to the contents of the Inventory database which was determined in "Fetch XML File list."
Archive Directory listing	Once per Volume Group processed		*EcDsAmArchiveCheckUtility*	The Archive Check utility obtains a list of files in the Volume Group being processed and compares it to the list of internal file names retrieved from the Inventory database.

Table 4.1-7. AIM Interfaces with DAAC Operators (Archive Check Utilities) (3 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Report Discrepancies	Once per invocation of the utility		*EcDsAmArchiveCheckUtility*	The Archive Check utility writes a discrepancy report to a text file. The report lists granules without data files or XML files in the file systems (phantoms). It also lists data files or XML files that are found in the file systems that are not present in the Inventory database (orphans).
Report XML Discrepancies	Once per invocation of the utility		*EcDsAmCheckXMLArchive*	The XML Archive Check utility writes a discrepancy report to a text file. The report lists granules without XML files in the XML Archive. It also reports the XML files that are found in the XML Archive not present in the Inventory database.

4.1.1.4 AIM interfaces with BMGT

The BMGT subsystem bypasses the AIM software modules and directly accesses the AIM Inventory database, XML Archive, and Granule Data File Archive. Access to the Inventory database is done through stored procedures in the Inventory database. Figure 4.1-7 illustrates these interfaces.

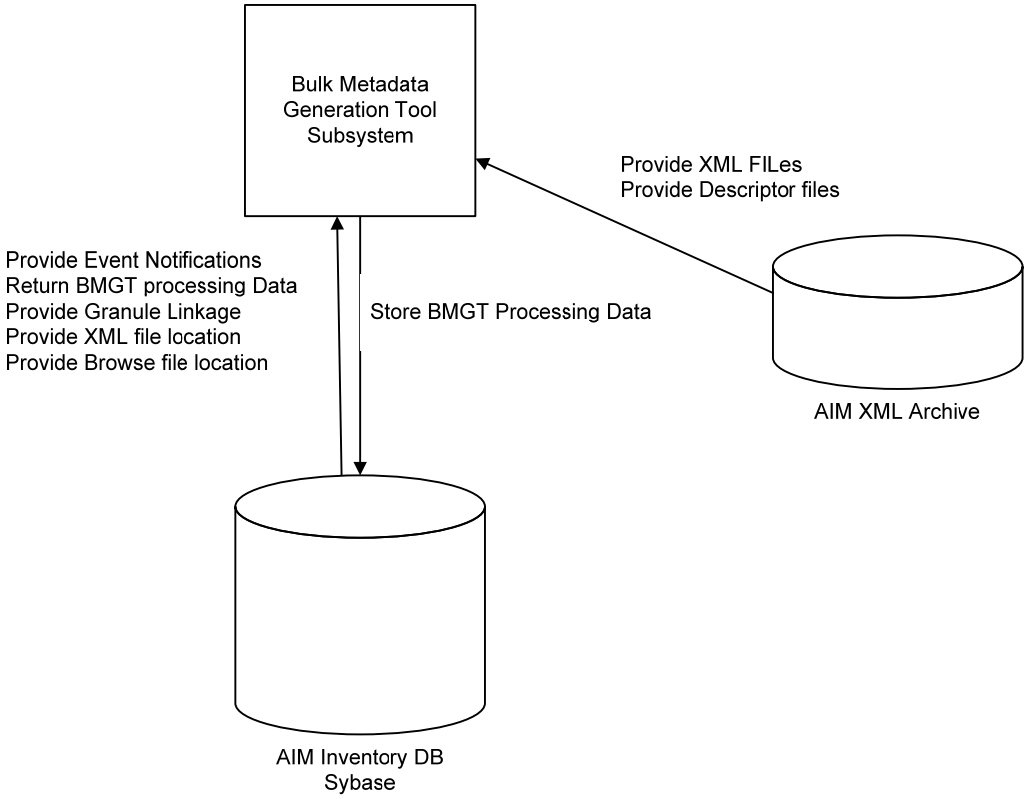

Figure 4.1-7. AIM Interfaces with BMGT

Table 4.1-8 describes each of the interfaces show above.

Table 4.1-8. AIM Interfaces with BMGT (1 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Store BMGT Processing Data	Constant throughout running of BMGT	*Stored Procedures*	*BMGT processes*	The BMGT processes store persistent data related to their processing in the Inventory database. Please refer to the BMGT subsystem for a more complete explanation of the information stored.

Table 4.1-8. AIM Interfaces with BMGT (2 of 2)

Event	Event Frequency	Interface	Initiated By	Event Description
Provide Event Notifications		*Stored Procedures*	*ECS processes*	The Inventory database provides a table for ECS components to log events. The events are retained for a configured time period and cleaned up by an AIM script. These events are used by BMGT to identify the events that occurred within a given cycle (the events are copied to a BMGT table). It is important to configure the event retention time to be long enough for BMGT to capture the events before they are removed.
Return BMGT processing data		*Stored Procedures*	*BMGT processes*	The BMGT retrieves its processing information directly from the Inventory database using a series of stored procedures.
Provide Granule Linkage		*Stored Procedures*	*BMGT processes*	The BMGT obtains Science to Browse granule linkage information directly from the Inventory database.
Provide XML file location	Once per granule processed	*Stored Procedure*	*BMGT processes*	The BMGT obtains the location of the XML file within the XML Archive directly from the Inventory database.
Provide Browse file location		*Stored Procedure*	*BMGT processes*	The BMGT obtains the location of Browse files directly from the Inventory database.
Provide XML Files	Once per granule processed		*BMGT processes*	The BMGT reads XML files directly from the XML archive to produce the ECSMETG products.
Provide Descriptor files	Once per ESDT processed		*BMGT processes*	The BMGT reads Descriptor files directly from the archive to produce the ECSMETC products. These files are in ODL format, and BMGT uses the xmlsvcs framework to convert them to XML.

4.1.1.5 AIM interfaces with the Order Manager and DataPool subsystems

Figure 4.1-8 shows the AIM context diagram for OMS and DPL. Table 4.1-9 shows the AIM interfaces with OMS and DPL.

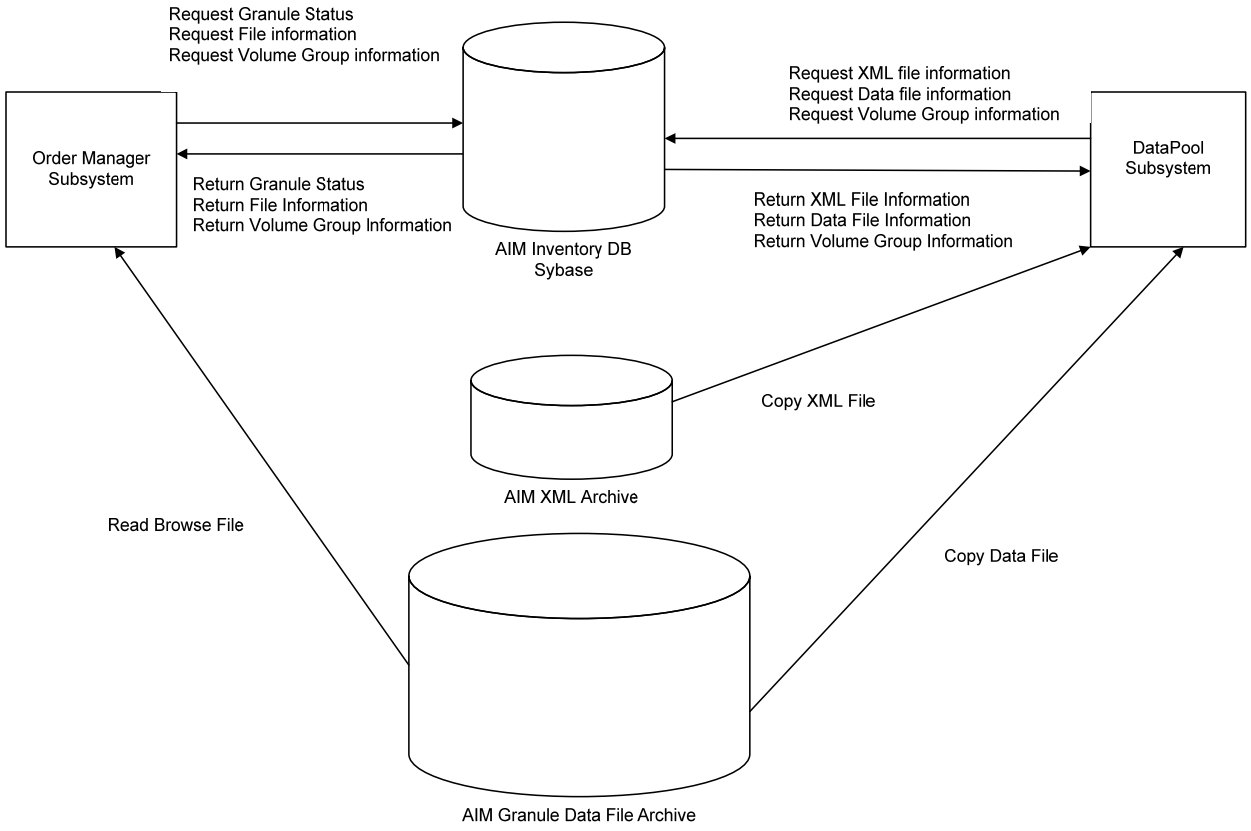

Figure 4.1-8. AIM Context Diagram (OMS and DPL)

Table 4.1-9. AIM Interfaces with OMS and DPL (1 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Granule Status	Once for each granule in an Order	*Stored Procedure*	*EcOmOrderManager*	The Order Manager server executes stored procedures to request the status information (deleted, active) for each granule ordered.
Request File Information	Once for each granule in an Order	*Stored Procedure*	*EcOmOrderManager*	The Order Manager server executes stored procedures to request file information, such as file size, for each granule ordered.

Table 4.1-9. AIM Interfaces with OMS and DPL (2 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Volume Group information	Once per granule processed	*Stored Procedure*	*OMS* *DPL DPAD*	The OMS executes stored procedures to request Volume Group information within the Inventory database. The DataPool Action Driver requests information about the location of the Granule Data file from the Inventory database when staging a granule into the DPL.
Return Granule Status	Once for each granule in an Order	*Stored Procedure*	*EcOmOrderManager*	The Inventory database returns information about the status of the granule within the archive. The Inventory database also maps Local Granule IDs to internal granule IDs for OMS.
Return File Information		*Stored Procedure*	*EcOmOrderManager*	The Inventory database returns information such as file size, checksum, etc. about each granule ordered in OMS.
Return Volume Group Information		*Stored Procedure*	*EcOmOrderManager*	The Inventory database returns information about the location of Browse files within the AIM Granule Archive.
Read Browse File	Once per browse granule ordered		*EcOmOrderManager*	The OMS distributes browse file directly out of the Granule Archive.
Request XML File Information	Once per granule processed	*Stored Procedure*	*DPL DPAD*	The DataPool Action Driver requests information about the XML file from the Inventory database when staging a granule into the DPL.
Request Data file information	Once per granule processed	*Stored Procedure*	*DPL DPAD*	The DataPool Action Driver requests information about the Granule Data file from the Inventory database when staging a granule (Science or Browse) into the DPL.
Return XML File information	Once per granule processed	*Stored Procedure*	*DPL DPAD*	The Inventory database provides information such as file name and path to the DPAD.

Table 4.1-9. AIM Interfaces with OMS and DPL (3 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Return Data File information	Once per granule processed	*Stored Procedure*	*DPL DPAD*	The Inventory database provides information such as file name (internal and distributed), checksum, file size, and LocalGranuleId to the DPAD.
Return Volume Group information	Once per granule processed	*Stored Procedure*	*DPL DPAD*	The Inventory database provides volume group information to the DPAD.
Copy XML File	Once per granule processed		*DPL DPAD*	The DPAD stages granules into the DataPool by copying the XML file directly from the XML Archive to the DataPool file system.
Copy Data File	Once per granule processed		*DPL DPAD*	The DPAD stages granules into the DataPool by copying the granule data file (Science or Browse) directly from the Granule Data File Archive to the DataPool file system.

4.1.2 DSS Error Handling and processing

The Inventory Insert Utility and the XML Validation Utility are Java processes and use a “pipe” based interface with the DPLIngest. They follow the coding standards for Java and contain only one log file. Requests and parameters are passed from the DPLIngest to the IIU and XVU by writing to the pipe. Spaces are used as delimiters between the arguments passed to these utilities. The final processing status of each request is written to the application log along with information to identify the granule being processed.

Both of these utilities return results to the DPLIngest by writing to the pipe. The IIU returns a value of 0 for success if all metadata was successfully inserted. In the case of an error, it returns a value of 1 along with a detailed text message describing the error. This utility doesn’t contain any persistent storage of previous requests; it assumes that DPLIngest will always link a specific granule ID to the same granule. Thus if it encounters a situation where metadata already exists for a given granule ID, the IIU assumes that it previously processed the granule and returns a success result.

The XVU has multiple possible return values. It returns a value of 0 for success; it returns a value of 2 if metadata in the request does not pass the validation process, it returns a value of 3 to indicate that the metadata passed validation but some optional elements were removed (this is considered a “warning” message), and it returns a value of 4 to indicate that errors were encountered attempting to validate the granule and that DPLIngest should try the request again. When the XVU returns 2, 3, or 4 it will also return a detailed text message describing the warning or error. The XVU has no persistent storage of requests, in the event that DPLIngest

validates the same granule more than one time, the XVU will process the XML metadata file without regard to previous any actions.

The ESDT Maintenance GUI responds directly to the operator, thus it displays error messages within the GUI and has a separate screen for displaying validation errors. It also has an application log for capturing processing information and error messages. The Granule Deletion utilities, the Quality Assurance Update utility, the XML Replacement utility, and the Archive Check utilities are command line utilities that interact directly with the Operator. They write to the operator console/xterm and use application logs to show detailed processing flow information and detailed error messages.

4.1.3 DSS Data Stores

Table 4.1-10 provides a description of the data stores for the AIM CSCI, and the conceptual model of the data store. The physical model for the AIM data stores can be found in the AIM Database Design and Schema Specifications for the EED Contract (CDRL 311).

Table 4.1-10. AIM CSCI Data Stores (1 of 2)

Data Store	Type	Description
AIM Inventory	Database	The primary purpose of the AIM Inventory database is to configure ESDTs and track the status and location of granules within the DSS. The Inventory database catalogs information about the following objects: • ESDT definitions • Collection level information • Browse data • Science data (as granules) • Quality Assessments • Delivered Algorithm Packages • Production History

Table 4.1-10. AIM CSCI Data Stores (2 of 2)

Data Store	Type	Description
XML Archive	File System	The XML archive is a SAN attached managed file system that is available to most of the processing blades. It contains directories for the following: ▪ A metadata directory stores an XML metadata file for each Science granule archived in the ECS. The directories for storing XML metadata are first separated by ESDT. Within each ESDT granule directories are partitioned into directories based upon the year and month of the acquisition time. If acquisition time is not recorded for an ESDT, then the insert time of the granule will be used. ▪ A descriptor directory stores a descriptor file along with the associated XML schema file for the ESDT installed in the mode. ▪ An MCF directory stores a Metadata Control File (MCF) for each ESDT installed in the system. ▪ A BMGT directory is used by the BMGT subsystem for persistent storage of production outputs and reports.
Granule Archive	File System	The Granule Archive is the primary data store for the ECS. It consists of a set of managed file systems that are broken up into directories that are assigned to the storage of files for a given ESDT or set of ESDTs. These directories are referred to as Volume Groups and typically have a specific time range of granules that are stored in the directory.

4.2 DPL Ingest Subsystem Overview

The Data Pool Ingest service handles the SIPS ingest interface[1], cross-DAAC ingest, EDOS ingest, ASTER Ingest and Polling without Delivery Record specifically for EMOS. Unlike the classic INGST subsystem, the Data Pool Ingest service will insert the ingested data into the Data Pool, in addition to inserting the ingested data into the archive.

DPL Ingest Subsystem Context

Figure 4.2-1 is the DPL Ingest context diagram. The diagram provides an illustration of the Data Pool Ingest and archiving steps. Table 4.2-1 provides descriptions of the interface events shown in the DPL Ingest Subsystem context diagram.

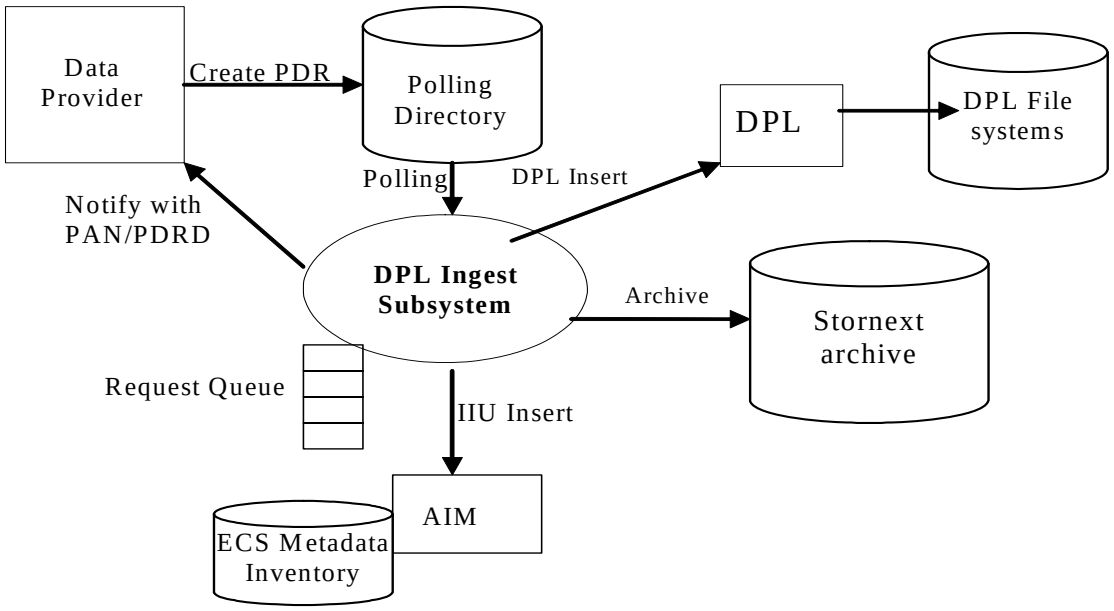

Figure 4.2-1. DPL Ingest Subsystem Context Diagram

[1] EOSDIS Document 423-41-57 Interface Control Document between the ECS and the SIPS Volume 0 revision H

Table 4.2-1. DPL Ingest Subsystem Interface Events

Event	Interface Event Description
Create PDR	SIPS providers place their data and PDR files into a polling directory. The directory can be local, e.g., accessible via a mount point; or remote, i.e., accessible via FTP/SCP. For Polling without Delivery Record provider, the data file is transferred to the polling directory rather than a PDR.
Polling	The DPL Ingest Service polls these directories as configured by the DAAC.
Request Queue	The DPL Ingest Service queues ingest requests for validation and processing. The DPL Ingest Service queues all PDR that it finds. To decide which validated PDR will be processed next, it checks available resources, timestamps and priorities of the requests.
ODL To XML Conversion	The DPL Ingest Service converts the input ODL metadata (if applicable) to XML metadata for insertion into DPL.
XML Metadata Validation	The DPL Ingest Service calls the XML Validation Utility to validate the XML metadata file.
DPL Insert	The granule files are copied into the Data Pool SAN, using hidden directories for that purpose unless the DAAC requested that the data be published on insert.
Archive	The DPL Ingest Service then copies the granules from the hidden Data Pool directories into the StorNext archive.
IIU Insert	The DPL Ingest Service calls the Inventory Insert Utility to insert the granule metadata into the AIM database.
SSS notification	The DPL Ingest service places a record for the Spatial subscription server to decide whether any subscription should fire based on the granule insert.
Notify with PAN or PDRD	The provider is notified of the ingest outcome in the format of PDRD (if PDR validation failed) or PAN (for reasons other than PDR validation failure).
Queue for Publication	The DPL Ingest service places a record for the granule to be published if its collection is marked for publication. The DPAD will then perform the publication.

DPL Ingest Subsystem Structure

The DPL Ingest Subsystem consists of four CSCIs: the Polling Service, the Processing Service, the Notification Service, and the DPL Ingest GUI. The Polling Service is responsible for the provision of work to the service via transferring Product Delivery Records (PDRs) into the system and registering them, or in the case of Polling without Delivery record, creating a PDR for each data file and registering them. The Processing Service picks up registered PDRs and attempt to ingest the data they describe into the Data Pool and the Archive, performing any additional processing required for specific inventory. The Processing component will checkpoint a particular PDR on completion of various steps during processing and register a notification (i.e. PDRD or PAN) for Notification Service to process when the PDR reached a terminal state. Terminal states are Successful, Partial_Failure, Failed, Cancelled, and Partially_Cancelled. Terminal states are conveyed to the provider by means of a Product Acceptance Notification (PAN) or Product Delivery Discrepancy Report (PDRD). The Notification Service will detect registered notifications and deliver them to the provider based on the provider configured notification methods. The DPL Ingest GUI is used to configure, monitor and control the operations of the DPL Ingest Service.

Use of COTS in the DPL Ingest Subsystem

- RogueWave's Tools.h++

  The Tools.h++ class libraries are used by the DPL Ingest Service to provide basic functions and objects such as strings and collections. These libraries must be installed with the DPL Ingest software for any of the DPL Ingest Service processes to run.

- Sybase Open Client / CT_LIB

  The Sybase Open Client provides access between DPL Ingest Service custom code and the Sybase SQL Server DBMS.

- Sybase Server

  The Sybase SQL server provides access for DPL Ingest Service to insert, update and delete DPL Ingest Requests, DPL Ingest configurations, and Operator Interventions. The Sybase SQL Server must be running during operations for the DPL Ingest Service to process DPL Ingest Requests.

- UNIX Network Services

  DNS, NFS, E-mail, FTP, TCP/IP and the other UNIX services provided are obtained from the CSS.

4.2.1 DPL Ingest Computer Software Configuration Item Description

4.2.1.1 DPL Ingest Service CSCI Functional Overview

The DPL Ingest Subsystem consists of four CSCIs: the Polling Service, the Processing Service, the Notification Service, and the DPL Ingest GUI. The Processing Service executes as a process and interacts with the following CSCIs: INGST Database, the Polling Service, the Notification Service, the XML Validation Utility (XVU), the Inventory Insert Utility (IIU), and the Data Pool System (DPL). The Polling Service transfers Product Delivery Records (PDRs) into the system and registers them to the INGST Database. Processing Service retrieves the PDRs from INGEST Database and validates them. If the PDR is valid, Processing Service attempts to ingest the inventory they describe into the Data Pool and Archive, performing any additional processing required for specific inventory. The Processing Service updates the status of a particular PDR on completion of various steps during processing. For an invalid PDR, a PDRD is generated. An operator intervention is created if the request encounters a processing problem. DAAC OPS personnel can use the DPL Ingest GUI to monitor and control the processing of the request. In response to an intervention, the Operator can view the error details, retry the erroneous granule or fail the request if the problem cannot be resolve through retries. Processing Service also generates notification for each request that has reached a terminal state and register notification in INGEST Database. The Notification Service will detect the registered notifications and deliver them to the provider based on the provider configured notification methods. Operator Alert is generated when an internal or external resource failure is detected. When an operator alert is generated, DPL Ingest services will halt dispatching of the requests that are utilizing those failed resources, retries the failed operation that caused the alert (if so configured) and automatically clears the alert if the operation succeeds on retry. Operator can view the operator alerts on DPL Ingest GUI and can manually clear the operator alerts through the GUI.

4.2.1.2 DPL Ingest Service CSCI Context

Figure 4.2-2 is the DPL Ingest Service CSCI context diagrams. The diagrams show the events sent to the DPL Ingest Service CSCIs and the events the DPL Ingest CSCIs send to other CSCIs. Table 4.2-2 provides descriptions of the interface events shown in the DPL Ingest Service CSCI context diagram.

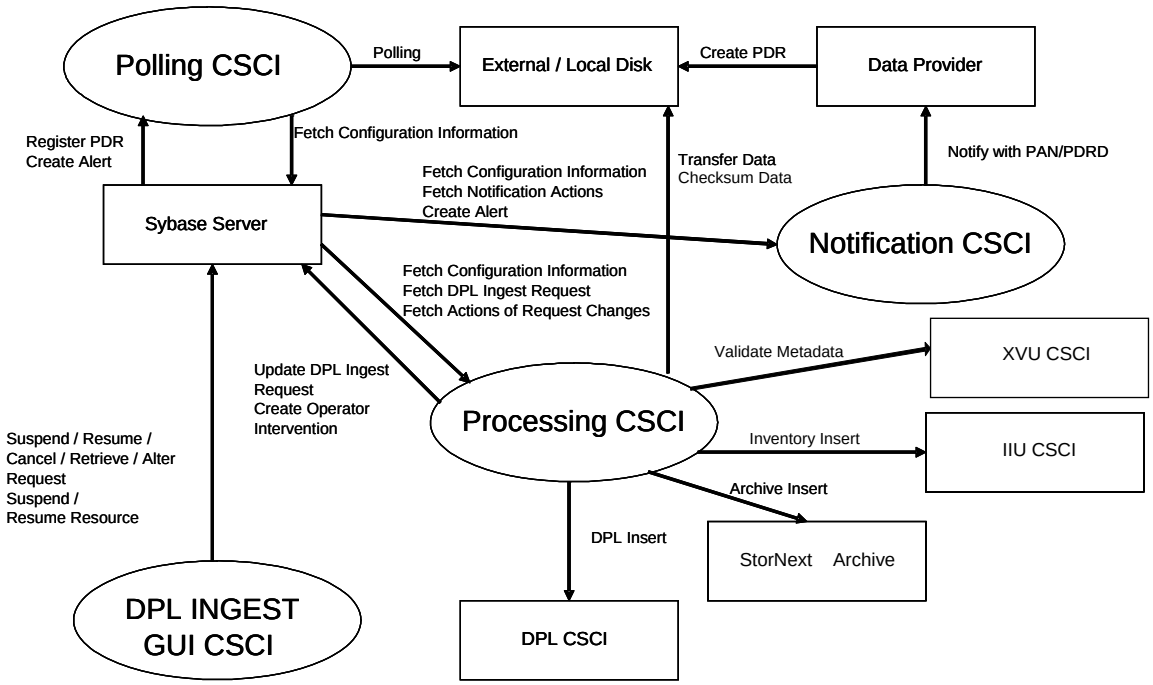

Figure 4.2-2. DPL Ingest CSCI Context Diagram

Table 4.2-2. DPL Ingest CSCI Interface Events (1 of 2)

Event	Interface Event Description
Create PDR	**SIPS providers** place their data and PDR files into a polling directory which will be polled by the Polling CSCI. The directory can be local, e.g., accessible via a mount point; or remote, i.e., accessible via FTP.
Polling	The Polling CSCI polls PDRs from directories (**External/Local Disk**) by **Data Provider** as configured by the DAAC.
Register PDR	The Polling CSCI queues ingest requests for validation and processing into the **Sybase Server (INGST Database)**. The Processing CSCI later queues all PDR that it finds. To decide which validated PDR will be processed next, it checks available resources and DAAC configured priorities.
ODL To XML Conversion	The DPL Ingest Service converts the input ODL metadata (if applicable) to XML metadata for insertion into DPL.
XML Metadata Validation	The DPL Ingest Service calls the **XML Validation Utility** to validate the XML metadata file.
DPL Insert	The Processing CSCI copies the granule files into the **Data Pool SAN**, using hidden directories for that purpose unless the DAAC requested that the data be published on insert.
Archive Insert	The Processing CSCI copies the granule files in to the **StorNext archive**.
IIU Insert	The DPL Ingest Service calls the **Inventory Insert Utility** to insert the granule metadata into the AIM database.
SSS notification	The DPL Ingest service places a record in the **SSS database** for the Spatial subscription server to decide whether any subscription should fire based on the granule insert.
Notify with PAN or PDRD	The Notification CSCI sends notification to the **Data Provider**, it could be immediate via PDRD if PDR validation failed, or later on via a short or long PAN.
Queue for Publication	The DPL Ingest service places a record in the **DPL database** for the granule to be published if its collection is marked for publication. The DPAD will then perform the publication.
Create Alert	The Polling CSCI, Processing CSCI and Notification CSCI creates an alert for resource failures and stores the alert in the **Sybase Server (INGST Database)**.
Fetch Config Info	The Polling CSCI, Processing CSCI and Notification CSCI retrieves the configuration information from **Sybase Server (INGST Database)**.
Update DPL Ingest Request	The Processing CSCI updates DPL Ingest request in the **Sybase Server (INGST Database)**.
Create Operator Intervention	The Polling CSCI, Processing CSCI creates new Operator Intervention for request failures in the **Sybase Server (INGST Database)**.
Fetch DPL Ingest Request	The Processing CSCI retrieves information associated with a DPL Ingest request from the **Sybase Server (INGST Database)**.
Fetch Actions of Request Changes	The Processing CSCI retrieves actions regarding request changes, such as, request priority change, cancel request, suspend request, and update request parameters from the **Sybase Server (INGST Database)**.
Validate Metadata	The Processing CSCI populates the metadata files and sends them to the **XVU CSCI** for validation.

Table 4.2-2. DPL Ingest CSCI Interface Events (2 of 2)

Event	Interface Event Description
Insert Metadata	The Processing CSCI sends the granule metadata to the **IIU CSCI** for insertion into the AIM database.
Fetch Notification Actions	The Notification CSCI retrieves actions regarding request notifications from the **Sybase Server (INGST Database)**.
Suspend/Resume/Cancel/Alter/Retrieve Request	The DPL Ingest GUI CSCI suspends, resumes, cancels, alters and retrieves requests from the **Sybase Server (INGST Database).**
Suspend/Resume Resource	The DPL Ingest GUI CSCI suspends or resumes dispatching to all or selected resources in the **Sybase Server (INGST Database).**
Transfer Data	The Processing CSCI transfers data files from the **External/Local Disk** specified in PDR.
Checksum Data	The Processing CSCI checksum data files from the **checksum information** specified in PDR or calculate the checksum based on the provider and system configuration.

4.2.1.3 DPL Ingest Architecture

The Polling Ingest Interface (EcDlInPollingService) polls accessible file system locations to detect data to be ingested. This process submits a Product Delivery Record (PDR). The Cross-DAAC Ingest Interface (EcDlInEmailGWServer) polls a configured directory for distribution notices (flat files converted from email messages). This process detects the distribution notice files and creates Product Delivery Record files, which are put in a polling directory and polled by the Polling Ingest Interface.

The Polling Ingest Interface queues ingest requests into the Sybase Server (INGST database) for Processing Service to pick up. The Processing Interface queues all PDR that it finds, to decide which validated PDR will be processed next, it checks available resources, timestamps and priorities of the request. The Processing Interface validates metadata using the XVU and inserts the granules into the AIM inventory. The Processing Interface copies the granule files into Data Pool SAN, using hidden directories for that purpose unless the DAAC requested that the data be published on insert. The processing Interface copies the granule files into the StorNext archive. The processing service copies the XML metadata file to the small file archive.

Figure 4.2-3 is the DPL Ingest CSCI architecture diagram. The diagram shows the events sent to the DPL Ingest CSCI processes and the events the DPL Ingest CSCI processes send to other processes.

Note: System startup and shutdown - Please refer to the release-related, current version of the Mission Operations Procedures for the EED Contract document (611) and the current EED Contract Training Material document (625).

4.2.1.4 DPL Ingest Process Descriptions

Table 4.2-3 provides the descriptions of the processes shown in the DPL Ingest CSCI architecture diagram (Figure 4.2-3).

Table 4.2-3. DPL Ingest CSCI Processes (1 of 2)

Process	Type	Hardware CI	Source	Functionality
EcDlInPollingService	Server	DPLHW	Developed	• Detect new Product Delivery Records (PDRs) and transfer them into system. • Creates a unique identifier for the request. • Register request.
EcDlInGui	GUI	INTHW	Developed	Provides Maintenance and Operations (M&O) personnel the capability, via GUI Interface, • To modify ingest configuration parameters. • To monitor the status of ongoing ingest requests, to suspend, resume, cancel, alter or retrieve DPL ingest requests. • To suspend or resume resource.
EcDlInProcessingService	Server	DPLHW	Developed	• Ingests granules associated with ingest requests (PDRS) into the Datapool and archive. • Registers granule information with AIM • Manages the DPL ingest request traffic and the processing of the DPL ingest requests. • Provides the capability to process multiple ingest requests concurrently by placing the request in a queue. • In the event of a failure, the EcDlInProcessingService process restores on-going requests from the Ingest database.
EcDlInNotificationService	Server	DPLHW	Developed	• Send the end-user Notification, either Product Acceptance (PAN) or Product Delivery Discrepancy Report (PDRD), on completing a ingest request.

Table 4.2-3. DPL Ingest CSCI Processes (2 of 2)

Process	Type	Hardware CI	Source	Functionality
Sybase	Server	ACMHW	COTS	• Stores and provides access to the DPL Ingest Service internal data. In particular, the database stores the Ingest operations databases – DPL Ingest History Logs and the DPL Ingest request checkpoint state, and template information. See Section 4.2.1.6 DPL Ingest Data Stores.

EMD Baseline Information System (EBIS) Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.2.1.5 DPL Ingest Process Interface Descriptions

Table 4.2-4 provides descriptions of the interface events shown in the DPL Ingest CSCI Architecture diagram.

Table 4.2-4. DPL Ingest CSCI Process Interface Events (1 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Create PDR	One per request	Directories on remote or local disk	External Data Provider	SIPS providers place their data and PDR files into a polling directory which will be polled by the EcDlInPollingService. The directory can be local, e.g., accessible via a mount point; or remote, i.e., accessible via FTP/SCP.
Polling	One per request	Directories on remote or local disk	*Process:* EcDlInPollingService *Class:* DpInPoller	The EcDlInPollingService polls PDRs from directories (External/Local Disk) by Data Provider as configured by the DAAC.

Table 4.2-4. DPL Ingest CSCI Process Interface Events (2 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Register PDR	One per request	*Process*: Sybase Server (COTS)	*Process*: EcDlInPollingService *Class*: DpInPoller	The EcDlInPollingService queues ingest requests for validation and processing into the Sybase Server (INGST database). The EcDlInProcessingService later queues all PDR that it finds. To decide which validated PDR will be processed next, it checks available resources and DAAC configured priorities.
DPL Insert	One per request	*Process:* Sybase Server (COTS)	*Process:* EcDlInProcessingService *Class:* DpInDplRegistrationQAction	The EcDlInProcessingService copies the granule files into the Data Pool SAN, using hidden directories for that purpose unless the DAAC requested that the data be published on insert.
Archive Insert	One per request	*Process:* StorNext copy	*Process:* EcDlInProcessingService *Class:* DpInArchiveQAction	The EcDlInProcessingService copies the granule files into the StoreNext archive file system.

Table 4.2-4. DPL Ingest CSCI Process Interface Events (3 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
IIU Insert	One per request	*Process:* EcDsAmIiu *Library*: iiu.jar	*Process:* EcDlInProcessingService *Class:* DpInInventoryInsertQAction	The EcDlInProcessingService also inserts the granules into the inventory.
Notify with PAN or PDRD	One per email notification request	*Process:* Sendmail (COTS) Ftp daemon (COTS)	*Process*: EcDlInNotificationService *Class*: DpInNotifyEmailAction DpInNotifyFtpAction	The EcDlInNotificationService sends notification to the Data Provider, it could be immediate via PDRD if PDR validation failed, or later on via a short or long PAN.
Create Alert	One per request	*Process*: Sybase Server (COTS)	*Process*: EcDlInPollingService EcDlInProcessingService EcDlInNotificationService *Class*: DpCoAlert	The EcDlInPollingService, EcDlInProcessingService and EcDlInNotificationService create an alert for resource failures and stores the alert in the Sybase Server (INGST Database).
Fetch Config Info	One per startup/ One per configurable interval	*Process*: Sybase Server (COTS)	*Process*: EcDlInPollingService EcDlInProcessingService EcDlInNotificationService *Class*: DpInNotifyDatabase	The EcDlInPollingService, EcDlInProcessingService and EcDlInNotificationService retrieve the configuration information from Sybase Server (INGST Database).
Update DPL Ingest Request	One per request	*Process*: Sybase Server (COTS)	*Process*: EcDlInProcessingService *Class*: DpInProcessingDbInterface	The EcDlInProcessingService updates DPL Ingest request in the Sybase Server (INGST Database).

Table 4.2-4. DPL Ingest CSCI Process Interface Events (4 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Create Operator Intervention	One per request	*Process*: Sybase Server (COTS)	*Process*: EcDlInProcessingService *Class*: DpInProcessingDbInterface	The EcDlInProcessingService creates new Operator Intervention for request failures in the Sybase Server (INGST Database).
Fetch DPL Ingest Request	One per request	*Process*: Sybase Server (COTS)	*Process*: EcDlInProcessingService *Class*: DpInProcessingDbInterface	The EcDlInProcessingService retrieves information associated with a DPL Ingest request from the Sybase Server (INGST Database).
Fetch Actions of Request Changes	One per configurable interval	*Process*: Sybase Server (COTS)	*Process*: EcDlInProcessingService *Class*: DpInProcessingDbInterface	The EcDlInProcessingService retrieves actions regarding request changes, such as, request priority change, cancel request, suspend request, and update request parameters from the Sybase Server (INGST Database).
ODL To XML Conversion	One per ODL metadata	*Process*: OdlToXmlConverter *Library*: odlToXml.jar	*Process*: EcDlInProcessingService *Class*: DpInGranuleScheduler	The EcDlInProcessingService invoke a java utility to convert the ODL metadata file into XML metadata file.
Validate XML Metadata	One per metadata validation	*Process*: EcDsAmXvu *Library*: xmlsvcs	*Process*: EcDlInProcessingService *Class*: DpInXmlValidationQAction	The EcDlInProcessingService populates the metadata files and sends them to the XVU for validation.

Table 4.2-4. DPL Ingest CSCI Process Interface Events (5 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
Request Data Search	One per granule pointer in linkage file	*Process*: Sybase Server (COTS)	*Process*: EcDlInProcessingService *Class*: DpInGranuleScheduler	The EcDlInProcessingService queries Sybase server (Inventory database), for the granule corresponding to a particular ESDT short name and version, which has a particular local granule id.
Fetch Notification Actions	One per configurable interval	*Process*: Sybase Server (COTS)	*Process*: EcDlInNotificaionService *Class*: DpInNotifyDatabase	The EcDlInNotificationService retrieves actions regarding request notifications from the Sybase Server (INGST Database).
Suspend/Resume/Cancel/Alter/Retrieve Request	One per click	*Process*: Sybase Server (COTS)	DPL Ingest GUI	The DPL Ingest GUI scripts send suspend, resume, cancel, alter and retrieve request command to the Sybase Server (INGST Database).
Suspend/Resume Resource	One per click	*Process*: Sybase Server (COTS)	DPL Ingest GUI	The DPL Ingest GUI scripts send, suspend, or resume resource command to the Sybase Server (INGST Database).
Transfer Data	One per science data file activity	*Process*: Ftpd (COTS) or sshd (COTS)	*Process*: EcDlInProcessingService *Class*: DpInInternalFtpTransferQAction/DpInInternalScpTransferQAction	The EcDlInProcessingService transfers data files from the External/Local Disk specified in PDR.
Checksum Data	One per data file activity	*Process*: Checksum utilities (COTS)	*Process*: EcDlInProcessingService *Class*: DpInChecksumQAction	The EcDlInProcessingService checksums data files from the checksum information specified in PDR or calculate the checksum based on the provider and system configuration.

4.2.1.6 Ingest Data Stores

The DPL Ingest CSCI uses the COTS product Sybase to store related DPL Ingest Information. Table 4.2-5 provides descriptions of the data stores.

Table 4.2-5. DPL Ingest CSCI Data Stores

Data Store	Type	Description
INGST Database	Sybase	INGST Database is designed to store the persistent information of user request, processing configuration, request aging configuration, and request cleanup configuration.
DPL Database	Sybase	The Data Pool (DPL) database implements the large majority of the persistent data requirements for the DPL subsystem which supports large online cache of important ECS data at each DAAC and avoids tape access to ECS archive.
Inventory Database	Sybase	The AIM Inventory database is designed to store the minimal metadata information of ingested granules and file storage information of data and metadata files.

4.3 Client Subsystem Overview

The Client Subsystem (CLS) is a set of CSCIs and processes that provide EED end-user services.

These services include allowing users to:

- View HDF formatted files

In addition, the workstations operating within an ECS CLS contains infrastructure support software as part of the CSS and platform support software.

Client Subsystem Context

Table 4.3-1 describes the Client Subsystem Interface events.

Table 4.3-1. Client Subsystem Interface Events

Event	Interface Event Description
Enter HDF File Name	This is a file name for a Hierarchical Data Format (HDF) file. The **user** opens the file to see the data in the file.

Client Subsystem Structure

4.3.1 Tools Description

Table 4.3-2 describes the SSI&T event descriptions.

Table 4.3-2. SSI&T Tool Events

Event	Event Frequency	Interface	Initiated By	Event Description
Hdiff hdiff	cmd line I/F and COTS binary	SSIT	Developed and COTS	Tools to compare binary and HDF files. The shell program EcClHDiff and is used to assist with the viewing and comparisons.

4.4 Data Management Subsystem Overview

The Data Management Subsystem (DMS) provides interoperability between the ECHO (EOS ClearingHouse) and the ECS. The DMS provides this service by supplying a gateway process. The ECHO WSDL (Web Service Definition Language) Order Component (EWOC) allows users to order ECS products via order tools that interface with ECHO. The ECS will no longer support search and browse capabilities as these will be handled internally by ECHO. The EWOC will provide the means for ECHO to present orders to ECS on behalf of the user.

DMS functionality includes:

- DMS validates and places orders that users submit using the clients that interface with ECHO.
- DMS submits orders to External Processors for granules that require external processing.
- DMS allows External Processors to update the status of requests in ECS.

Data Management Subsystem Context

Figure 4.4-1 is the Data Management Subsystem context diagrams. The diagrams show the events sent to the Data Management Subsystem and the events the Data Management Subsystem sends to other external systems and CSMS subsystems. Table 4.4-1 provides descriptions of the interface events shown in the Data Management Subsystem context diagrams.

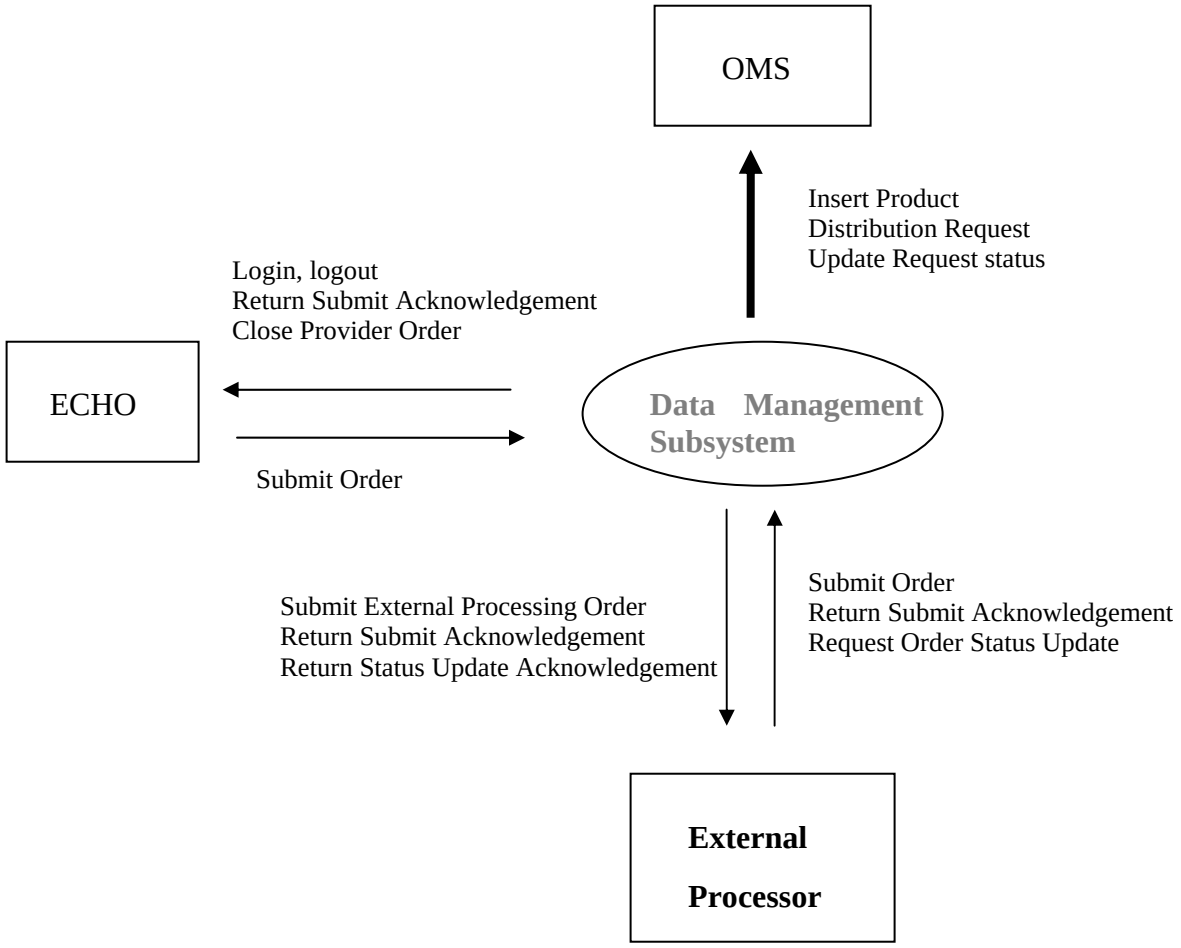

Figure 4.4-1. Data Management Subsystem Context Diagram

Table 4.4-1. Data Management Subsystem Interface Events

Event	Interface Event Description
Insert Product Distribution Request	The Data Management Subsystem (DMS) inserts product distribution requests in the Order Manager Data Base Management System within the **Order Manager Subsystem (OMS)**.
Update Request Status	The DMS submits a request status update to the **OMS**
Submit Order	The DMS receives product requests from the **External Processor*.**
Return Submit Acknowledgement	The DMS receives confirmation of an external processing order from the **External Processor.**
Request Status Update	The DMS receives status update requests from the **External Processor.**
Submit External Processing Order	The DMS submits an external processing request in the **External Processor.**
Return Submit Acknowledgement	The DMS returns a confirmation of an order submitted from **External Processor.**
Return Status Update Acknowledgement	The DMS returns a confirmation of a status update request by the **External Processor.**
Submit Order	The DMS receives product requests from the **ECHO** on behalf of an external ECS user.
Login, logout	The DMS logins and logouts to Authentication Service at **ECHO** to obtain security token.
Return Submit Acknowledgement	The DMS returns a confirmation of order submitted from **ECHO.**
Close Provider Order	The DMS updates the status of requests at **ECHO** when the requests reach terminal states at ECS.

**Note: For the purpose of this document, "External Processor" refers to either an External Subsetter (HAS) or an On-Demand Processor (S4PM), both of which are treated the same by the DMS.*

Data Management Subsystem Structure

The DMS consist of one CSCI:

- The ECHO WSDL Order Component (EWOC) is a software configuration item. The EWOC provides a gateway between ECHO and ECS by allowing users using external client to submit ECS orders through ECHO. The EWOC validates and submits orders to ECS for product distribution. The EWOC also allows interaction with External Processors. External Processors receive external processing orders from the EWOC and submit status update requests to the EWOC.

Use of COTS in the Data Management Subsystem

- Apache Axis 1.4

  The Apache Axis packages are used to generate JAVA web service which uses SOAP messages for communication.

Error Handling and processing

EMD Process Framework package is used for general error reporting. The functions can catch exceptions coming from try blocks and the exception stack trace is logged in the log files. Exceptions that occur during interaction with ECHO will be propagated to the ECHO to indicate order status to the user.

There are three kinds of logs: operations, debugging and performance.

Each conforms to and is supported by the process framework package under /ecs/formal/COMMON/java/gov/nasa/emd/processframework which wraps the Java Logging utility. Each type of log provides for four different levels of output: NONE, INFORMATION, VERBOSE and XVERBOSE.

For writing messages to the log, the following function from LogWrapper class is used:

Public static void log(int level, boolean debug, boolean ops, String message),

For example, the following writes to operations log with output level of INFORMATION.

LogWrapper.log (Logger.INFORMATION, false, true, “EWOC Initialization complete”);

For writing messages to the debug log, the following function calls are used:

LogWrapper.log (Logger.VERBOSE, true, false, “CloseRequestPoller”);

4.4.1 ECHO WSDL Order Component Software Description

4.4.1.1 ECHO WSDL Order Component Functional Overview

The ECHO WSDL Order Component (EWOC) CSCI provides a gateway between ECHO and ECS systems. The users using the client will search, browse and order data using ECHO and submit orders to the EWOC CSCI. The EWOC CSCI then validates the order according to the ECS rules, and submits the requests to the Order Manager Subsystem for product distribution. The EWOC CSCI also updates the status of the request at ECHO to provide the user with an order status.

The EWOC CSCI is a web service. Apache Axis handles service layer and receives messages via SOAP format. Once the order is received, the EWOC returns submit acknowledgement which describes whether order submission was successful. For orders that require external processing, the EWOC places a request at the External Processor and accepts status update requests from the External Processor.

4.4.1.2 ECHO WSDL Order Component Context

Figure 4.4-2 is the EWOC CSCI context diagrams. The diagrams show the events sent to other CSCIs or CSCs and the events the EWOC CSCI receives from other CSCIs and CSCs. Table 4.4-2 provides descriptions of the interface events shown in the EWOC CSCI context diagrams.

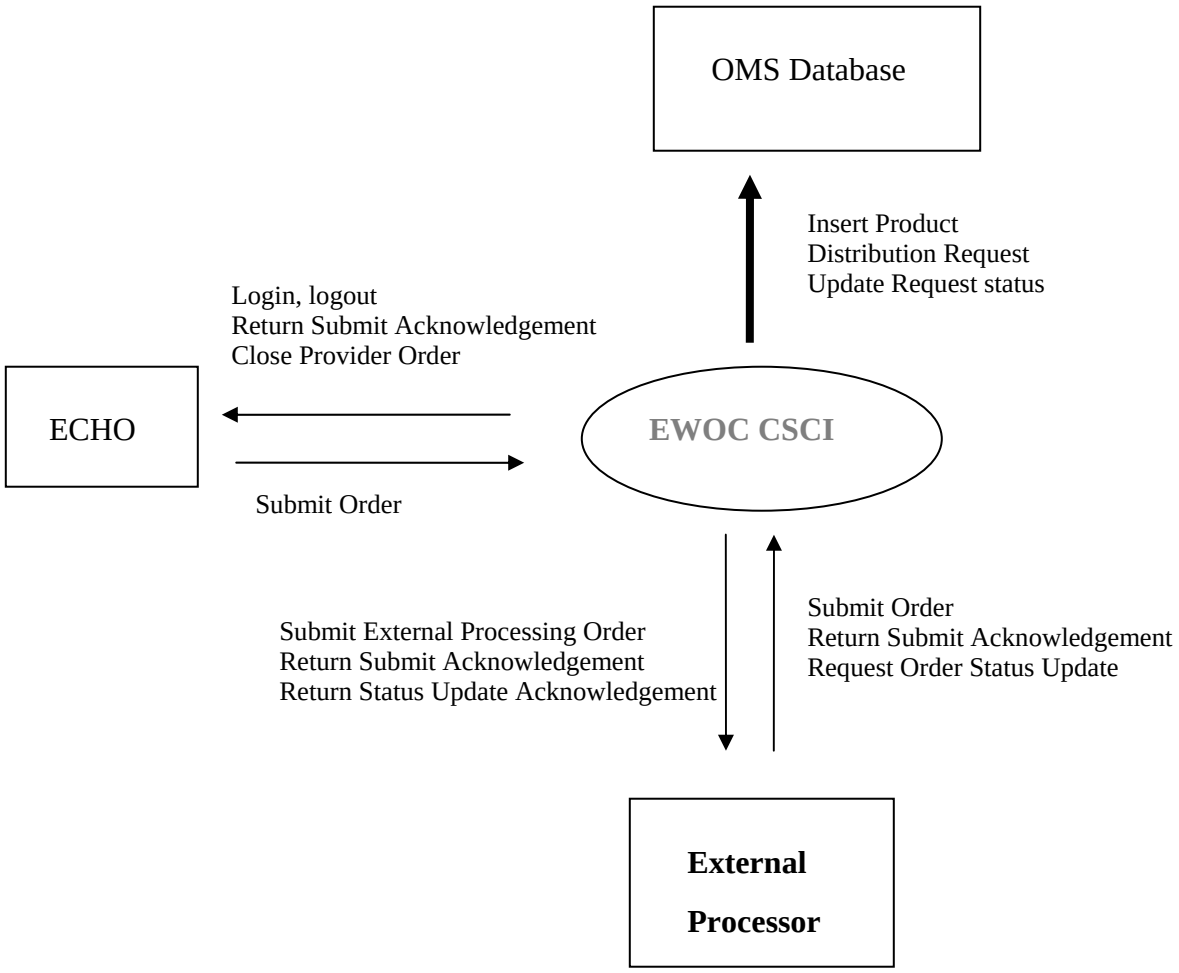

Figure 4.4-2. ECHO WSDL Order Component CSCI Context Diagram

Table 4.4-2. ECHO WSDL Order Component CSCI Interface Events

Event	Interface Event Description
Insert Product Distribution Request	The ECHO WSDL Ordering Component (EWOC) inserts product distribution requests in the Order Manager Data Base Management System within the **Order Manager Subsystem (OMS)**.
Update Request Status	The EWOC submits a request status update to the **OMS.**
Submit Order	The EWOC receives product requests from the **External Processor.**
Return Submit Acknowledgement	The EWOC receives a confirmation of an external processing order from the **External Processor.**
Request Order Status Update	The EWOC receives status update requests from the **External Processor.**
Submit External Processing Order	The EWOC submits an external processing request in the **External Processor.**
Return Submit Acknowledgement	The EWOC returns a confirmation of an order submitted from **External Processor.**
Return Status Update Acknowledgement	The EWOC returns a confirmation of a status update request by the **External Processor.**
Submit Order	The EWOC receives product requests from the **ECHO** on behalf of an external ECS user
Login, logout	The EWOC logins and logouts to Authentication Service at **ECHO** to obtain security token.
Return Submit Acknowledgement	The EWOC returns a confirmation of order submitted from **ECHO.**
Close Provider Order	The EWOC updates the status of requests at **ECHO** when the requests reach terminal states at ECS.

4.4.1.3 ECHO WSDL Order Component Architecture

Figure 4.4-3 is the EWOC CSCI architecture diagram. The diagram shows the events sent to the EWOC CSCI processes and the events the EWOC CSCI process sends to other processes.

The EWOC CSCI is one process, the EcDmEwoc as shown in the ECHO WSDL Order Component CSCI Architecture Diagram.

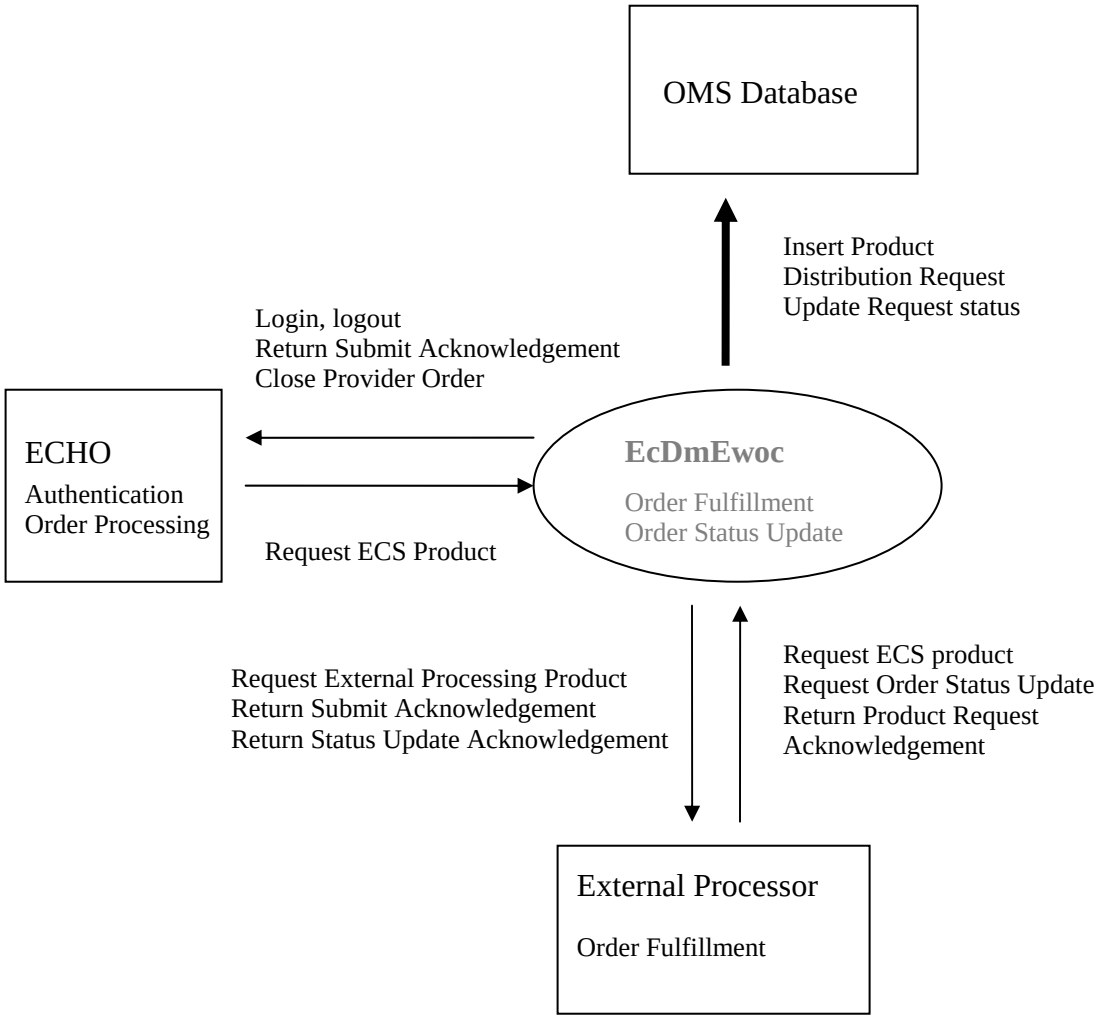

Figure 4.4-3. ECHO WSDL Order Component CSCI Architecture Diagram

4.4.1.4 ECHO WSDL Order Component Process Descriptions

Table 4.4-3 provides descriptions of the processes shown in the EWOC CSCI architecture diagram.

Table 4.4-3. EWOC CSCI Processes

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcDmEwoc	Web Service	DMGHW	Developed	The ECHO WSDL Order Component is a web service that runs on Tomcat/Apache. The EWOC offers two basic interfaces OrderFulfillment Port: External systems such as ECHO or External Subsetter can submit orders to this endpoint. The EWOC validates the order according to ECS rules and then bundles the order into separate requests before submitting the requests to OMS for product distribution. If the order is an external processing order, the EWOC submits an order to the external processor on behalf of ECS. OrderStatusUpdate Port: External processors can submit requests to update the status of an order using this interface.

4.4.1.5 ECHO WSDL Order Component Process Interface Descriptions

Table 4.4-4 provides descriptions of the interface events shown in the EWOC CSCI architecture diagram.

Table 4.4-4. EWOC CSCI Process Interface Events (1 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Insert Product Distribution Request	One per product request	*Process:* Sybase Server (COTS)	*Process:* EcDmEwoc *Classes:* OmsDataAccessImpl	The EcDmEwoc inserts product distribution requests into the Order Manager DBMS by invoking OMS stored procedures.
Update Request Status	One per status update request	*Process:* Sybase Server (COTS)	*Process:* EcDmEwoc	The EcDmEwoc updates the request status in the MSS database by invoking OMS stored procedure.

Table 4.4-4. EWOC CSCI Process Interface Events (2 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Request ECS Product	One per product request	*Process:* OrderFulfillment Port *Classes*: OrderFulfillmentPortBindingImpl	*Process:* External Processor, ECHO	The External Processor or the ECHO submits an order through EWOC's OrderFulfillment Port.
Request Order Status Update	One per status update request	*Process:* OrderStatusUpdate Port *Class:* OrderServiceImpl	*Process:* External Processor	The External Processor submits a request status update through EWOC's OrderStatusUpdate Port.
Return Product Request Acknowledgement	One per status update request	*Process:* OrderFulfillment Port *Classes:* OrderFulfillmentPortBindingImpl	*Process:* External Processor	The Extermal Processor returns SubmitAcknowledgement which contains the information regarding the success of order submission to the External Processor.
Request Subsetted Product	One per product request	*Process:* OrderFulfillment Port	*Process:* EcDmEwoc *Class:* EPDataAcessImpl	The EcDmEwoc places a request at External Processor for the granule to be subsetted.
Return Submit Acknowledgement	One per product request	*Process:* OrderFulfillment Port *Class:* OrderFulfillmentProtBindingImpl	*Process:* EcDmEwoc *Class:* OrderFulfillmentPortBindingImpl	The EWOC returns SubmitAcknowledgement which contains the information regarding the success of order submission from the External Processor.
Return Status Update Acknowledgement	One per status update request	*Process:* OrderStatusUpdate Port *Class:* OrderServiceImpl	*Process:* EcDmEwoc *Class:* OrderFulfillmentPortBindingImpl	The EWOC returns UpdateStatus with information regarding the success of request status update

Table 4.4-4. EWOC CSCI Process Interface Events (3 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Login and logout	One every polling cycle	*Process:* AuthenticationService *Class:* AuthenticationServicePortImpl	*Process:* EcDmEwoc *Class:* CloseRequestHandlerImpl	The EWOC tries to login to ECHO's Authentication Service and receive a security token.
Return Submit Acknowledgement	One per product request	*Process:* OrderFulfillment Port *Class:* OrderFulfillmentProtBindingImpl	*Process:* EcDmEwoc *Class:* OrderFulfillmentPortBindingImpl	The EWOC returns SubmitAcknowledgement which contains the information regarding the success of order submission from the ECHO.
Close Provider Order	One every polling cycle	*Process:* OrderProcessingService *Class:* OrderProcessingServicePortImpl	*Process:* EcDmEwoc *Class:* CloseRequestHandlerImpl	The EWOC tries to update the status of order at ECHO for orders that are in terminal states.

4.4.1.6 ECHO WSDL Order Component CSCI Data Stores

The EWOC CSCI calls OMS stored procedures to access OMS DB. The EWOC will have an OMS database interface only; it will not have an interface with the MSS database. The creation of MSS order and request objects will be handled through OMS stored procedure calls. Table 4.4-5 provides descriptions of the data stores shown in the EWOC CSCI architecture diagram.

Table 4.4-5. ECHO WSDL Order Component CSCI Data Stores

Data Store	Type	Functionality
OMS DB	Database	OMS Database is designed to store the persistent information of user request, processing mode configuration, media configuration, staging policy configuration, Ftp Push policy configuration, request aging configuration, and request cleanup configuration.

4.4.2 Data Management Subsystem Hardware

The primary components of the Data Management Subsystem include one hardware CI, Data Management Hardware CI (DMGHW). The general-purpose workstations are standalone hosts without fail-over capability. In the event of a host failure, any of the available workstations could be used to support end user DAAC maintenance.

4.4.2.1 Data Management Hardware CI (DMGHW) Description

The DMGHW CI includes Linux workstations. In the EBIS Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping. These workstations are used as end user workstations in maintenance of each of the respective DAAC sites.

4.5 Order Manager Subsystem Overview

The Order Manager subsystem (OMS) manages all orders placed through ECHO via WIST, the Spatial Subscription Server, the EPD Server (i.e., external subsetter request and S4PM) and the Data Pool Web GUI (including HEG orders).

Once a request comes into the OMS subsystem, the server validates the request. Upon successful validation, the server stages the order in the Data Pool storage area. For Ftp Pull requests, links are created from the FtpPull directory to the staged files in the Data Pool storage. For Ftp Push requests, the OMS Ftp Push driver directly distributes the data. Upon successful shipment, OMS sends a Distribution Notice to the end user. An order is considered shipped as soon as the request status is updated to “Shipped”. For an FtpPull order, the request status is updated to “Shipped” after the order is staged and file links are made in the Data Pool storage; for an FtpPush order, request status is “Shipped” after the Order Manager Server finishes pushing all the data associated with the order to its destination.

Special orders such as HEG and External Subsetter orders require further processing by the HEG Server or the External Subsetter. For HEG orders, the Order Manager creates HEG requests per granule based on the processing instructions in the original HEG order. The HEG requests are submitted to the HEG Server through the HEG API. The HEG server processes the HEG requests and returns the final output to the Order Manager Server which then distributes the final output to the end user. For External Subsetter Orders, the External Subsetter creates output granules which are then associated with the Order by the EPD Server. These output granules are later distributed by the Order Manager Server. The Order Manager Subsystem also includes a database that stores all order information persistently as soon as an order is received by ECS and before its receipt is acknowledged. This allows operators to resubmit an order if it encounters errors downstream, and allows the Order Management Service to perform some up front checks on the order and generate an operator intervention for the operator to handle the error conditions.

Order Manager Subsystem Context

Figure 4.5-1 is the Order Manager Subsystem context diagram. The diagram shows the events sent to the Order Manager Subsystem and the events the Order Manager Subsystem sends to other subsystems. Table 4.5-1 provides descriptions of the interface events shown in the Order Manager Subsystem context diagram.

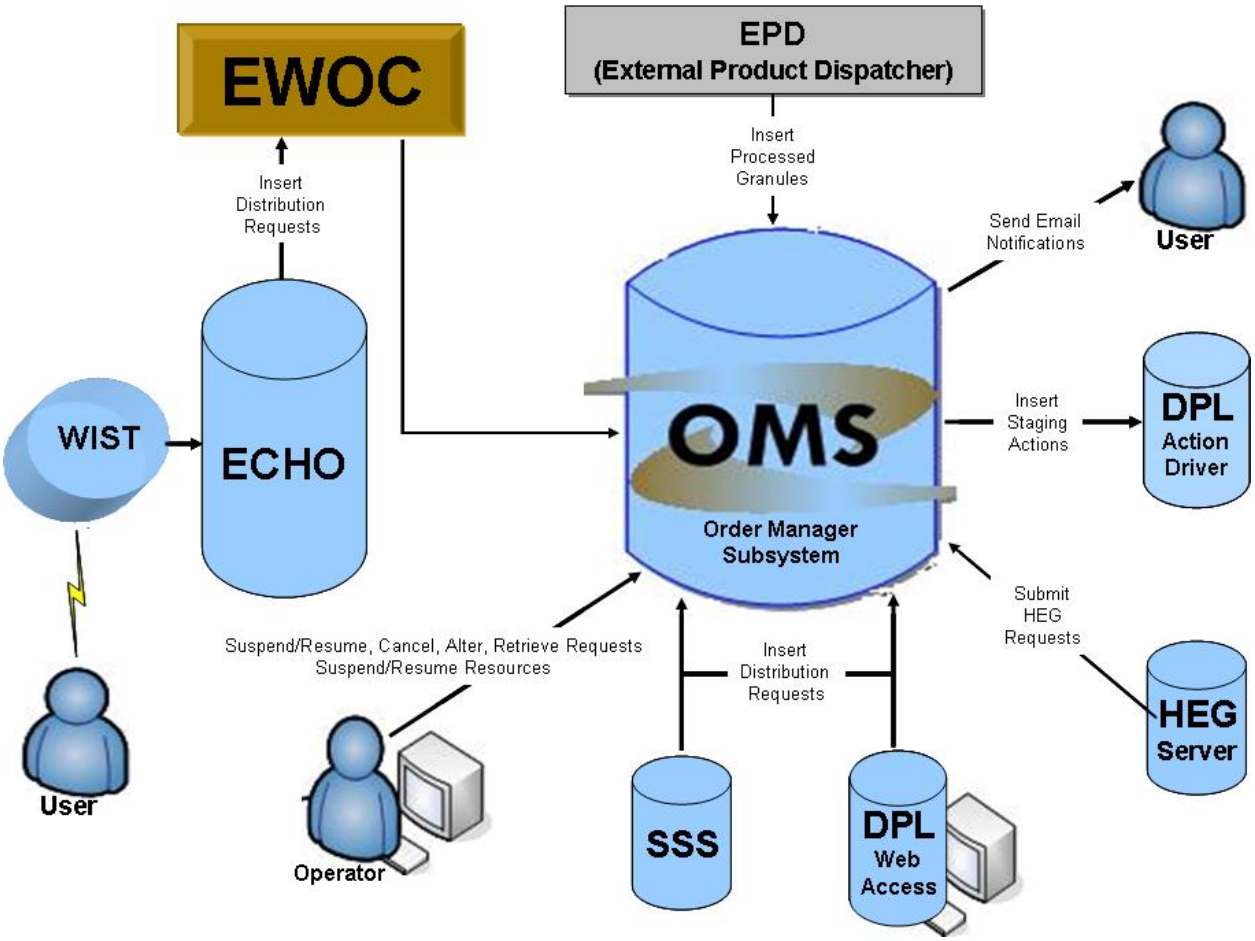

Figure 4.5-1. Order Manager Subsystem Context Diagram

Table 4.5-1. Order Manager Subsystem Interface Events (1 of 2)

Event	Interface Event Description
Insert Distribution Requests	The OMS receives Distribution Requests from the **Spatial Subscription Server** (SSS), the EPD Server (External Product Dispatcher), and the **DPL Subsystem** (Web GUI).
Insert Staging Actions	The OMS submits DPL insert actions to the **DPL Subsystem**.
Suspend/Resume, Cancel, Alter and Retrieve Requests	The **Operator** suspends, resumes, cancels, alters and retrieves requests from the OMS (OMS DB).
Suspend/Resume Resources	The Operator suspends and resumes dispatching to all or selected resources.

Table 4.5-1. Order Manager Subsystem Interface Events (2 of 2)

Event	Interface Event Description
Send Email Notification	The OMS sends email notification to the requesting **user** when a request is altered, canceled or shipped or when a request enters operator intervention, or when an alert or intervention is generated. The OMS sends email notification to users of the **DPL Web Access GUI** when OMS receives requests.
Submit HEG Request	The OMS submits HEG requests to the HEG Server.

Order Manager Subsystem Structure

- The Order Manager Subsystem consists of two CSCIs: the OMSRV and the OM GUI (described in the 609 document). The Order Manager Server (EcOmOrderManager) is a software configuration item. The Order Manager Server receives product distribution requests and submits them to the appropriate ECScomponent based upon the media type specified for the request. The OMS server stages granules associated with a request in the DPL storage area and then distributes the data via electronic media (i.e., Ftp Push, Ftp Pull, SCP. For special orders such as HEG Orders, the Order Manager creates HEG requests based on processing instructions and submits the HEG requests to the HEG Server. The output of a HEG request is distributed to the end user. Similar to HEG Orders, output granules associated with an External Subsetter request are distributed to the end user after being processed by the external subsetter. Order Manager Subsystem information is stored persistently in a relational Database Management System (DBMS). The Order Manager GUI (OMGUI) is used to monitor and control the operations of the Order Manager Server. In addition, the OMGUI is used to respond to Operator Intervention Requests generated by the Order Manager Server.

Use of COTS in the Order Manager Subsystem

- RogueWave's Tools.h++

  The Tools.h++ class libraries are used by the OMS to provide basic functions and objects such as strings and collections. These libraries must be installed with the OMS software for any of the OMS processes to run.

- Sybase Open Client / CT_LIB

  The Sybase Open Client provides access between OMS custom code and the Sybase SQL Server DBMS.

- Sybase Server

  The Sybase SQL server provides access for OMS to insert, update and delete Distribution Requests, OMS configurations, and Operator Interventions. The Sybase SQL Server must be running during operations for the OMS to process Product Distribution Requests.

4.5.1 Order Manager Subsystem Software Description

4.5.1.1 Order Manager Server CSCI Functional Overview

The Order Manager Server (OMSRV) CSCI executes as a process and interacts with the Order Manager Database and the Data Pool System (DPL) Database. The Spatial Subscription Server (SSS), EWOC, HEG Server, EPD Server and the Data Pool Web GUI submit distribution requests to the OMS. These requests are all validated. Upon successful validation, the server stages the granules in a request in the DPL storage area. Electronic Ftp push media requests are directly pushed to the end user. Note that Order Manager Server dispatches HEG requests to the HEG Server for processing before being distributed to the end user. For invalid requests, an Operator Intervention is generated. DAAC OPS personnel can use the Order Manager GUI to correct and resubmit the request. In response to an intervention, the Operator can also generate an email message, which is sent to the user by the Order Manager Server. The Order Manager Server also generates an alert and sends an email to a pre-configured email address when it detects internal or external resource failure. While a resource is suspended, the OMS Server halts dispatching of the requests that are utilizing the suspended resource.

4.5.1.2 Order Manager Server CSCI Context

Figure 4.5-2 is the Order Manager Server CSCI context diagram. The diagram shows the events sent to the Order Manager Server CSCI and the events the Order Manager Server CSCI sends to other CSCIs. Table 4.5-2 provides descriptions of the interface events shown in the Order Manager Server CSCI context diagrams.

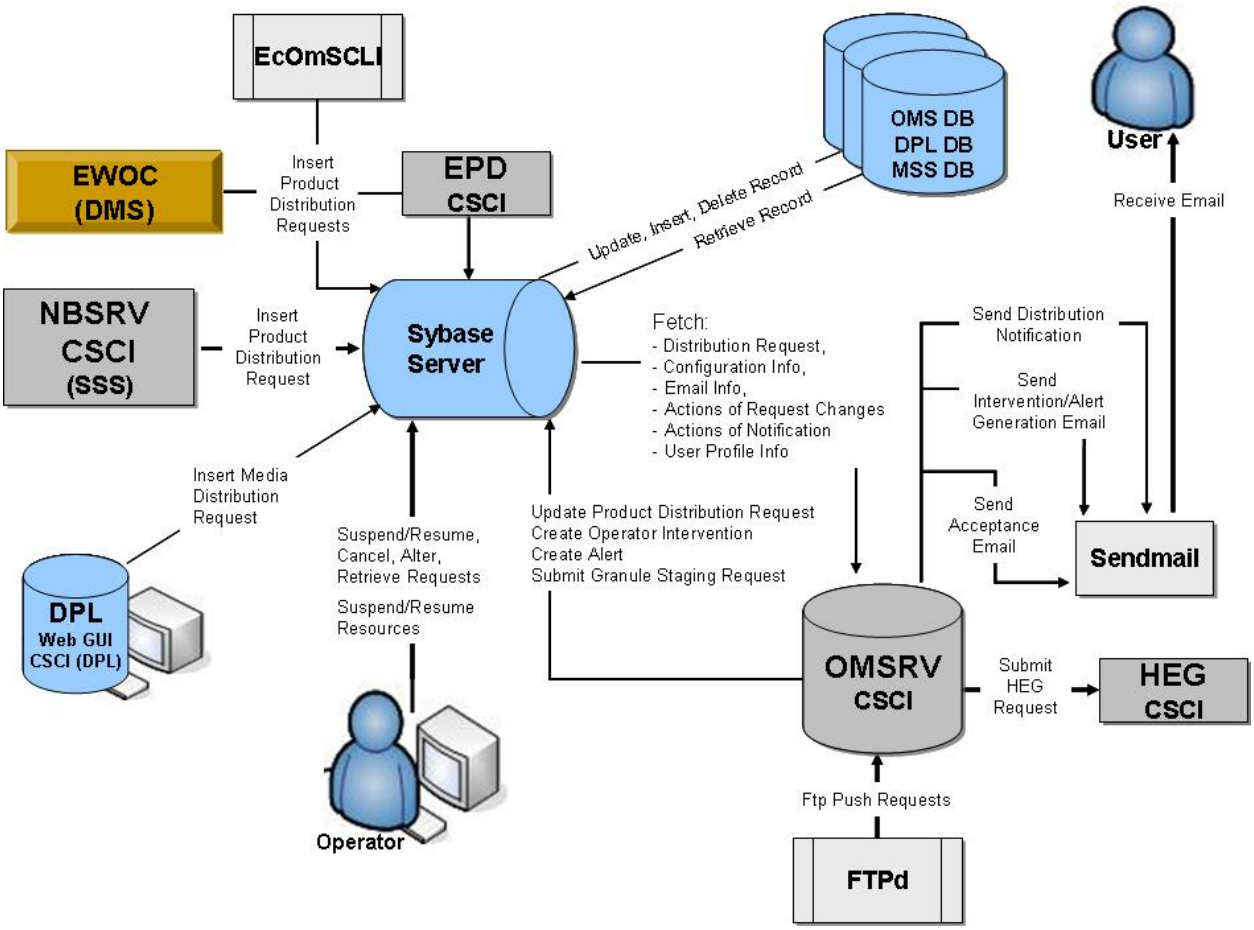

Figure 4.5-2. Order Manager Server CSCI Context Diagram

Table 4.5-2. Order Manager Server CSCI Interface Events (1 of 2)

Event	Interface Event Description
Send Distribution Notification	The OMSRV CSCI sends distribution notifications to the **end-users**.
Send Intervention/Alert Generation Email	The OMSRV CSCI sends intervention/alert generation email to the **end-users**.
Send Acceptance Email	The OMSRV CSCI sends a request acceptance email to the **DPL Web GUI** users upon receiving the request in the OMS DB.
Receive Email	**User** receives a status email sent by the OMSRV CSCI when request is interventioned, shipped or failed.

Table 4.5-2. Order Manager Server CSCI Interface Events (2 of 2)

Event	Interface Event Description
Insert Product Distribution Request	The NBSRV CSCI, SCLI CSCI, and DMS CSCI insert product distribution requests into the **Sybase Server** (**OMS DB**) to be queued for processing by the OMSRV CSCI.
Insert Media Distribution Request	The DPL Web GUI inserts media distribution request into the **Sybase Server** (**OMS DB**).
Update Product Distribution Request	The OMSRV CSCI updates product distribution requests in the **Sybase Server** (**OMS DB**).
Create Operator Intervention	The OMSRV CSCI creates new Operator Intervention for request failures in the **Sybase Server (OMS database)**.
Create Alert	The OMSRV CSCI creates an alert for resource failures and stores the alert in the **Sybase Server (OMS database)**.
Submit granule staging request	The OMSRV CSCI submits a request to stage a granule to the **Sybase Server** (DPL storage in the OMS DB), which in turn calls DPL stored procedures to insert an action into the DPL DB.
Fetch Distribution Request	The OMSRV CSCI retrieves information associated with a product distribution request from the **Sybase Server** (OM Database).
Fetch Configuration Info	The OMSRV CSCI retrieves the OMSRV Configuration information from the **Sybase Server**.
Fetch Email Info	The OMSRV CSCI retrieves information related to an operator intervention required to generate an email notification from the **Sybase Server**.
Fetch actions of Request changes	The OMSRV CSCI retrieves actions regarding request changes, such as, request priority change, cancel request, suspend request, and update request ftppush parameters from the **Sybase Server**.
Fetch actions of notification	The OMSRV CSCI retrieves actions regarding granule staged and DPL file system modified notification from the **Sybase Server**.
FtpPush Requests	The OMSRV CSCI Ftp Pushes a request to the **end-user**.
Submit HEG Request	The OMSRV CSCI submits HEG requests to the HEG Services for processing.
Suspend/Resume, Cancel, Alter and Retrieve Request	The Operator suspends, resumes, cancels, alters and retrieves requests from the **Sybase Server** (**OMS DB**).
Suspend/Resume Resources	The OMSRV CSCI suspends or resumes dispatching to all or selected resources in the **Sybase Server**.
Update, Insert, Delete Record	The **Sybase Server** performs update, insert, and delete database operations to the OMS DB and DPL DB.
Retrieve Record	The **Sybase Server** performs retrieval database operations to/from the OMS DB and DPL DB.

4.5.1.3 Order Manager Server CSCI Architecture

Figure 4.5-3 is the Order Manager Server (OMSRV) CSCI architecture diagram. The diagram shows the events sent to the OMSRV CSCI processes and the events the OMSRV CSCI processes send to other processes.

The OM Server CSCI consists of one process. This process is the EcOmOrderManager process.

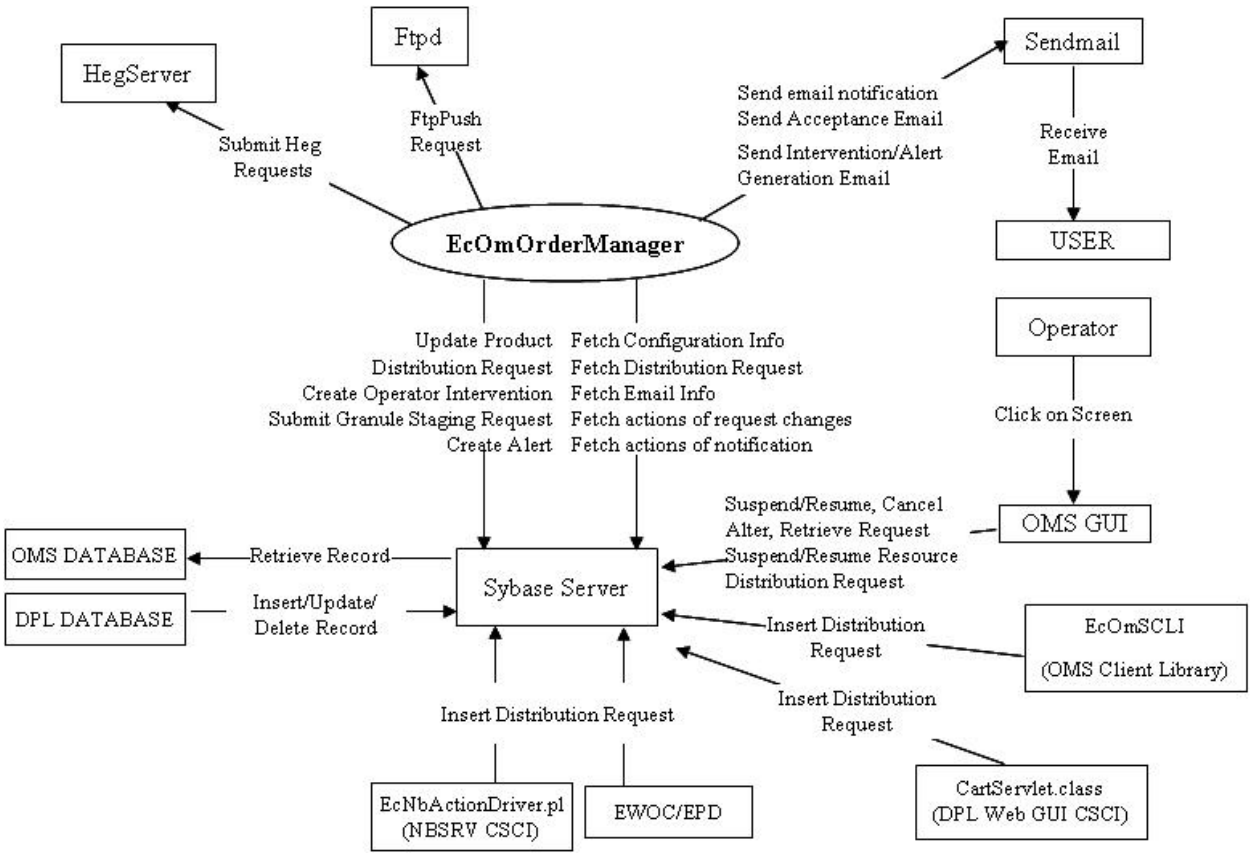

Figure 4.5-3. Order Manager Server CSCI Architecture Diagram

4.5.1.4 Order Manager Server CSCI Process Description

Table 4.5-3 provides descriptions of the processes shown in the OMSRV CSCI architecture diagram.

Table 4.5-3. OMSRV CSCI Process

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcOmOrderManager	Server	OMSHW	Developed	The Order Manager Server prepares the request data in the DPL storage for distribution. FTP Push requests are directly pushed to the end user. Note that HEG and external subsetter requests are first dispatched to the HEG Server or subsetter for processing. The processed output is then distributed to the end user. Order Manager Server sends a Distribution Notification to the end-user on completing an order.

EMD Baseline Information System (EBIS) Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.5.1.5 Order Manager Server CSCI Interface Description

Table 4.5-4 provides descriptions of the interface events shown in the Order Manager Server (OMSRV) CSCI architecture diagram.

Table 4.5-4. Order Manager Server CSCI Process Interface Events (1 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Insert Product Distribution Request	One per service request	*Process:* Sybase Server (COTS)	*Process:* EWOC, EPD *Script:* EcNbActionDriver.pl	EWOC, EPD and EcNbActionDriver.pl (SSS) insert product distribution request into OMS DB.
Submit Heg Request	One or more per Service Request	*Process:* *HgServer*	*Process:* EcOmOrderManager *Class:* OmSrHegProcessingAction	The EcOmOrderManager submits HEG requests to the HEG Server.
Insert Media Distribution Request	One per service request	*Process:* Sybase Server (COTS)	*Process:* CartServlet.class	The CartServlet.class (DPL Web GUI CSCI) inserts media distribution request. Media distribution request could be HEG request.
Send acceptance email	One per email notification request	*Process:* Sendmail (COTS)	*Process:* EcOmOrderManager *Class:*	The EcOmOrderManager sends email notification to the DPL Web GUI end-users upon receipt of the request.
Send Intervention/ Alert Generation email	One per email notification request	*Process:* Sendmail (COTS)	*Process:* EcOmOrderManager *Class:*	The EcOmOrderManager sends intervention/ alert generation email to a configured user email account.
Receive Email	One per email notification	*End User:* specified in request	*Process:* Sendmail (COTS)	The User receives email sent by the EcOmOrderManager.
Send email notification	One per email notification request	*Process:* Sendmail (COTS)	*Process:* EcOmOrderManager *Class:* OmSrEmailRequest	The EcOmOrderManager sends email notifications to the end-users.

Table 4.5-4. Order Manager Server CSCI Process Interface Events (2 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Fetch configuration info	One per startup	*Process:* Sybase Server (COTS)	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Class:* OmSrDbInterface	The EcOmOrderManager retrieves configuration information from the Sybase Server (OMS database).
Fetch Distribution Request	One per configurable interval	*Process:* Sybase Server (COTS)	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Classes:* OmSrDbInterface, OmSrDistributionRequest	The EcOmOrderManager retrieves information associated with a product distribution request from the database.
Fetch actions of request changes	One per configurable interval	*Process:* Sybase Server (COTS)	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Classes:* OmSrDbInterface, OmSrDistributionRequest	The EcOmOrderManager retrieves actions regarding request changes, such as, request priority change, cancel request, suspend request, and update request Ftp Push parameters.
Fetch actions of notification	One per configurable interval	*Process:* Sybase Server (COTS)	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Classes:* OmSrDbInterface, OmSrDistributionRequest	The EcOmOrderManager retrieves actions regarding granule staged, and DPL file system modified notification.
Fetch email info	One per configurable interval	*Process:* Sybase Server (COTS)	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Classes:* OmSrDbInterface, OmSrDistributionRequest, OmSrEmailRequest	The EcOmOrderManager retrieves information related to an operator intervention required to generate an email notification from the Sybase Server (OM DB).

Table 4.5-4. Order Manager Server CSCI Process Interface Events (3 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Update Product Distribution Request	One per request	*Process:* Sybase Server (COTS)	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Classes:* OmSrDbInterface, OmSrDistributionRequest	The EcOmOrderManager updates existing product distribution requests in the Sybase Server (OMS DB).
Create Operator Intervention	One per request	*Process:* Sybase Server (COTS)	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Class:* OmSrDbInterface	The EcOmOrderManager creates a new Operator Intervention request in the Sybase Server (OM DB).
Create Alert	One per resource	*Process:* Sybase Server (COTS)	*Process:* EcOmOrderManager *Library:* Sybase Ct-library *Class:* OmSrDbInterface	The EcOmOrderManager creates an alert related to a resource failure such as Ftp Push Destination, Archive, and DPL File System failure to store in the Sybase Server.
Insert Product Distribution Request	One per service request	*Process:* Sybase Server (COTS)	*Process:* EWOC EPD EcOmCLI *Script:* EcNbActionDriver.pl *Library:* OmClientlib *Classes:* OmSrDbInterface	EWOC, EPDm the EcNbActionDriver, the EcOmSCLI, and the DPL Web Access GUI insert product distribution requests into the Sybase Server (Order Manager DB).
Insert/Update/ Delete/Retrieve Record	One per request	*Database:* OMS DB, DPL DB (COTS)	*Process:* Sybase Server (COTS)	The Sybase server performs database operations (inserts, updates, deletions and retrievals) to the OMS DB and DPL DB
Ftppush request	One per file	*Process:* Ftpd (COTS)	*Process:* EcOmOrderManager	The EcOmOrderManager Ftp Pushes request data to the end-user.
Click on Screen	One per click	*Scripts:* OMS GUI scripts.	Operator	The Operator clicks on the screen to select the action.

Table 4.5-4. Order Manager Server CSCI Process Interface Events (4 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
Suspend/ Resume, Cancel, Alter and Retrieve Requests	One per click	*Process:* Sybase Server (COTS)	*Script:* OMS GUI script	The OMS GUI scripts send suspend/resume, cancel, alter and retrieve request commands to the Sybase Server.
Suspend/ Resume Resource	One per click	*Process:* Sybase Server (COTS)	*Script:* OMS GUI script	The OMS GUI scripts send suspend/resume resource commands to the Sybase Server.

4.5.1.6 Data Stores

There are data stores associated with the Order Manager Server. They are the OMS DB, DPL DB. Table 4.5-5 provides a description of these data stores.

Table 4.5-5. CSCI Data Stores

Data Store	Type	Description
OMS DB	Sybase	OMS Database is designed to store the persistent information of user request, processing mode configuration, media configuration, staging policy configuration, Ftp Push policy configuration, request aging configuration, and request cleanup configuration.
DPL DB	Sybase	The Data Pool (DPL) database implements the large majority of the persistent data requirements for the DPL subsystem which supports Large online cache of the majority of ECS data at each DAAC and avoids tape access to ECS archive.

4.6 Communications Subsystem Overview

The Communications Subsystem (CSS) provides the capability to:

- Transfer information internal to the Earth Observing System Data and Information System (EOSDIS) Evolution and Development (EED)
- Transfer information between the EED sites
- Provide connections between the ECS users and service providers
- Manage the ECS communications functions

Communications Subsystem Context Diagram

Table 4.6-1 provides descriptions of the interface events shown in the CSS context diagrams. **NOTE**: In Table 4.6-1 Request Communications Support is shown as a single event to simplify the table and provide a list of services available from CSS to the other CSMS subsystems.

Table 4.6-1. Communications Subsystem (CSS) Interface Events

Event	Interface Event Description
Request Communications Support (RCS)	The CSS provides a library of services available to **each CSMS** subsystem. The subsystem services required to perform specific assignments are requested from the CSS. These services include: • CCS Middleware Support • Database Connection Services • Fault Handling Services • Mode Information
Export Location Information	The **OMS**, **DMS** send physical and logical server location information to the CSS for data location.

Communications Subsystem Structure

Note: The CSS logical names used in this document do not exactly match the physical names in the directory structure where the software is maintained. Therefore, after the logical name of each Computer Software Component (CSC) in parentheses, there is a physical directory structure name where the software is found. For example, the DCCI CSCI software can be found under the directory structure Distributed Object Framework (DOF) and the Server Request Framework software can be found under the directory structure /ecs/formal/common/CSCI_SRF.

The CSS is composed of one CSCI, the Distributed Computing Configuration Item (DCCI, the software is found in directory DOF) and one HWCI. The CSS software is used to provide communication functions, processing capability, and storage.

Use of COTS in the Communications Subsystem

Note: The following RogueWave Libraries are currently delivered with custom code as static libraries. A separate installation of dynamic libraries is no longer required.

- RogueWave's Tools.h++

  The Tools.h++ class libraries provide basic functions and objects such as strings and collections.

- RogueWave's DBTools.h++

  The DBTools.h++ C++ class libraries provide interaction, in an object-oriented manner, to the Sybase ASE database SQL server. The DBTools provide a buffer between the CSS processes and the relational database used.

- RogueWave's Net.h++

  The Net.h++ C++ class libraries, which provide functions and templates that facilitate writing applications, which communicate with other applications.

Other COTS products include:

- Sybase Adaptive Server Enterprise (ASE)

  The Sybase ASE provides access for the Subscription Server to insert, update and delete Subscription Server database information. The Sybase ASE must be running during CSS operations for the Subscription Server to execute database requests.

  In addition, the Configuration Registry stores configuration values for ECS applications in the Sybase database. The Configuration Registry Server retrieves the values from the database via a Sybase ASE.

- CCS Middleware

  CSS middleware is no longer actively used by the servers but is built in to the binaries. Instead, the servers use direct host/port sockets to communicate with other servers.

- UNIX Network Services

  UNIX Network Services contain DNS, NFS, E-mail service, FTP, and TCP/IP capabilities.

Error Handling and processing

EcUtStatus is a class used throughout the ECS custom code for general error reporting. It is almost always used as a return value for functions and allows detailed error codes to be passed back up function stacks.

When an error occurs, the error is logged into the applications log (ALOG). The Communications Subsystem (CSS) and System Management Subsystem (MSS) have two main mechanisms to handle the error:

1. Return an error status
2. Throw an exception.

The CSS uses the following classes for error handling and processing:

The EcUtStatus class is used to describe the operational status of many functions. The values most often reported are "failed" and "ok." But depending upon the application, detailed values could be set and sent. Please refer to the definition of this class (located in /ecs/formal/COMMON/CSCI_Util/src/Logging/EcUtStatus.h) for all possible values.

The RWCString is used to store some status value returned by applications.

Integer is used to return some error status by applications. This is used specifically between client and server communications.

4.6.1 The Distributed Computing Configuration Item Software Description

The DCCI CSCI (the software is found in directory DOF) consists mainly of COTS software and hardware providing servers, gateways, and software library services to other CSMS CSCIs. The CSCI is composed of 17 computer software components (CSCs) briefly described here followed by a description of the HWCI.

The CSCI is composed of 17 computer software components (CSCs) briefly described here as processes followed by a description of the HWCI.

The remote file access group provides the capability to transfer and manage files using the following five functions: FTP, FTP Notification, Bulk Data Server (BDS), Network File System (NFS), and Filecopy.

1. The NFS (no physical directory) provides a distributed file sharing system among computers. NFS consists of a number of components, including a mounting protocol and server, a file locking protocol and server, and daemons that coordinate basic file service. A server exports (or shares) a filesystem when it makes the filesystem available for use by other machines in the network. An NFS client must explicitly mount a filesystem before using it.
2. The Filecopy utility (the software is found in directory /ecs/formal/common/CSCI_Util/src/CopyProg) copies files from a specified source location to a specified destination location with options available for data compression. The DPL CSCI uses the Filecopy utility to transfer large files.
3. Virtual Terminal (no physical directory) provides the capability for the Operations staff on an ECS platform to remotely log onto another ECS machine.
4. Cryptographic Management Interface (CMI, the software is found in directory /ecs/formal/CSS/DOF/src/AUTHN) provides processes a means for obtaining random passwords and gaining access to Sybase.

5. The Domain Name Service (DNS, the software is found in directory ecs/formal/CSS/NameServer/src) provides host names and addresses to a specified network by querying and answering queries. DNS provides naming services between the hosts on the local administrative domain and also across domain boundaries. DNS is distributed among a set of servers (name servers); each of which implements DNS by running a daemon called in.named. On the client side, the service is provided through the resolver, which is not a daemon. The resolver resolves user queries by needing the address of at least one name server (provided in a configuration file parameter). Each domain must have at least two kinds of DNS servers (a primary and secondary server) maintaining the data corresponding to the domain. The primary server obtains the master copy of the data from disk when it starts up the in.named. The primary server delegates authority to other servers in or outside of the domain. The secondary server maintains a copy of the data for the domain. When the secondary server starts in.named, the server requests all data for the given domain from the primary server. The secondary server checks periodically with the primary server for updates. DNS namespace has a hierarchical organization consisting of nested domains like directories. The DNS namespace consists of a tree of domains. See Figure 4.6-5 for an illustration of the domain tree hierarchy.

6. The Infrastructure Library provides a set of services including the following.

   - Process Framework (PF) (the software is located in directory /ecs/formal/CSS/DOF/src/PF/pf): The PF is a software library of services, which provides a flexible mechanism (encapsulation) for the ECS Client and Server applications to transparently include specific ECS infrastructure features from the library of services. (Library services include: process configuration and initialization, mode management and event handling, life cycle services (server start-up and shut-down), and set-up of security parameters.) The PF process is the encapsulation of an object with ECS infrastructure features and therefore the encapsulated object is fully equipped with the attributes needed to perform the activities assigned to it. The PF was developed for the ECS custom developed applications and is not meant for use by any COTS software applications. The PF ensures design and implementation consistency between the ECS Client and Server applications through encapsulation of the implementation details of the ECS infrastructure services. Encapsulation therefore removes, for example, the task of each programmer repeatedly writing common initialization code. The PF is built by first developing a process classification for the EED project from the client/server perspective. Then the required capabilities are allocated for each respective process level and type. PF-based ECS applications use Process Framework to read in their configuration information at startup. PF-based servers use Process Framework to initialize themselves as a CCS Middleware server and put it in a listen state to begin to accept requests from appropriate clients.

   - Universal References (the software is found in directory /ecs/formal/ COMMON/CSCI_UR/src/UR/framework): Universal References (URs) provide applications and users a system wide mechanism for referencing ECS data and service objects. Manipulating logical entities represented at run time as C++ objects

in virtual memory performs ECS functions. Users and applications require references to the logical entities beyond the effective computational time to keep the objects in memory. Therefore, applications and users are given URs to these objects. Once an UR is made for an object, the object can be disposed of and later reconstituted from the UR. URs take up a small fraction of the space to keep in memory and can be externalized into an ASCII string, which an end user can manage. URs have the capability of re-accessing and/or reconstituting the object into memory as needed. Therefore, the object does not have to remain in memory, and can if appropriate, be written to a secondary storage system, like a database. While the UR mechanism guarantees reliable data externalization and internalization, the content of each type of UR is application specific. Only the object (this is referred to as the "UR Provider") that initially provides the UR is allowed to access and understand its content. URs are strongly typed to enforce appropriate access control to internal data both at compile time and during run time. Since URs are typed and have object specific data in them, separate UR object classes exist for each UR Provider class referred to. All of these UR classes use the mechanisms provided by the UR framework.

- Event Logging (the software is found in directoryLOGGING): Event logging is the capability of recording events into files and provides a convenient way to generate and report detailed events. All ECS CSCIs use event and error logging as an audit trail for all transactions that occur during the ECS data processing and distributing.
- CSS software is executed on multiple hardware hosts throughout the ECS system to provide communication functions, processing capability, and storage. The software and hardware relationships are discussed in the CSS Hardware CI description.

4.6.1.1 CCS Middleware Support Group Description

CSS middleware is no longer actively used by the servers but is built in to the binaries. Instead, the servers use direct host/port sockets to communicate with other servers

4.6.1.2 Virtual Terminal Description

4.6.1.2.1 Virtual Terminal Functional Overview

The Virtual Terminal (VT) effectively hides the terminal characteristics and data handling conventions from both the server host and Operations staff, and enables the Operations staff to remotely log on to other ECS machines. The CSS provides the secure shell server (sshd2) on available systems and common capability support for the ECS remote terminal service.

4.6.1.2.2 Virtual Terminal Context

The CSS provides the secure shell (sshd2) remote access to the ECS systems. SSH is distributed as a third party remote server access service. The SSH service provides users with access to the ECS character-based user interface (CHUI) search and order tool. Figure 4.6-1 is the Virtual

Terminal context diagram and Table 4.6-2 provides the descriptions of the interface events shown in the Virtual Terminal context diagram.

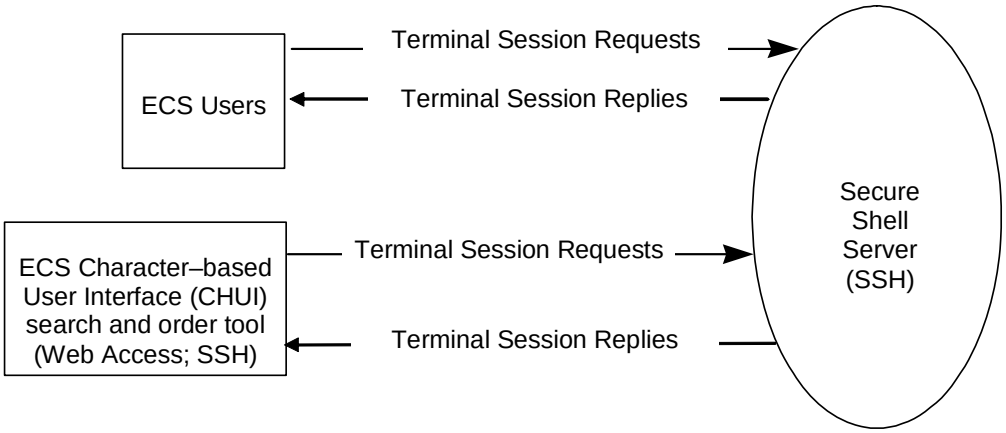

Figure 4.6-1. Virtual Terminal Context Diagram

Table 4.6-2. Virtual Terminal Interface Events

Event	Interface Event Description
Terminal Session Requests (Web access)	**ECS users** request a connection to a specified host via SSH.
Terminal Session Requests (ECS Users)	**ECS users** request a telnet session with a specified ECS host.
Terminal Session Replies (from SSH to ECS or other remote users)	The SSH Server residing on the ECS host responds to the terminal session requests and interacts via the successful connection.

4.6.1.2.3 Virtual Terminal Architecture

Figure 4.6-2 is the Virtual Terminal architecture diagram. The diagram shows the event traffic between the Remote Terminal Session with ECS Users and SSH with remote users.

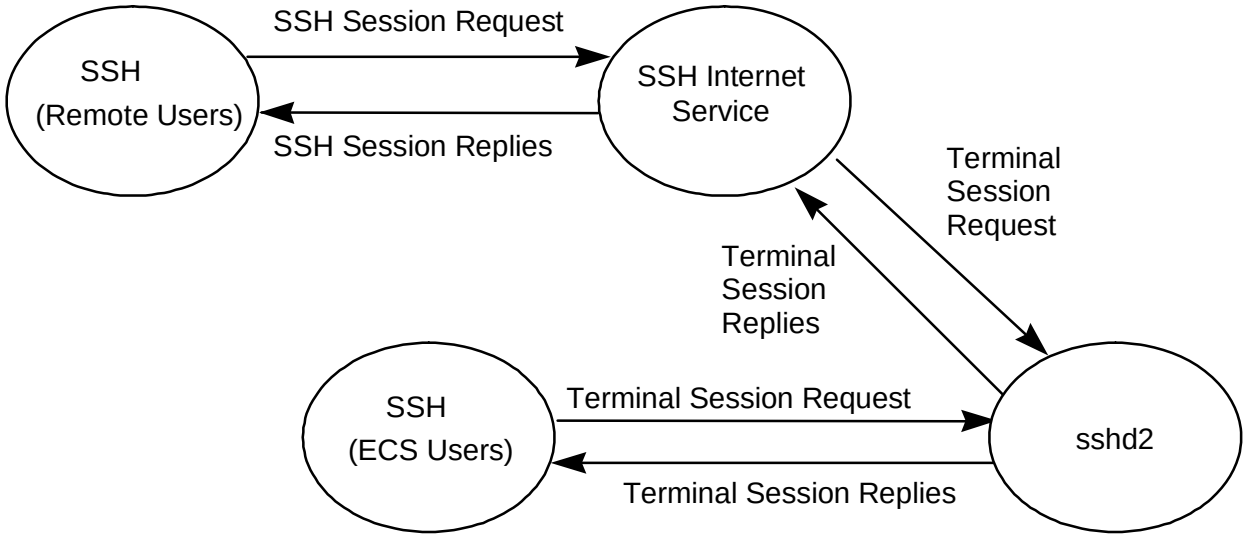

Figure 4.6-2. Virtual Terminal Architecture Diagram

4.6.1.2.4 Virtual Terminal Process Descriptions

Table 4.6-3 provides the descriptions of the processes shown in the Virtual terminal architecture diagram.

Table 4.6-3. Virtual Terminal Processes

Process	Type	Hardware CI	COTS/ Developed	Functionality
SSH (Remote Users)	Client	CSSHW	COTS	Provides the dial-up terminal session as requested on the client-side via remote service.
SSH (ECS Users)	Client	CSSHW	COTS	Provides the user interface to a remote system using the SSH protocol.
(Internet Service)	Server/Client	CSSHW	COTS	Enables users to interact with the host through a remote service.
sshd2	Server	CSSHW	COTS	Function provides servers supporting SSH with virtual terminal protocol.

EBIS Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.6.1.2.5 Virtual Terminal Process Interface Descriptions

Table 4.6-4 provides the descriptions of the interface events shown in the Virtual Terminal architecture diagram.

Table 4.6-4. Virtual Terminal Process Interface Events

Event	Event Frequency	Interface	Initiated by	Event Description
SSHSession Request	One per connection request	*Process:* sshd2 (dial-up service)	*Process:* SSH (COTS – remote users)	Any **ECS user** requiring a logon to another machine from the current machine. Users request to establish connection to a specified host via SSH.
SSH Session Replies	One per session reply	*Process:* SSH (remote service)	*Process:* sshd2 (remote service)	The **SSH Server** service provides **users** a remote session to request a terminal session to the secure shell client.
Terminal Session Request (SSH)	One per request to establish a session	*Process:* sshd2	*Process:* sshd2 (remote service)	Either the user or the client application service requests to establish a session with the specified host.
Terminal Session Replies (SSH)	One per connection request	*Process:* SSH (ECS users)	*Process:* sshd2	The Host Virtual Terminal Process, sshd2, responds to the connection requests and establishes or maintains the sessions.

4.6.1.2.6 Virtual Terminal Data Stores

Data stores are not applicable for the Virtual Terminal.

4.6.1.3 Cryptographic Management Interface Software Description

4.6.1.3.1 Cryptographic Management Interface Functional Overview

The Cryptographic Management Interface (CMI) classes provide the requesting process with a server account and a randomly generated password so the server can access security required services (i.e., Sybase ASE). These passwords (and optionally login names) are generated dynamically based on a psuedo-random number used as the seed for the password.

4.6.1.3.2 Cryptographic Management Interface Context

Figure 4.6-3 is the Cryptographic Management Interface context diagram. Servers (PF or non-PF) use the CMI with a need for access to security required services. Table 4.6-5 provides descriptions of the interface events shown in the Cryptographic Management Interface context diagram.

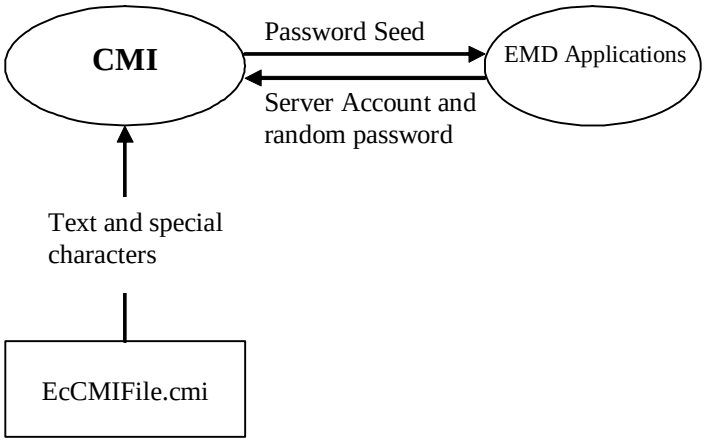

Figure 4.6-3. Cryptographic Management Interface Context Diagram

Table 4.6-5. Cryptographic Management Interface Events

Event	Interface Event Description
Password seed	The **ECS applications** request an account and provide a password seed to CMI.
Server account and random password	Account with random passwords created for the server is passed back to the server.
Text and special characters	Text and special characters read from a file for password generation.

4.6.1.3.3 Cryptographic Management Interface Architecture

Figure 4.6-4 is the Cryptographic Management Interface (CMI) architecture diagram. The diagram shows the event traffic between the CMI process and the ECS applications that interact with CMI for database connections.

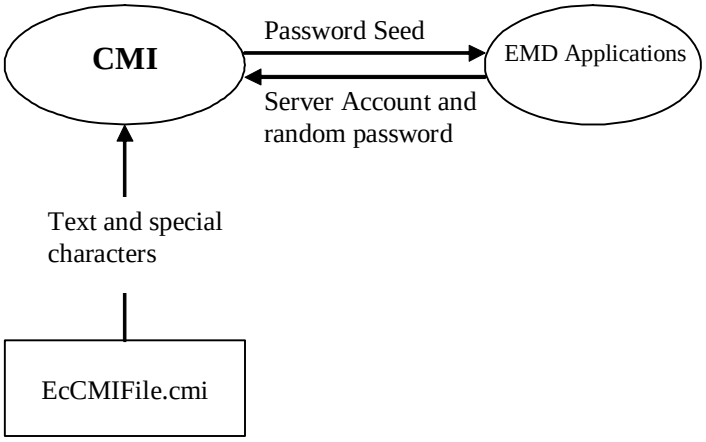

Figure 4.6-4. Cryptographic Management Interface Architecture Diagram

4.6.1.3.4 Cryptographic Management Interface Process Descriptions

Table 4.6-6 provides descriptions of the processes shown in the Cryptographic Management Interface context diagram.

Table 4.6-6. Cryptographic Management Interface Processes

Process	Type	Hardware CI	COTS/ Developed	Functionality
ECS Process Names	Server	CSSHW	Developed	Requests account with random password for access to security required services.
CMI	Other	CSSHW	Developed	A server account and randomly generated password are returned to the requesting server.

EBIS Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.6.1.3.5 Cryptographic Management Interface Process Interface Descriptions

Table 4.6-7 provides the descriptions of the interface events shown in the Cryptographic Management Interface architecture diagram.

Table 4.6-7. Cryptographic Management Interface Process Interface Events

Event	Event Frequency	Interface	Initiated by	Event Description
Password seed	One per password seed	*Process:* CMI *Library:* EcSeCmi *Class:* EcSeCmi	*DCCI Process:* EcCsRegistry	The server provides a unique number as a seed for generating a password to the **ECS Applications**.
Server Account and random password	One per account and password	*DCCI Process:* EcCsRegistry	*Process:* CMI *Library:* EcSeCmi *Class:* EcSeCmi	CMI generates a random password for the account based on the seed.

4.6.1.3.6 Cryptographic Management Interface Data Stores

Table 4.6-8 provides descriptions of the data store shown in the Cryptographic Management Interface architecture diagram.

Table 4.6-8. Cryptographic Management Interface Data Stores

Data Store	Type	Functionality
EcCMIFile.cmi	File	This is a flat file of textual and special characters used by the CMI password generation algorithm to create passwords.

4.6.1.4 Domain Name Server Software Description

4.6.1.4.1 Domain Name Server Functional Overview

Domain Name Server (DNS) performs name-to-address and address-to-name resolution between hosts within the local administrative domain and across domain boundaries. DNS is COTS software implemented as server by running a daemon called “in.named.” Servers running the in.named daemon are referred to as name servers.

The server is implemented through a resolver instead of a daemon from the client side. The function of in.named is to resolve user queries for device names or addresses (DNS requires the address of at least one name server to be in the file /etc/resolv.conf). The name server, when queried for a name or an address, returns the answer to the query or a referral to another name server to query for the answers.

Each domain uses at least two kinds of DNS servers (primary and secondary) to maintain the name and address data corresponding to the domain. The primary server keeps the master copy of the data when it starts up in the “in.named,” daemon and delegates authority to other servers

both inside and outside of its domain. A secondary server maintains a copy of the name and address data for the domain. When secondary server boots in.named, it requests the data for a given domain from the primary server. The secondary server then checks with the primary server periodically and requests updates to the daemon data so the secondary server is kept up to date with the primary.

DNS namespace is hierarchically organized, with nested domains, like directories. The DNS namespace consists of a tree of domains. Figure 4.6-5 is an Internet domain hierarchy diagram. The top-level domains are edu, arpa, com, gov, net, and for simplicity, not showing org, mil, and int, at the root level. The second level domain is nasa for gov. The third level domain is ecs for the EED project for nasa.gov.

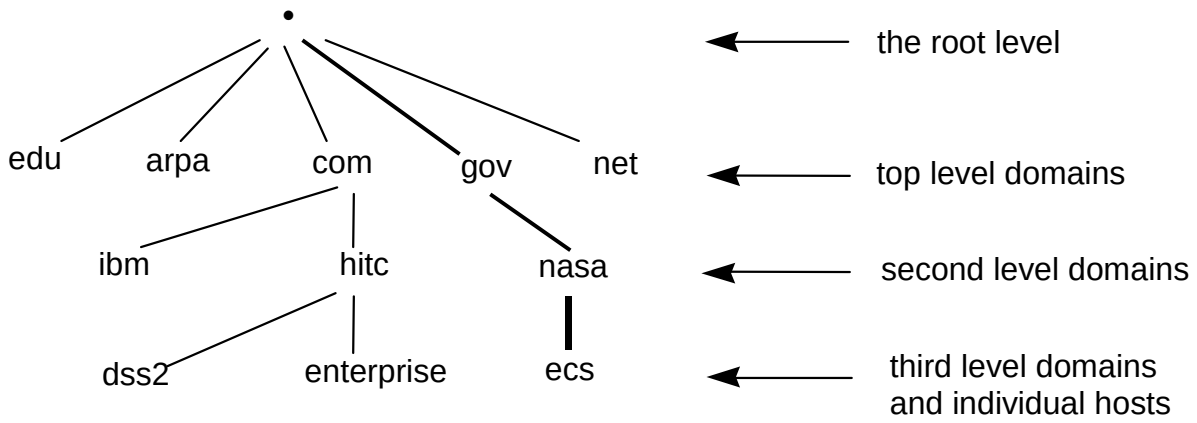

Figure 4.6-5. Domains Hierarchy Diagram

The fourth level domains in the EED project include domains of DAACs: gsfcb, gsfcmo, and etc. Figure 4.6-6 is the hierarchy diagram of the fourth level domains in the EED project. The DAAC and M&O domains are part of the overall DNS. The top-level domain is ecs.nasa.gov and the two lower level domains for the DAACs, for example, gsfcb.ecs.nasa.gov and gsfcmo.ecs.nasa.gov for the GSFC DAAC. The former is for the production network and the latter are for the GSFC M&O network.

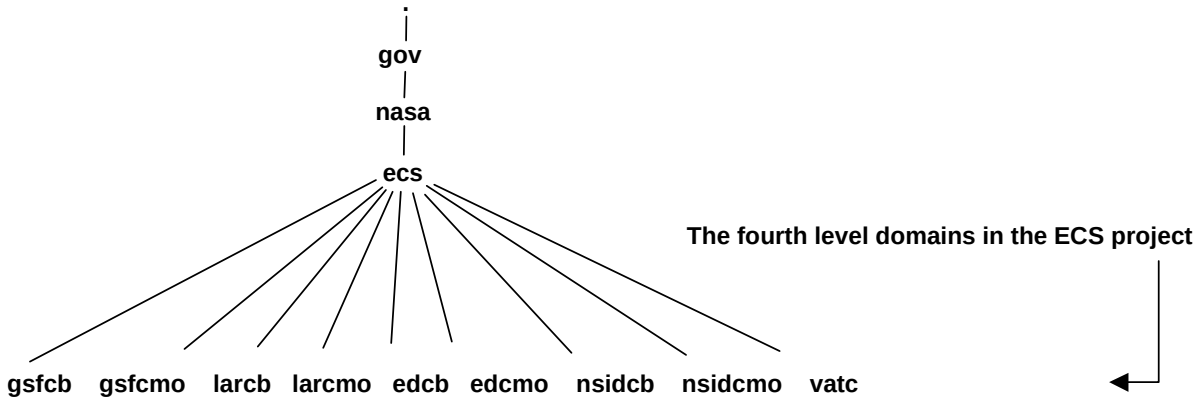

Figure 4.6-6. DNS Domains of the EED Project Diagram

Figure 4.6-7 is the ECS topology domain diagram.

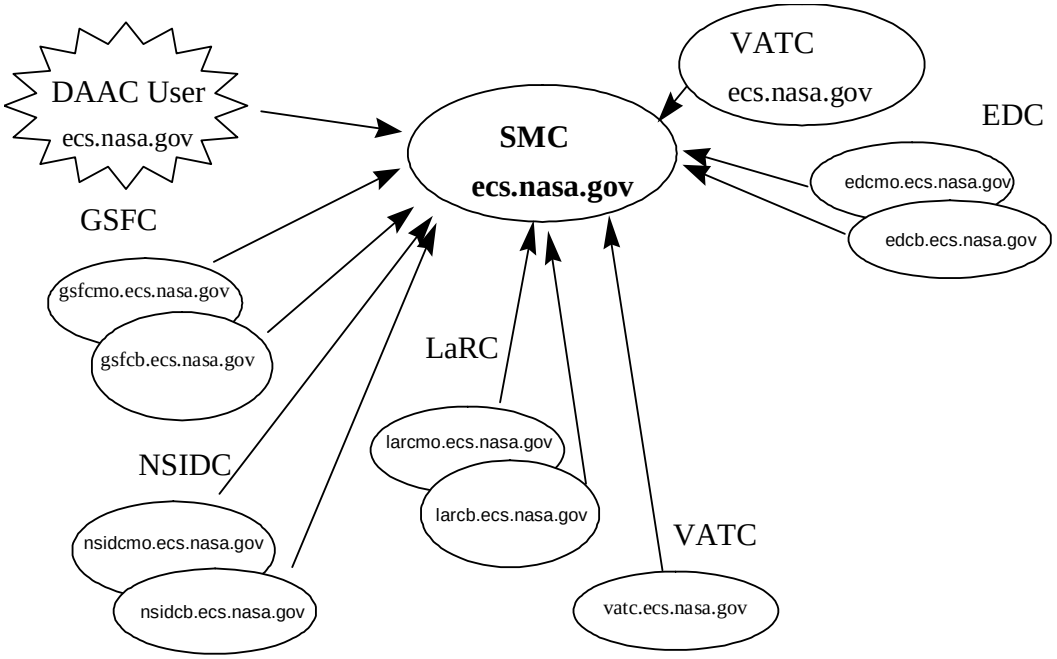

Figure 4.6-7. ECS Topology Domains Diagram

4.6.1.4.2 Domain Name Server Context

Figure 4.6-8 is the Domain Name Server context diagram.

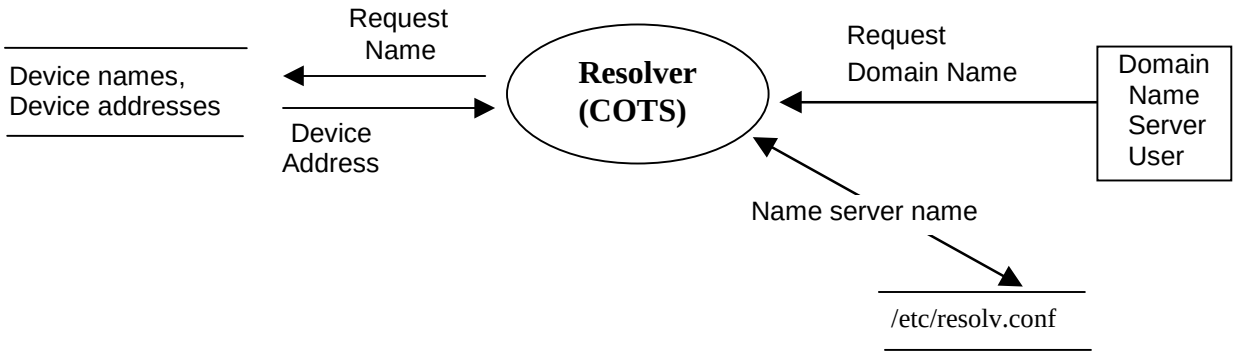

Figure 4.6-8. Domain Name Server Context Diagram

4.6.1.4.3 Domain Name Server Architecture

The Domain Name Server architecture diagram is the same as the context diagram and is not duplicated here. When the DNS client has a request for data, it searches the servers listed in the /etc/resolv.conf file in the order the servers were added to the file. When the first server does not contain the information of interest for the client, the second server in the list is searched and the search continues until the information is found.

4.6.1.4.4 Domain Name Server Process Descriptions

Table 4.6-9 provides descriptions of the Domain Name Server processes shown in the Domain Name Server context diagram.

Table 4.6-9. Domain Name Server Process

Process	Type	Hardware CI	COTS/ Developed	Functionality
resolver	Client	CSSHW	COTS	Searches data store of device names and device addresses for information requested in the Domain Name Request. First entry in the file /etc/resolv.conf is used as the place to start searching.

EBIS Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.6.1.4.5 Domain Name Server Process Interface Descriptions

Table 4.6-10 provides descriptions of the interface events shown in the Domain Name Server architecture diagram.

Table 4.6-10. Domain Name Server Process Interface Events

Event	Event Frequency	Interface	Initiated by	Event Description
Request Domain Name	One per user request	*COTS Software:* resolver	User	A DNS **user** requests data.
Name server name	One per search directory change	Data Store	*COTS Software:* resolver	The resolver retrieves the pathname for the directory to search for the user requested data from the /etc/resolv.conf database table. New file names are added to the list in the order they are stored.
Device Address	One per resolved address	*COTS Software:* resolver	*COTS Software:* name server	Returns the resolved address to the domain name requester via the Resolver.
Request Name	One per domain name request	*COTS Software:* name server	*COTS Software:* resolver	The resolver retrieves the domain name (device name and address) for the name server from an internal file used by the COTS software.

4.6.1.4.6 Domain Name Server Data Stores

Table 4.6-11 provides descriptions of the data store shown in the Domain Name Server architecture diagram.

Table 4.6-11. Domain Name Server Data Stores

Data Store	Type	Functionality
/etc/resolv.conf	Other	Stores the primary and secondary server names.

4.6.1.5 Infrastructure Libraries Group Description

4.6.1.5.1 Infrastructure Libraries Group Functional Overview

The Infrastructure Library Group (ILG) is a library of reusable software frameworks and infrastructures used by ECS servers configured as a distributed client-server system. Table 4.6-12 provides descriptions of the infrastructures in the ILG.

Table 4.6-12. Infrastructure Libraries (1 of 2)

Library	Description
Process Framework (PF)	The PF is a software library of services, which provides a flexible mechanism (encapsulation) for the ECS client and server applications to transparently include specific ECS infrastructure features from the library of services, such as mode management, error and event logging, life-cycle services, and the CCS Middleware Naming Service.

Table 4.6-12. Infrastructure Libraries (2 of 2)

Library	Description
Server Request Framework (SRF)	The SRF infrastructure provides the standard for ECS synchronous and asynchronous communications between ECS applications. SRF is used to provide the client-server communications between the DPL INGEST Request Manager and Granule Server. SRF provides enhanced CCS Middleware call message passing and persistent storage as a CSS support capability with the described features available by subsystem request.
Universal References (UR)	A Universal Reference provider object from C++ objects generates UR during their run time in virtual memory. The UR is a representation of the original object. URs can be transformed from an object to an ASCII representation and again returned to an object. URs are objects the users and applications use with full capabilities. Once the UR is obtained, the original object can be discarded and later reconstituted and used. URs can refer to objects local or remote to an address space. Therefore, the object does not have to remain in memory, and can, as appropriate, be written to a secondary storage system like a database.
Error/Event Logging	Event/Error logging is the capability of recording events into files and provides a convenient way to generate and report detailed events. All ECS CSCIs use event and error logging as an audit trail for all transactions (requests for data or services) that occur during the ECS data processing and distributing.
Message Passing (MP)	Message Passing provides peer-to-peer asynchronous communications service, which notifies clients of specific event triggers. This service is provided upon subsystem request by the CSS. It is an alternative means of communication.
ServerUR	Provides unique identification (universal reference) for data and service objects in the ECS. The Server Locator is a class that enables servers to register their location without referring to its physical location and be uniquely identified and located in the ECS. Client applications use the Server Locator to find any registered server. The Server Locator is used in ECS in any client-server CCS Middleware-based communication.
Fault Handling (FH)	The Failure Recovery Framework provides a general-purpose fault recovery routine enabling client applications to reconnect with servers after the initial connection is lost. This is accomplished through the CCS Naming Service, through which the Failure Recovery Framework can determine whether a server is listening. The Failure Recovery Framework provides a default and configurable amount of retries and duration between retries. This fault recovery takes effect for each attempt by the client to communicate with the server for all applications that employ the Failure Recovery Framework.
DBWrapper directory	The DBWrapper directory is the DBMS Interface Infrastructure Library used by ECS applications to connect to the Sybase ASEs. Sybase ASEs operate by ECS defined guidelines for mode management, thread safety, error handling, error recovery, security, configuration management, and performance of database connections.

4.6.1.5.2 Infrastructure Libraries Group Context

A context diagram is not applicable to the Infrastructure Libraries Group.

4.6.1.5.3 Infrastructure Libraries Group Architecture

An architecture diagram is not applicable to the Infrastructure Libraries Group.

4.6.1.5.4 Infrastructure Libraries Group Process Descriptions

Descriptions of the individual processes in the Infrastructure Libraries Group are not applicable.

4.6.1.5.5 Infrastructure Libraries Group Interface Descriptions

Table 4.6-13 provides descriptions of the interfaces the Infrastructure Libraries Group.

Table 4.6-13. Infrastructure Libraries Group Interfaces (1 of 3)

Library	Interface	Initiated by	Library Description
Process Framework (PF)	*Library:* EcPf *Classes:* EcPfManagedServer, EcPfClient	EcCsRegistry, DPL Servers OMS	The PF is a software library of services, which provides a flexible mechanism (encapsulation) for the ECS client and server applications to transparently include specific ECS infrastructure features from the library of services. Features and services include: • Mode management • Error and event logging • Life-cycle services • CCS Naming Service

Table 4.6-13. Infrastructure Libraries Group Interfaces (2 of 3)

Library	Interface	Initiated by	Library Description
Universal References (UR)	*Library (Common)*: EcUr	Object Origination	A Universal Reference provider object from C++ objects generates UR during their run time in virtual memory. The UR is a representation of the original object. URs can be transformed from an object to an ASCII representation and again returned to an object. URs are objects the users and applications use with full capabilities. Once the UR is obtained, the original object can be discarded and later reconstituted and used. URs can refer to objects local or remote to an address space. Therefore, the object does not have to remain in memory, and can, as appropriate, be written to a secondary storage system like a database.
Error/Event Logging	*Library:* event *Class:* EcLgErrorMsg	DPL Servers OMS	Event/Error logging is the capability of recording events into files and provides a convenient way to generate and report detailed events. All ECS CSCIs use event and error logging as an audit trail for all transactions (requests for data or services) that occur during the ECS data processing and distributing.
ServerUR	*Library (Common):* EcUr *Class:* EcUrServerUR	*Processes:* EcOmOrderManager DPL Servers *Classes:* EcNsServiceLoc *DSS Libraries:* DsBt, DsDe1, DsGe	Provides unique identification (universal reference) for data and service objects in the ECS. The Server Locator is a class that enables servers to register their location without referring to its physical location and be uniquely identified and located in the ECS. Client applications use the Server Locator to find any registered server. The Server Locator is used in ECS in any client-server CCS Middleware-based communication.

Table 4.6-13. Infrastructure Libraries Group Interfaces (3 of 3)

Library	Interface	Initiated by	Library Description
Fault Handling (FH)	*Library:* EcFh *Class:* EcFhExecutor	EcOmOrderManager DPL Servers	The Failure Recovery Framework provides a general-purpose fault recovery routine enabling client applications to reconnect with servers after the initial connection is lost. This is accomplished through the CCS Naming Service, through which the Failure Recovery Framework can determine whether a server is listening. The Failure Recovery Framework provides a default and configurable amount of retries and duration between retries. This fault recovery takes effect for each attempt by the client to communicate with the server for all applications that employ the Failure Recovery Framework.

4.6.1.5.6 Infrastructure Library Group Data Stores

Data Stores are not applicable for the Infrastructure Library Group.

4.6.2 The Distributed Computing Configuration Item Context

Figure 4.6-9 is the Distributed Computing Configuration Item (DCCI) CSCI context diagrams. The diagrams show the events sent to the DCCI CSCI and the events the DCCI CSCI sends to other CSCIs. Table 4.6-14 provides descriptions of the interface events shown in the DCCI CSCI context diagrams.

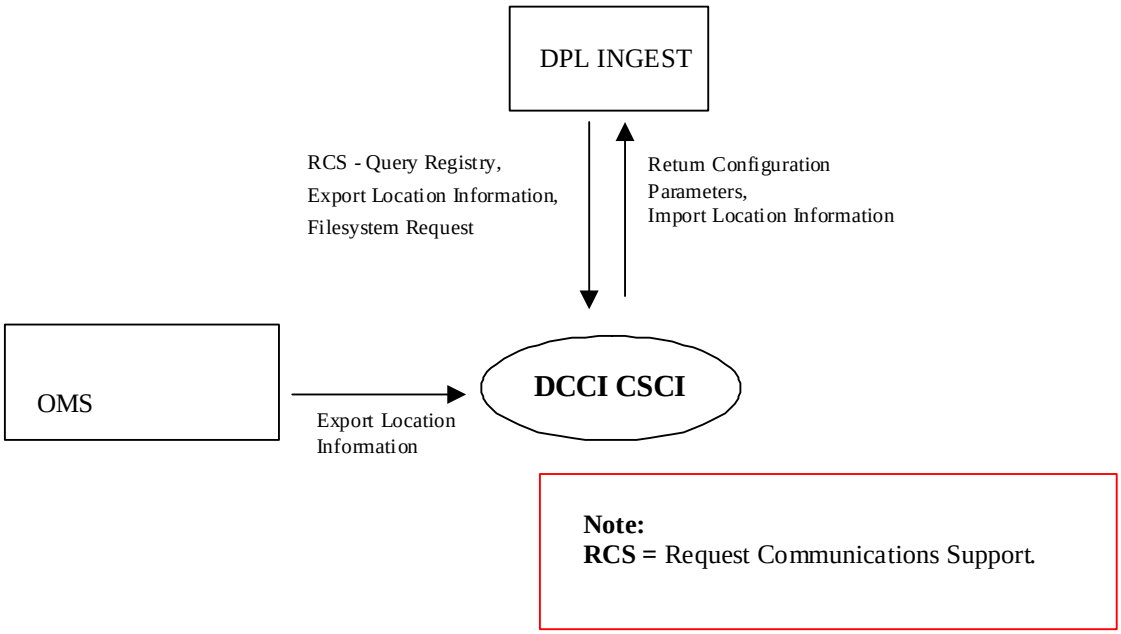

Figure 4.6-9. Distributed Computing Configuration Item (DCCI) CSCI Context Diagram

Table 4.6-14. Distributed Computing Configuration Item (DCCI) CSCI Interface Events (1 of 2)

Event	Interface Event Description
Filesystem Request	The NFS clients request ECS files or directories via an established mount point. The NFS Server makes the storage device(s) and its data accessible for use by the clients.
Submit Subscription	The **DPL CSCI** submits a subscription request to the DCCI CSCI using the advertisement subscribing to an insert event for an ESDT.
Password Seed	The **DPL CSCI** requests an account and provides a password seed to the CMI.
Notify of Subscription	The DCCI CSCI sends notification (via message passing) to the **DPL CSCI** when the subscribed event occurs.
Server Account and random password	An account with random passwords, created for the server, is passed back to the server in the **DPL** CSCI from the DCCI CSCI.
Password Seed	The **DPL CSCI** requests an account and provides a password seed to the CMI.
Import Location Information	The **DPL CSCI** request server location information from the CCS Name Server.
Server Account and random password	An account with random passwords, created for the server, is passed back to the server in the **DPL CSCI** from the DCCI CSCI.

Table 4.6-14. Distributed Computing Configuration Item (DCCI) CSCI Interface Events (2 of 2)

Event	Interface Event Description
Import Location Information	The **DPL INGEST CSCI** request server location information from the CCS Name Server.
Export Location Information	The DCCI CSCI stores physical and logical location information, received from the **DPL INGEST CSCIs,** in the CCS Name Server.
Request Communications Support	The DCCI CSCI provides a library of services available to the **DPL INGEST CSCI**. The CSCI services required to perform specific assignments are requested from the DCCI CSCI. These services include: • CCS Middleware Support • Database Connection Services • File Transfer Services • Network & Distributed File Services • Bulk Data Transfer Services • File Copying Services • Name/Address Services • Password Services • Server Request Framework (SRF) • Universal Reference (UR) • Error/Event Logging • Message Passing • Fault Handling Services • Mode Information • Query Registry - Retrieving the requested configuration attribute-value pairs from the Configuration Registry Request Distribution Media Options from the Configuration Registry

4.6.3 Distributed Computing Configuration Item Architecture

An architecture diagram is not applicable for the DCCI CSCI. However, Table 4.6-15 shows the mapping between CSMS CSCIs and CSS CSCs.

Table 4.6-15. CSMS CSCI to CSS CSC Mappings

CSMS CSCI	CSS CSC	Process Used	CSS Libraries Used
DPL INGEST	– CCS Middleware – E-Mail Parser Gateway Server – FTP – NFS – Configuration Registry Server	– EcCsIdNameServer – EcCsEmailParser – ftp_popen – NFS Client – EcCsRegistry	– PF – ServerUR – Error Logging – Event Logging – UR – Fault Handling Services – Server Request Framework (SRF) – CCS Middleware

4.6.4 Distributed Computing Configuration Item Process Descriptions

Process descriptions are not applicable for the DCCI CSCI.

4.6.5 Distributed Computing Configuration Item Process Interface Descriptions

Process interface descriptions are not applicable for the DCCI CSCI.

4.6.6 Distributed Computing Configuration Item Data Stores

Data stores are not applicable for the DCCI CSCI.

4.6.7 Communications Subsystem Hardware CI Description

Document 920-TDx-001 (HW Design Diagram) provides descriptions of the Distributed Computing Configuration HWCI and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

Three DCCI software programs run on this host including the Domain Name Server (DNS), Network Information Services (NIS), and Mail Server. DNS enables host names to be distinguished based on their host name and IP address relationship. NIS is a service that stores information that users, workstations, and applications must have to communicate across the network. This information includes machine addresses, user names, passwords, and network access permissions. The Mail Server provides standard electronic mail capability.

The CSS Server is a stand-alone host and intrinsically does not have fail-over capability. DNS and Distributed Time Service (DTS) are loaded on multiple hosts designated as secondary. Any one of these hosts can operate as primary servers for the DNS or DTS services in the event of non-recoverable hardware failure of the primary host.

4.7 Internetworking Subsystem (ISS) Overview

The Internetworking Subsystem (ISS) contains one hardware configuration item (HWCI), the Internetworking HWCI. INCI provides internetworking services based on protocols and standards corresponding to the lower four layers of the OSI reference model as described below.

Transport Protocols

EED provides IP-based connection-oriented and connectionless transport services. The connection-oriented service is implemented using TCP, while User Datagram Protocol (UDP) is used for connectionless transport. Higher layer applications use one or the other based on such requirements as performance and reliability.

Transmission Control Protocol (TCP), specified in RFC 793, is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols to support multi-network applications. It provides for reliable inter-process communication between pairs of processes in host computers attached to networks within and outside EED. Because TCP assumes it may obtain potentially unreliable datagram service from the lower level protocols, it involves additional overhead due to the implementation of re-transmission and acknowledgment processes.

The UDP, specified in RFC 768, provides a procedure for application programs to send messages to other programs with minimal overhead. The protocol is transaction oriented and delivery of data is not guaranteed, since there is no acknowledgment process or re-transmission mechanism. Therefore, applications requiring ordered and reliable delivery of data would use TCP.

Network Layer Protocols

The network layer provides the functional and procedural means to transparently exchange network data units between transport entities over network connections, both for connection-mode and connectionless-mode communications. It relieves the transport layer from concern of all routing and relay operations associated with network connections.

The Internet protocol (IP) Version 4, specified in RFC 791, is the EED supported network protocol, based on its dominance in industry usage and wide community support. As part of IP support, ICMP and ARP are also supported.

Physical/Datalink Protocols

Physical and data-link protocols describe the procedural and functional means of accessing a particular network topology. For the DAAC networks, the data-link/physical protocol is 10/100/1000 Mbps Ethernet.

Internetworking Hardware HWCI (INCI)

This HWCI provides the networking hardware for internal and external DAAC connectivity. The HWCI includes Ethernet switches and cabling; routers and cabling; and network test equipment. Each network hardware device is discussed in detail in Section 4.7.2.

4.7.1 Internetworking Subsystem Description

4.7.1.1 DAAC LAN Architecture

This section provides an overview of the DAAC network architecture. Information on DAAC specific implementation level detailed designs can be found in Section 4.7.1.4.

The generic architecture for DAAC Local Area Networks (LANs) is illustrated in Figure 4.7-1. The topology consists of a Production Network, and a SAN LAN Network. The Juniper SRX650 protects the production network at NSIDC. The general network layout is displayed in the diagram below.

Note that not all sites have the complete complement of hardware and subsystems shown in Figure 4.7-1. For instance EDC and LaRC do not have EED managed routers or firewalls, their networks are protected by local perimeter campus firewalls at their site locations. There are 2Gbsec Etherchannels interconnecting Cat6506 core switches and the IBM BladeCenter switches in EDC and LaRC.

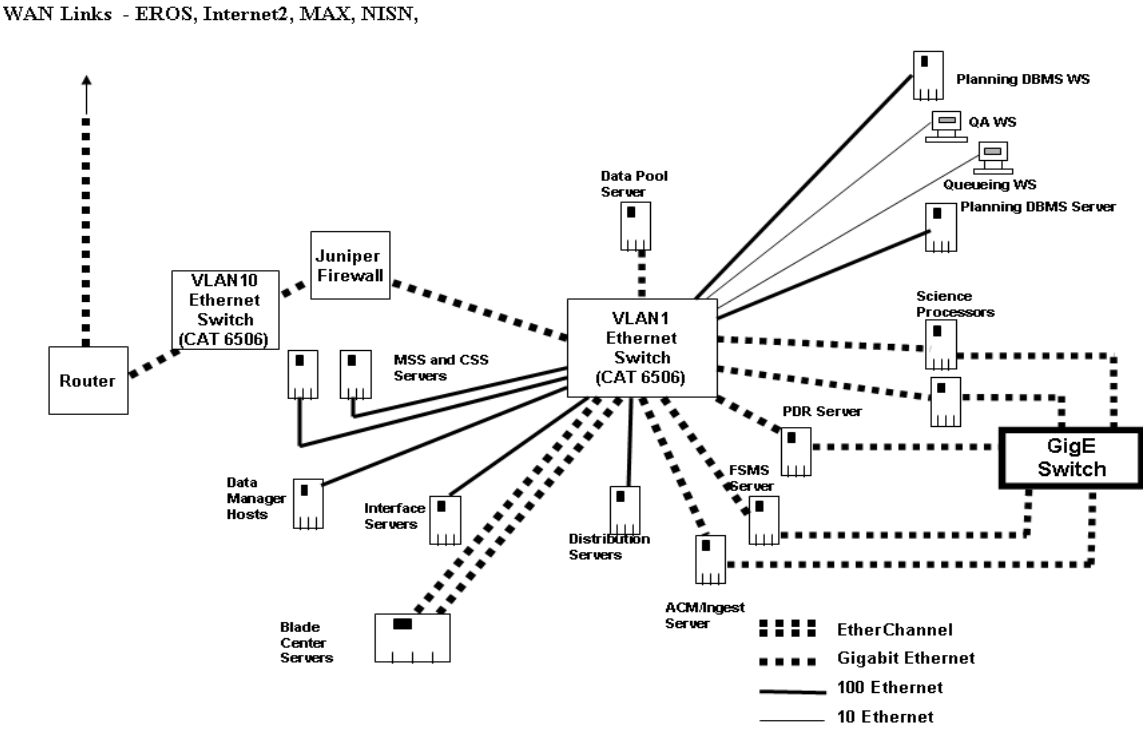

Figure 4.7-1. DAAC Networks: Generic Architecture Diagram

The Production Network consists of a Catalyst 6506 multi-port Ethernet Switch. All servers, workstations, printers, and the BladeCenter chassis are connected to individual switch ports.

The SAN LAN Network consists of a Catalyst 3560 multi-port Ethernet Switch. This network is used for the StorNext file system MetaData and to manage the storage arrays. All servers which use the StorNext file system and storage arrays are connected to this network. The servers in the BladeCenter chassis connect via the two internal Cisco Gigabit Ethernet Blade modules to both the Production and the SAN LAN Ethernet switches.

4.7.1.2 DAAC Addressing and Routing Architecture

Most of the devices connected to the Production Network are assigned Class C address space with the exception that LaRC, uses a /23 subnet for production network NSIDC devices connected to the Production Network and all devices that are connected to the SAN LAN are assigned private addresses as specified in RFC 1918 (as of 02/96). Documents that list IP address assignments to all hosts and network attached devices are listed in Table 4.7-1. All external EED address space (except for addresses used on the SAN LAN Ethernet networks) is provided from address blocks designated by NISN.

The use of static routes is the main peering protocol used to route IP packets within EED. Routing Information Protocol (RIPv2) is used to route IP packets from the PVC and VATC Production networks. EED Production Networks are advertised to EBnet via NISN.

4.7.1.3 Network-based Security Architecture

The network architecture provides a strong level of security by implementing stateful firewalls at the DAACs. The firewalls block incoming network traffic unless there is a rule specifically allowing the traffic to pass into the DAACs. Note: that in addition to network-based security; EED has implemented other security measures, such as two-factor authentication, secure shell (SSH) and host access lists (ACLs), which are discussed in the CSS sections of this document.

4.7.1.4 Internetworking Subsystem Detailed Design

The ISS implementation level detailed design is documented in the documents listed in Table 4.7-1. Document 920-TDx-001 (HW Design Diagram) provides descriptions of the ISS HWCI and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

All of the documents are under configuration control and can be obtained from EED Configuration Management (CM). The documents are not on line for security reasons. Therefore special authorization is needed for their release.

Table 4.7-1. Internetworking Subsystem Baseline Documentation List

Document Name	EDC	LaRC	NSIDC
Hardware/Network Diagram	921-TDE-002	921-TDL-002	921-TDN-002
Host IP Address Assignment Table	921-TDE-003	921-TDL-003	921-TDN-003
Network Hardware IP Address Assignment	921-TDE-004	921-TDL-004	921-TDN-004

4.7.2 Network COTS Hardware

The DAAC LANs contain three types of COTS hardware: Firewall, Ethernet switches, and Routers. All hosts in the DAACs are attached to Ethernet switches. The Routers are used to provide access to external networks (NISN, Abilene, and Campus nets). Table 4.7-2 provides a list of networking hardware used in EED networks.

The following descriptions of Network Hardware devices are provided as illustrative detail. All details of the hardware configuration should be verified with the appropriate Hardware/Network documents listed in Table 4.7-1.

Table 4.7-2. Networking Hardware for EED Networks

Networking Hardware	Vendor
Firewall	Portus IBM PowerPC Server
Router (EED Router)	Cisco 7200
Ethernet Switch	Catalyst 3560G
Ethernet Switch	Catalyst 6506
Ethernet Switch	Catalyst 2924
Ethernet Cables	10baseT, 100baseT, or 1000baseT connection to servers, workstations, printers, PCs, and x-terms

4.7.2.1 EED Ethernet Switch

The EED Ethernet switch is the Cisco Catalyst 6506 with multiple 10/100/1000 Mbps ports and powerful packet engines. The switch has a switching fabric of 32Gbps. It forms the core of the EED Production network by interconnecting all servers, workstations, printers, PCs, and x-terms. The switch has redundant power supply and fan units. It also has redundant packet engines. All modules are hot swappable.

4.7.2.2 EED Router

The EED Router is a Cisco 7200 series router running Cisco's Internetwork Operating System (IOS). The router has three 1000 Mbps Ethernet ports. The EED Router is only used at NSIDC

and it provides connectivity to EED sites and the Internet via its interfaces with NISN and the local campus network.

The ECS Router has redundant power supply and fan units.

For support purposes, the PVC and VATC also have 7200 routers which interface with EBnet at GSFC.

4.7.2.3 SAN LAN Ethernet Switch

The EED SAN LAN Ethernet switch is a Cisco Catalyst 3560G switch capable of supporting up to 48 10/100/1000 Mbps ports.

4.7.2.4 Firewall

The EED Firewall is Juniper SRX650 machine. It is a stateful type firewall, which is capable of supporting several 1 Gbps Ethernet interfaces with 7Gbps total throughput The firewall has a special software build version to accommodate the large FTP traffic.

At EDC and LaRC, the layer three Cisco Catalyst 6506 switches interface directly with the Campus firewalls. The Campus routers provide all external network connectivity.

4.8 EED General Process Failure Recovery Concepts

During Earth Observing System Data and Information System (EOSDIS) Evolution and Development (EED) processing, client or server failures can occur. These failures cause certain recovery events to take place within the EED. To understand the General Process Failure Recovery of the EED processes, several key concepts must be described. These failure recovery concepts are:

1) Client-Server Rebinding
2) Sybase Reconnecting
3) Request Identification
4) Request Responsibility
5) Queues
6) Request Responses
7) Duplicate Request Detection
8) Server Crash and Restart
9) Client Crash and Restart

These concepts compose the general philosophy of the EED process failure recovery. The General Process is performed as a "process chain" to service requests for data or other services (e.g., order tracking or data retrieval from another processing system) via the client/server architecture. A brief description of each of the key concepts for General Process Failure Recovery follows.

4.8.1 Client-Server Rebinding

EED performs communication between client and server through the database. A client will insert information concerning its request into the database and the server will poll the database to pick up new requests. This allows for persistent client/server messaging that is inherently fault tolerant. The exceptions to this are the EcDlQuickServer, HegServer, and EcDlFtpService, which communicate with clients using sockets. If a client of any of these services is unable to connect, it should suspend the resource associated with that service after a configurable number of retries.

4.8.2 Sybase Reconnecting

Communication with the Sybase database relies on standard libraries specific to each programming language. A Perl program should use the perl DBI module interface, a Java process should use a JDBC interface. A custom class that wraps the Sybase client library, EcCmDb is used for C++ code. In each of these cases the calling program should attempt a connection for a configurable number of times, waiting for a configurable amount of time between each connection attempt.

EED applications that require the establishment of a large number of connections should implement a connection pool that allows them to reuse connections. EED applications should handle deadlock errors with a configurable retry. If the request to the database succeeds before the retries are exhausted, the application continues normally and completes the transaction in progress when the Sybase fault occurred.

However, operations should be aware of the following facts:

- Not all EED applications are able to use the EED Sybase interface code
- Not all EED applications are able to use Sybase transactions and automatic re-connection in the manner described.

4.8.3 Request Identification

EED identifies requests with primary keys stored in the database. The use of RPC ID is restricted to a small set of OMS clients including the acquire script which is called by the MISR Order Tool. These clients generate a unique identifier for each type of request that requires fault-handling provisions.

Many kinds of requests do not pose recovery issues and thus, do not employ request identifiers.

4.8.4 Request Responsibility

The responsibility for the handling of recoverable requests by a server is given in the EED by determining if the request is synchronous or asynchronous.

Synchronous Requests

On a synchronous request, the application submitting the request is waiting for a response. Regardless of how the request is handled downstream, whether it succeeded or failed depends on the response the waiting application gets back. From its perspective, the request is not complete until it receives a response.

Therefore, if an EED application initializes a request and submits it synchronously, it has the responsibility for getting the request completed. This means if the request does not complete, for example, because the connection is lost to the server to which the request is submitted, the application needs to submit it again.

EED examples of synchronous interfaces include the Quick Server and Ftp Service.

Asynchronous Requests

When an application sends an asynchronous request, the receiving server is responsible for completing the request once it accepts the request. For example, the server may need to save the request (perhaps in a queue in a database) before sending an acknowledgment to the originating application. Of course, the server (Server A), can eventually complete processing the request and pass it on to another server (Server B), also asynchronously. Once Server B accepts the request, it is responsible for seeing it to completion.

EED examples of asynchronous interfaces include the Order Manager Server and its component servers.

4.8.5 Queues

The reason queues are mentioned here is because they represent an important aspect of recovery. If a server uses queues to defer work until later, it needs to be concerned about what happens to the queue if the server crashes. The recovery rules in the request responsibility section state:

- If the server queues up synchronous requests, the client application is responsible for recovering the synchronous requests.
- If the server queues up asynchronous requests after accepting them, it is responsible for the asynchronous requests, which means, if a queue contains asynchronous requests, the server must make sure it can recover the queue in case of a crash.

Instead, a server handling asynchronous requests must keep a queue in a safe place so it can be recovered in case of a restart (such a restart that recovers the current requests is called a "warm start"). If a warm start takes place with asynchronous requests, the sending application does not even notice there was a problem. The processing gets completed eventually.

Note, however, that queued synchronous requests require special consideration: If a warm start takes place and some of the queued requests are synchronous, the sending application is generally aware of the failure (it had to rebind, See Client-server Rebinding Section). Since it did not receive a response, it re-submits the request. The server must recognize the request as a re-submission and either ignore it or - if it already completed by the time the re-submission is received - return the completion status as a response to this request. Moreover, servers that might handle a large number of concurrent synchronous requests have to be able to deal with a sudden spike of request submissions following a warm start, as their clients re-submit these requests.

A warm start can cause a problem; for instance, one of the active requests may be the reason the server crashed. This could result in a warm restart loop: each time the warm start is attempted the server crashes again because of the request. In such a case, operations can use a cold start to empty the queue of all requests (at the expense of having to recover queued asynchronous requests that were lost manually).

4.8.6 Request Responses

Servers have the responsibility to classify a response appropriately. Client applications have the responsibility to process a response appropriately, depending on its type.

Client applications can pass the response on to the calling application (e.g., success, warning, or fatal error); or retry (retry error). At the beginning of a request chain, there may be a user or operator (if this is a user or operator submitted request). In this case, the error is passed back to the user/operator for action where possible.

Where this is not possible (e.g., system generated requests, or if a data order runs into an error after it was already accepted and the user/operator is no longer connected), errors are logged.

They are brought to the attention of the DAAC operations staff for action if there is a corresponding server GUI for the operator. However, not all EED servers are associated with an operator GUI. In these cases, operators need to monitor the server logs for errors on a regular basis.

Failure events are classified as having any of three severity levels:

- Fatal errors,
- Retry errors and
- Warnings

Fatal errors are returned when a request cannot be serviced, even with operator intervention. For example, if a request is made to distribute data via FTP to a non-existent host, the request is failed with a fatal error.

Retry errors can be recovered from and such errors should be returned back to the client only when the server cannot recover from the error automatically. Retry errors may also necessitate operator assistance for recovery purposes, such as in the case of a tape left in a device that must be manually removed.

Warnings are provided where operations can proceed without interruption, but where an unexpected circumstance was detected. For example, if a client requests a file to be removed, and the file does not exist, there is no error, per se, but a warning is generated to caution the client the file to be removed did not exist in the first place.

The situation where a server does not return a response represents a special case. It can occur, for example, when an application calls a server and the server crashes before it can send a response or there is a communication error that prevents a response within a reasonable time. The situation is important because now the client application does not really know what happened to the request:

a. Did it reach the server?

b. Did the server start the request but not complete it?

c. Did the server complete the request with an error but was not able to send the error response?

d. Did the server process it successfully?

The EED recovery policy is that in such situations, the client application should either re-submit the request or if it is possible, return an appropriate error to the user/operator who submitted the request (to avoid leaving them with a hanging GUI while EED goes through endless retries).

Note that if the request did reach the server, the server now sees the request twice (i.e., this has become a duplicate request). Therefore, there need to be provisions to handle duplicate requests gracefully.

Table 4.8-1 summarizes the five categories of request responses, and the specific requirements for the application or server currently responsible for the request. EED servers have been directed to classify their responses accordingly.

Table 4.8-1. Request Responses

Request Response	Response Description
Success	The server sends back a message to acknowledge the successful completion of the request to the client. The request is considered complete.
Warning	This is provided where operations proceed without interruption, but where an unexpected circumstance is detected. The calling application needs to determine whether to alert the user or operator of the situation.
Error, retry the request	This can happen if the server encountered a temporary error condition, such as a media error on output. The request can be "retried" and the application responsible for the request should re-submit it after a suitable wait time. However, if the request does not succeed after a (configurable) number of retries, it should be considered "failed." If a GUI supports the application, the request may be suspended (if it makes sense to alert the operations staff to remedy the situation).
Error, cannot retry the request	This can happen if the server encounters an error condition that is sure to re-occur if the same request is submitted again. Examples might be a syntax error in the request (indicating some internal software problem), or an attempt to retrieve a non-existent granule. The request is considered "failed." The server responsible for the request sends back a failure notification. If a GUI supports the application, the request may be suspended (if it makes sense to get the operations staff involved at this point), but the operations staff may or may not be able to help.
No response returned by server	This can happen, for example, if the server to which the request was submitted crashes before a response or an acknowledgment is returned. In this case, the client can make no assumptions about the request. The client responsible for the request should send the request again or retry the request.

Transient errors such as network errors are always retry errors. In general, clients and servers that experience transient, retry errors can first attempt to recover by retrying the operation automatically. One special case of this is “rebinding”. Rebinding refers to the process by which a client automatically attempts to re-establish communications with a socket server in the event communications are disrupted. This disruption may be caused by transient network failure, or by the server being brought down or crashing. In any case, the client automatically attempts to reconnect to the server for a configurable period of time on a client-by-client basis.

EED processes that encounter an error or receive an error from a server request may either pass the error back to a higher-level client or present it to the operator for operator intervention. The fault handling policies are detailed in Table 4.8-2.

Table 4.8-2. Fault Handling Policies (1 of 2)

CI	Client Processes	Fault Handling Policy
DPLINGEST	EcDlInPollingService	Retry errors: Errors are retried a configurable number of times for resources, then the resource is suspended. Examples of retriable errors are connection failures and timeouts for file transfers. Fatal errors: Resources are suspended immediately for non-transient errors. Examples of non-transient errors are host address does not exist, and login to host failed.
	EcDlInProcessingService	Retry errors: Errors are retried a configurable number of times for resources, then the resource is suspended. Examples of retrable errors are connection failures and timeouts for quick server operations such as checksum and band extraction. Fatal errors: Resources are suspended immediately for non-transient errors. Examples of non-transient errors are quick server not running and failures to login to a transfer host.
	EcDlInNotificationService	Retry errors: Errors are retried a configurable number of times for resources, then the resource is suspended. Examples of retriable errors are connection failures and timeouts for file transfers. Fatal errors: Resources are suspended immediately for non-transient errors. Examples of non-transient errors are host address does not exist, and login to host failed.

Table 4.8-2. Fault Handling Policies (2 of 2)

CI	Client Processes	Fault Handling Policy
SSS	EcNbSubscribedEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	All errors are logged. Failed attempts to connect to Sybase are retried. Failed database queries are retried if the reason for failure was deadlock.
OMS	EcOmOrderManagerServer	All errors are logged. Failed attempts to connect to Sybase are retried. Retry errors: Errors are retried a configurable number of times, then passed back to the calling process. Fatal errors: Errors are logged and the request is suspended and operator intervention is generated. Operators then have a choice to hold, fail or resubmit the request.
BMGT	EcBmBMGT	All errors are logged in one of the BMGT log files. Errors for automatic cycles are always retried. Errors for all other types are retried unless they occur after generation has completed in which case they may be considered fatal. Database errors are retried to a configured retry limit, however serious database errors such as a missing stored procedure will cause a server to terminate. Repeated errors may suspend BMGT product generation and export and raise an alert in the operator GUI, depending on the type of error.
DPL	EcDlActionDriver	Retriable errors are retried a configurable number of times, after which the request will be failed. If the error is associated with a resource, such as the unavailability of a file system, the resource will be suspended, and all requests using that resource will also be suspended. The Action Driver will periodically test the associated resource, and automatically resume all affected requests once the resource becomes available again. Fatal errors, such as invalid granules, are failed immediately. All errors are logged in both application ALOG files, as well as in the database (a request that fails as a result of an error will have an error message recorded in the database).

4.8.7 Duplicate Request Detection

The above scheme for handling requests in cases of faults poses a potential problem. The request could have been re-submitted because there was no response returned by the server. But, in fact, the server completed the request but was unable to get the status back to the client (e.g., because

of communications problems or a machine crash). The following measures are intended to deal with this situation:

- **Trivial duplicate requests**. There are many interfaces where sending a new request to retry a service whose outcome is unknown either has no or negligible impact on the EED. This is because many EED services have been designed with this goal in mind. In these cases, the new request will simply replace the old and have the same end result as the original request and no unintended consequences.

- **Recognize non-trivial duplicate requests**. Where executing the same request more than once can have undesirable consequences, EED provides a mechanism for recognizing re-submitted requests. Each request is tagged with a unique identifier (see Request Identification Section). Upon submission of a request, the receiving server of the request must check the identifier and recognize when it is a re-submission of a previous request it received. For example, the server may realize the request has been completed and simply acknowledges the successful completion.

4.8.8 Server Crash and Restart

- **Server Crash**

  When a server crashes, the only impact on the system is that clients cannot continue to submit requests for processing. Synchronous requests in progress result in an exception being thrown back to the client process, which enters a rebinding failure recovery mode (see Client-Server Rebinding section above). Attempts to submit requests while the server is down results in the client blocking until a communications timeout has been reached.

- **Server Restart**

  When a server restarts, it may perform various re-synchronization activities in order to recover from an unexpected termination. In the event of a server cold start or cold restart, the server also cancels all outstanding requests and reclaims all associated resources. Note that the distinction between cold start and cold restart is described in the section above on Start Temperature. Specifics of server startup behavior are detailed in Table 4.8-3. Unless otherwise stated, existing request queues are always retained for warm restarts and cleared for cold starts or cold restarts.

Table 4.8-3. Server Response versus Restart Temperature

CI	Server(s)	Special Behavior on Warm Restart	Special Behavior on Cold Start or Cold Restart
DPLINGEST	EcDlInPollingService	None.	None.
DPLINGEST	EcDlInProcessingService	All ingest requests that did not reach a terminal state in the previous processing run will be re-queued in processing and executed from their last persisted state.	All ingest requests that did not reach a terminal state in the previous processing run will be moved to the state 'TERMINATED'. They will not be re-queued.
DPLINGEST	EcDlInNotificationService	None.	None.
SSS	EcNbSubscribedEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	N/A	N/A
OMS	EcOmOrderManagerServer	Full Recovery	N/A
DPL	EcDlActionDriver	The Action Driver recovers all existing requests from the database.	N/A The Action Driver does not have a cold restart capability.

- **Request Re-submission**

  Upon restarting a crashed client or server, requests are typically re-submitted. If the restarted process was started warm, the fault recovery capabilities permit the server to resume processing of the request from its last check-pointed state. This prevents needless repetition of potentially time-consuming activities. Specific behavior of servers upon re-submission of a request is detailed in Table 4.8-4. Note that a cell value of N/A means the server either has no clients or the clients do not re-submit requests.

Table 4.8-4. Server Response for Request Re-submission

CI	Server(s)	Behavior on Request Re-submission
DPLINGEST	EcDlInPollingService EcDlInNotificationService	N/A
DPLINGEST	EcDlInProcessingService.	The newly resubmitted request will have the same requestid and continue being processed from the last check-pointed state from the last processing run.
SSS	EcNbEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	There is no resubmission of requests. EcNbRecoverDriver monitors the SSS database for events or actions that did not run to completion and re-enqueues them.
OMS	EcOmOrderManagerServer	Incomplete requests in OMS are picked up and processed upon restarting OMS Server. The incomplete requests have the same requestid
DPL	EcDlActionDriver	N/A The Action Driver is asynchronous, with requests submitted via the database. Clients cannot detect whether or not the Action Driver is executing. However, normal operations frequently results in duplicate requests (requests for the same granule) from different clients, and the Action Driver is therefore designed to handle this.

4.8.9 Client Crash and Restart

- **Client Crash**

When a client crashes in the EED system, fault recovery-enabled servers have several possible responses. Servers may continue to service client requests, independent of the client's status. Servers may choose to suspend processing of client requests, but permit the requests to be resumed upon client recovery. Or, servers may terminate servicing of the client requests, canceling all work done on the requests. The behavior of each CI is detailed in Table 4.8-5. Note that the behavior of a server in the event of a client crash does not vary from client to client.

Table 4.8-5. Server Responses to Client Failures

CI	Server(s)	Behavior on Client Crash
DPLINGEST	EcDlInProcessingService EcDlInNotificationService	Requests in process are serviced to completion.
	EcDlInPollingService	N/A
SSS	EcNbSubscribedEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	Processing is database driven and not influenced by outside processes.
OMS	EcOmOrderManagerServer	Requests in process are serviced to completion.
DPL	EcDlActionDriver	Requests in process are serviced to completion.

- **Client Restart**

  Client restart behavior depends upon the type of client. For command line tools used by operators, it is generally the responsibility of the operator to decide whether or not the client needs to be restarted. Server processes acting as clients to other servers or processes will generally determine how to restart processing of the request using persisted checkpoints (for example, in the database).

Some known limitations within the EED are:

a.) Requests with many sub-requests can experience timing problems because of nested retries or because one of the requests is suspended.

b.) Coding errors can cause unanticipated fault behavior that is different from what is described above (and such occurrences should be reported as NCRs.)

c.) System engineers and designers may have made mistakes in classifying errors (e.g., as fatal versus retry.)

d.) Not all EED applications use the error recovery capabilities of the EED Sybase interface infrastructure.

4.9 Spatial Subscription Server (SSS) Subsystem Overview

The Spatial Subscription Server (SSS) subsystem is the principal means by which users can establish standing orders for data. Users enter subscriptions for specific ESDTs using a GUI or command line interface (CLI). A subscription may be qualified by specifying one or more constraints on the metadata of matching granules. This includes the capability of qualifying the subscription spatially by specifying a geographic area (rectangle) over which the data was collected. A subscription has one or more associated actions such as data distribution, email notification, Data Pool insertion, or bundling, i.e. adding a granule to an Order Manager bundle.

In addition to the subscription creation components, the SSS subsystem is comprised of a database, installed on a Sybase ASE server, and four runtime drivers: an event driver to match subscriptions with granule events, an action driver to execute the actions of matched subscriptions, a recovery driver to restart stalled events or actions, and a deletion driver to clean up the database.

Spatial Subscription Server (SSS) Context

Figure 4.9-1 is the Spatial Subscription Server context diagram. Table 4.9-1 provides descriptions of the interface events in the Spatial Subscription Server context diagram.

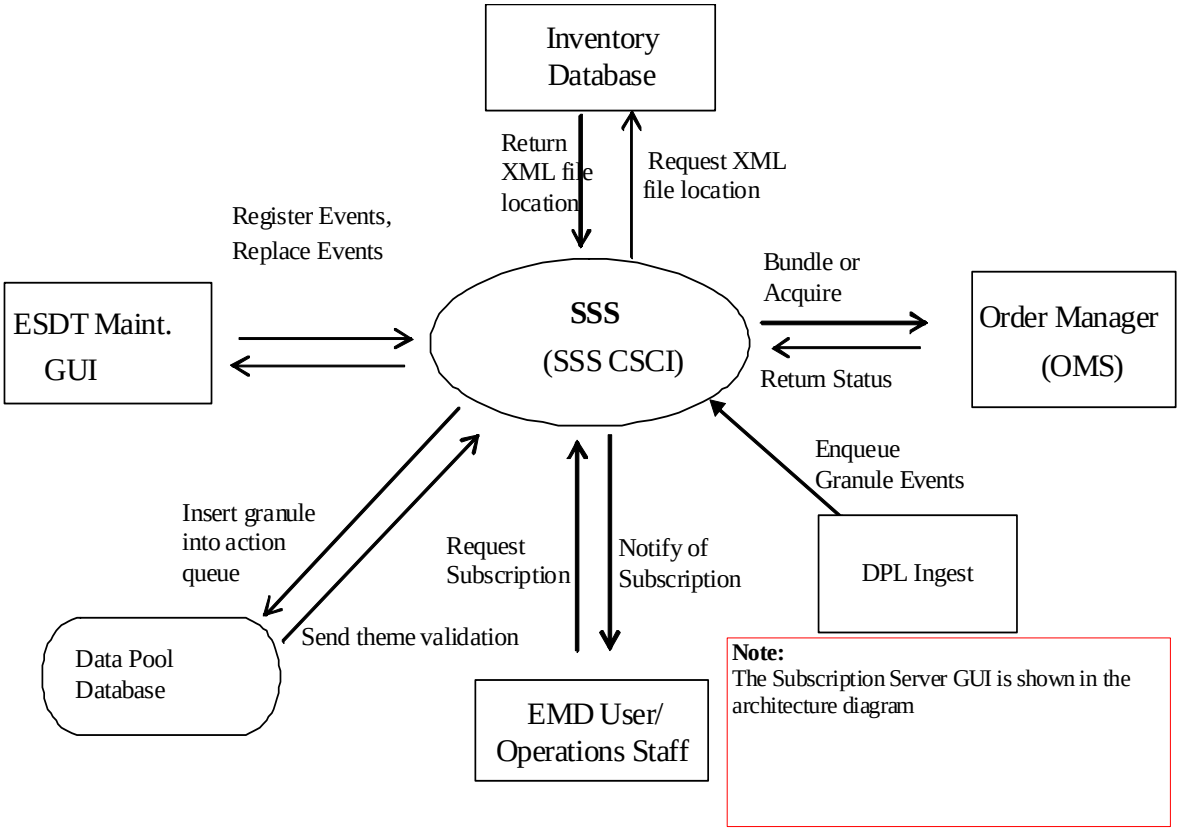

Figure 4.9-1. Spatial Subscription Server Context Diagram

Table 4.9-1. Subscription Server Interface Events

Event	Interface Event Description
Request XML file location	The **Inventory database** contains information about new granules, including the location of their XML metadata files. The SSS event driver queries the Inventory database for this information.
Notify of Subscription	The SSS CSC sends email notification to the **EMD User** when the subscribed event occurs, provided that a notification action was requested in the subscription.
Request Subscription	A subscriber (**EMD user** requests **Operations Staff** to create the subscription) sends information (ESDT and, optionally, acceptable metadata values) with the subscription, specifying one or more actions (e.g., acquire and/or notification) to be taken when the subscribed event occurs.
Return status	Status returned by a stored procedure to indicate whether or not the call succeeded.
Register Events	The **ESDT Maintenance GUI** inserts information about an Earth Science Data Type (ESDT) into the SSS database when an ESDT is installed into the system.
Enqueue Granule Events	DPL Ingest will enqueue new granule events via an SSS stored procedure call.
Replace Events	The **ESDT Maintenance GUI** modifies the SSS database when an ESDT is deleted from the system.
Bundle or Acquire	SSS notifies OMS, via a stored procedure call, when a granule has matched a subscription. If the subscription is bundled, i.e. associated with an OMS bundling order, then the granule is inserted into the appropriate OMS bundle. If the subscription is not bundled, then an acquire request is sent to OMS.
Insert granule into action queue	If a subscription has an associated Data Pool action, then SSS will insert a row into the Data Pool database action queue table, indicating that the granule that matched the subscription should be inserted into the Data Pool.
Send theme validation	If a subscription's Data Pool action is associated with a Data Pool theme, then the Data Pool will verify, via stored procedure call that the theme exists and is enabled for insert.

4.9.1 Spatial Subscription Server Architecture

Figure 4.9-2 is the Spatial Subscription Server architecture diagram. The diagram shows the events sent to the Spatial Subscription Server processes and the events the Subscription Server processes send to other processes.

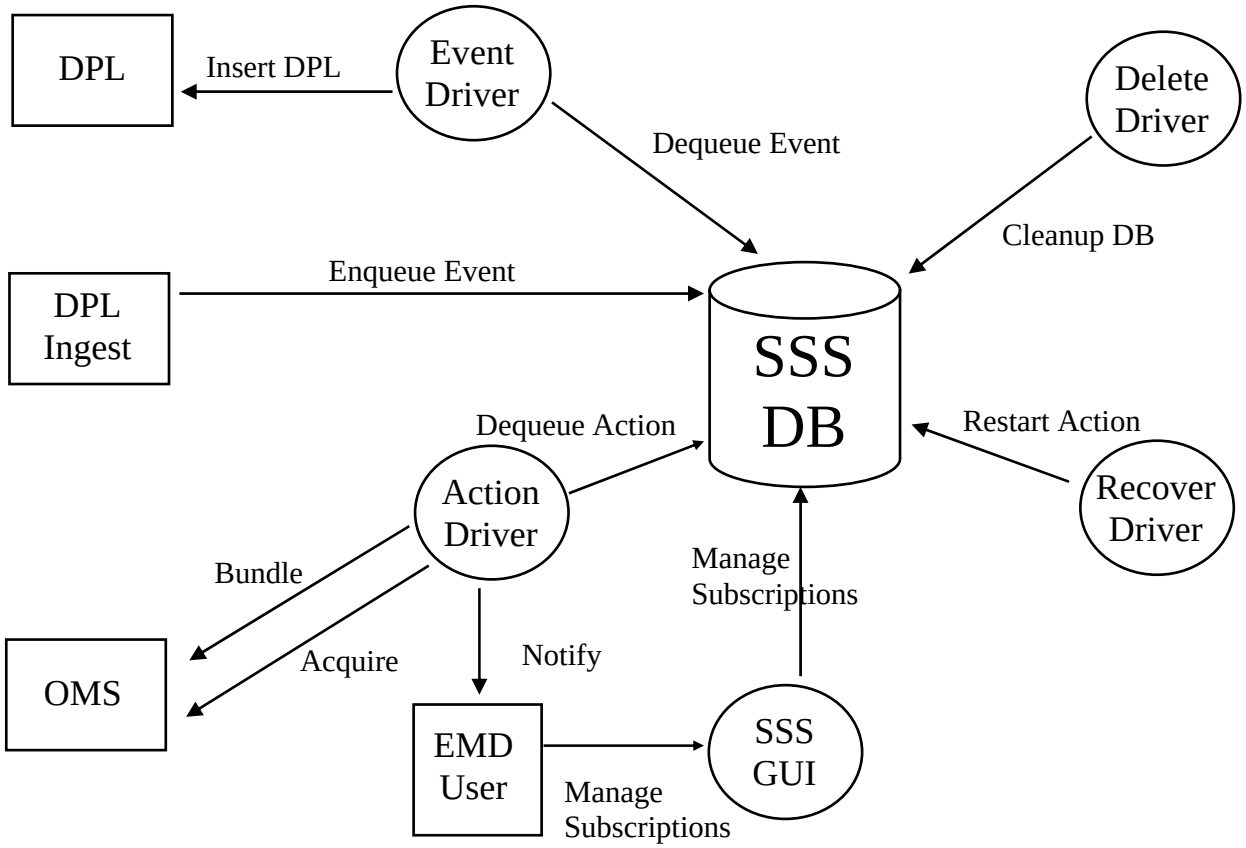

Figure 4.9-2. Spatial Subscription Server Architecture Diagram

Table 4.9-2 provides descriptions of the processes shown in the Spatial Subscription Server architecture diagram.

Table 4.9-2. Spatial Subscription Server Processes

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcNbSubscribedEventDriver (“Event Driver”)	Server	OMSHW	Developed	The SSS event driver dequeues events and matches them with active subscriptions. Information about matched subscriptions is placed in the action queue. If a matched subscription has a Data Pool action, the event driver inserts information into the Data Pool database.
EcNbActionDriver (“Action Driver”)	Server	OMSHW	Developed	The SSS action driver dequeues matched subscriptions and executes their associated actions (acquire or notification). An acquire directed to the Order Manager. If a subscription is bundled, then the granule that matched it is added to that bundle via an OMS interface.
EcNbDeleteRequestDriver (“Delete Driver”)	Server	OMSHW	Developed	The SSS delete driver dequeues from the delete request queue and cleans up database storage for the completed action or event.
EcNbRecoverDriver (“Recover Driver”)	Server	OMSHW	Developed	The SSS recover driver monitors the event and action queues for stalled events/actions and reenqueues them so that they will be tried again.
EcNbSubscriptionGUI (“SSS GUI”)	GUI	OMSHW	Developed	The SSS GUI provides an operator interface for submitting, updating and deleting subscriptions. It is also used for creating OMS bundling orders and for bundling subscriptions to bundling orders.
Sybase ASE	Server	ACMHW	COTS	The Sybase ASE is where the SSS database resides.

EMD Baseline Information System (EBIS) Document 920-TDx-001 (Hardware Design Diagram) provides descriptions of the HWCI, and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.9.1.1 Subscription Server Process Interface Descriptions

Table 4.9-3 provides descriptions of the interface events shown in the Subscription Server architecture diagram.

Table 4.9-3. Spatial Subscription Server Process Interface Events (1 of 2)

Event	Event Frequency	Interface	Initiated by	Event Description
Enqueue Event	Once per granule ingest	*Process:* ProcSubscribedEventEnqueue	*Process:* DPL Ingest	The stored procedure is called when a new granule is ingested by DPL Ingest.
Dequeue Event	Once per event	*Process:* ProcSubscribedEventDequeue	*Process:* EcNbSubscribedEventDriver	An event driver instance will dequeue up to 10 events from the event queue at one time. It will then process the events sequentially by getting the metadata for each granule and comparing it with the list of active subscriptions. If a subscription matches a granule event, information about the match is placed into the action queue.
Insert DPL	Once per event	*Process:* TrigInsEcNbDpEventDetails	*Process:* EcNbSubscribedEventDriver	When a granule event matches one or more subscriptions, at least one of which has an associated Data Pool action, the event driver will insert information about the granule (with subscription numbers) into the Data Pool database. A single insert per event is performed by an insert trigger on the table EcNbDpEventDetails.
Dequeue Action	Once per matched subscription	*Process:* ProcActionDequeue	*Process:* EcNbActionDriver	An action driver instance will dequeue up to 10 matched subscriptions from the action queue at one time. It will then process them sequentially by getting the actions for each subscription. If the subscription is bundled, then the granule is added to the current bundle for that bundling order via a stored procedure call to the OMS database. Otherwise, the action driver will initiate an acquire of the granule or send email notification to the user, depending on how the subscription was set up.

Table 4.9-3. Spatial Subscription Server Process Interface Events (2 of 2)

Event	Event Frequency	Interface	Initiated by	Event Description
Acquire	Once per matched subscription	*Process:* OmCreateNonBundlingOrder (OMS case)	*Process:* EcNbActionDriver	If a matched subscription has an associated acquire action, the action driver will initiate the acquire by a stored procedure call to OMS.
Bundle	Once per matched subscription	*Process:* OmInsertBundleRequest	*Process:* EcNbActionDriver	If a matched subscription is a bundled subscription, the action driver will send information about the granule to OMS via a stored procedure call.
Notify	Once per matched subscription	*Process:* mailx	*Process:* EcNbActionDriver	If a matched subscription has an associated notification action, the action driver will compose an email message and send it to the address specified in the subscription definition.
Restart Action	Once per action or event	*Process:* ProcActionReEnqueue, ProcSubscribedEventReEnqueue	*Process:* EcNbRecoverDriver	If an action or event appears to have stalled, i.e. did not run to completion based on evidence in the log tables, the recover driver will reenqueue the action or event in its appropriate queue.
Cleanup DB	Once per action or event	*Process:* ProcDequeueDeleteRequest, ProcDeleteProcessedSub, ProcDeleteProcessedEvent	*Process:* EcNbDeleteRequestDriver	The delete driver will clean up tables in the database based on entries in the delete request queue. Each entry in this queue corresponds to one action or one event.
Manage Subscrip-tions	Various	*Process:* EcNbSubscriptionGUI	*Process:* EcNbSubscriptionGUI	The SSS GUI allows a user to create, delete, edit or view subscriptions. Or to create, delete, edit or view bundling orders and bundle subscriptions to them.

4.9.1.2 Subscription Server Data Stores

Spatial Subscription Server uses the COTS software Sybase Adaptive Server Enterprise (ASE) for the storage of persistent data. The following is a brief description of the principal types of data contained in the database:

- **Attributes**: includes the ESDTs for which subscriptions can be created and the metadata attributes that can be used to qualify those subscriptions.
- **Subscriptions:** information about subscriptions that have been created for users, their associated qualifying expressions, and their associated actions.

- **Events**: information about newly arrived data granules, their metadata, and the subscriptions that match them.
- **Actions:** information about actions for matched subscriptions that need to be carried out, e.g. acquire or email notification.

Table 4.9-4 provides descriptions of the data found in the principal Sybase ASE data stores used by the Spatial Subscription Server. More detail on these and other data stores can be found in the Spatial Subscription Server Database Design and Schema Specifications for the EED Contract (311).

Table 4.9-4. Spatial Subscription Server Data Stores (1 of 2)

Data Store	Type	Functionality
EcNbEventDefinition	Attributes	Contains the list of events to which a user can subscribe.
EcNbEventMetadataAttrDef	Attributes	Contains the list of attributes which can be used to qualify a subscription.
EcNbEventAttrXref	Attributes	Cross-references subscribable events with the metadata attributes pertaining to them.
EcNbSubscription	Subscriptions	Contains the list of user subscriptions.
EcNbMatchingExpression	Subscriptions	Contains the list of expressions used to qualify subscriptions.
EcNbSubMatchExp_XREF	Subscriptions	Cross-references subscriptions with matching expressions (qualifiers).
EcNbSubMatchingExpInteger	Subscriptions	Contains the range of integer values used to qualify a subscription by an integer attribute.
EcNbSubMatchingExpFloat	Subscriptions	Contains the range of float values used to qualify a subscription by a float attribute.
EcNbSubMatchingExpString	Subscriptions	Contains the string values used to qualify a subscription by a string attribute.
EcNbSubMatchingExpDate	Subscriptions	Contains the range of date values used to qualify a subscription by a date attribute.
EcNbNoseMatchingExpressio n	Subscriptions	Contains the values used to qualify a subscription by orbit data.
EcNbSpatialMatchingExpress ion	Subscriptions	Contains the values used to qualify a subscription spatially.
EcNbActionDefinition	Subscriptions	Contains the list of actions associated with subscriptions.
EcNbOrderAction	Subscriptions	Contains detailed information about acquire actions associated with subscriptions.
EcNbNotificationAction	Subscriptions	Contains detailed information about email notification actions associated with subscriptions.
EcNbDpAction	Subscriptions	Contains detailed information about Data Pool actions associated with subscriptions.
EcNbSubscribedEventQueue	Events	Contains information about granules which have entered the system that could match user subscriptions.

Table 4.9-4. Spatial Subscription Server Data Stores (2 of 2)

Data Store	Type	Functionality
EcNbSubEventQueueLog	Events	A log of all operations performed on the subscribed event queue.
EcNbEventMetadataInteger	Events	Contains metadata values for integer attributes of granule events.
EcNbEventMetadataFloat	Events	Contains metadata values for float attributes of granule events.
EcNbEventMetadataString	Events	Contains metadata values for string attributes of granule events.
EcNbEventMetadataDate	Events	Contains metadata values for date attributes of granule events.
EcNbEventMetadataNose	Events	Contains metadata values for attributes of granule events relating to orbit data.
EcNbDpEventDetails	Events	Used by the event driver to process Data Pool actions.
EcNbEventTruth	Events	Used by the event driver as part of the matching algorithm between granule events and user subscriptions.
EcNbActionQueue	Actions	Contains information about subscriptions which have been matched with granule events.
EcNbActionQueueLog	Actions	A log of all operations performed on the action queue.
EcNbDistribution	Actions	Used by the action driver to suppress duplicate distribution of granules.
EcNbDeleteRequestQueue	Events, Actions	A list of actions and events that can be removed from the database.

4.10 Data Pool Subsystem Overview

The Data Pool is a large online archive of ECS data at each DAAC. Science, metadata (in xml format), and browse files (in jpg format) are stored in the public Data Pool.

Hidden directories in the Data Pool file systems (/datapool/<mode>/user/<fs>/.orderdata) are used as staging areas for all granules being inserted into the Data Pool, granules whose collections are configured hidden and for granules being ordered via the Order Management Subsystem (OMS).

The Data Pool subsystem consists of the following components and supporting utilities:

1. **Data Pool Insert**: inserts ECS data into the Data Pool. ECS data is copied from the ECS archive into the Data Pool, based on an ECS granule id. The Data Pool database inventory is updated for each granule inserted in the Data Pool. Data Pool Insert consists of six major subcomponents:
   a) the Data Pool Action Driver (DPAD): a C++ executable which schedules Data Pool insert actions based on a queue of Data Pool insert actions populated by the Spatial Subscription Server, the Batch Insert Utility, Data Pool Ingest, the Order Manager Server, or the Migration processes.
   b) the New (7.20) Data Pool Insert Utility (NDPIU), a java executable which manages the registration and publication of an ECS data granule into the Data Pool;
   c) the Data Pool Quick Server, a C++ executable which is installed on the ECS service hosts. The Quick Server is used by the DPAD to perform copy and checksum operations. It is also used by DPL Ingest and OMS to perform operations which are performed on ECS service hosts for load balancing reasons, or which cannot be performed on the local host due to lack of data access (mount points, etc.)
   d) the band extraction utility (bandtool), a C executable invoked by the DPAD, which extracts band information from HDF-EOS granules and stores the extracted information in a .BandHeader file in the temp area on the ESDT file system. The .BandHeader file is used by the NDPIU during granule registration. The bandtool is invoked only if the granule being inserted is from a collection eligible for conversion by the HDF-EOS to GeoTiff Conversion Tool (HEG);
   e) the jpeg extraction utility (hdf2jpeg), a C executable invoked by the NDPIU, which extracts browse images (jpeg or raster) from a browse hdfeos file on browse publication.
2. **Data Pool On-line Archive Cleanup and Validation**: The functionalities of EcDlCleanupDataPool.pl have been grouped and built into separate scripts for performance improvement. Additional validation between AIM database and Data Pool on-line archive is added as part of the on-line archive capability.
   a) **EcDlCleanupFilesOnDisk.pl** a perl utility, which reports and fixes inconsistencies between Data Pool directories and the database, namely orphan/phantom validation and orphan cleanup.

b) **EcDlCleanupGranules.pl**: a perl utility, which cleans granules from Data Pool on-line archive and the database.

c) **EcDlInventoryValidationTool.pl**: a perl utility, which identifies discrepancies between AIM and DPL inventories.

d) **EcDlLinkCheck.ksh**: a korn shell script, which provides report on invalid soft links in Data Pool on-line archive and optionally remove them.

e) EcDlXcu.pl: a perl utility, which checks for corruption of XML files in the Data Pool.

3. **Data Pool Web Access (EcDlWebAccess)**: a java-based web application, which runs with the apache web server and related COTS. The Data Pool Web Access application allows end-users to perform drill-down searches for Data Pool data, to view metadata and browse images online, and to convert and/or order Data Pool data.
4. **Data Pool Maintenance GUI (EcDlDpm)**: a perl-based web GUI that allows DAAC operations staff to monitor Data Pool insert activity and to control the Data Pool configuration.
5. **Data Pool Access Statistics utilities**: perl utilities which parse firewall ftp logs (EcDlRollupFwFtpLogs.pl) and Data Pool Web Access custom code logs (EcDlRollupWebLogs.pl) for accesses to the Data Pool directories, and then roll up access information for storage in the Data Pool database.
6. **Data Pool FTP Server**: customized wu-ftp daemon, which supports ftp access to Data Pool directories and also provides a checksum-on-download service.
7. **Data Pool Update Granule Expiration utility (EcDlUpdateGranule.pl)**: a perl utility, which allows operations staff to update the Data Pool expiration date and retention priority for specified Data Pool granules.
8. **Data Pool Batch Insert Utility (EcDlBatchInsert.pl-being deprecated)**: a perl utility, which allows operations staff to queue ECS and non-ECS data for insert into the Data Pool.
9. **Data Pool Most Recent Insert Utility (EcDlMostRecentInsert.pl)**: a perl utility, which creates files at the file system and data collection level of the Data Pool directory structure which contain information about granules recently inserted into the Data Pool at those levels.
10. **Data Pool Collection Remapping Utility (EcDlRemap.pl)**: a perl utility, which allows DAAC operations staff to remap all data in a Data Pool collection directory from one higher level collection group directory to another.
11. **Data Pool Move Collection Utility (EcDlMoveCollection.pl)**: a perl utility, which allows DAAC operations staff to move a Data Pool collection from one file system to another.
12. **Data Pool Density Map Utility (EcDlDensityMapUtility.pl)**: a perl utility, which calculates spatial density map information about Data Pool collections and stores this information in the Data Pool database. This utility normally runs as a cron job.

13. **Data Pool Statistics Table Population Utility (EcDlPopulateStatTables.pl)**: a perl utility, which populates tables in the Data Pool database which maintain counts of granules by collection and collection group, for use by the Web Access drill down web pages. This utility normally runs as a cron job.
14. **Data Pool Hidden Scrambler Utility (EcDlHiddenScrambler.pl)**: a perl utility, which creates new names for specified hidden directories, saves these names, renames the existing hidden directories, and updates existing FTP Pull links that point to the previous hidden directories to point to the corresponding renamed directory.
15. **Data Pool Database (DataPool[_<MODE>])**: a Sybase database which stores Data Pool inventory and configuration information.
16. **Data Pool Checksum Verification Utility (being deprecated)**: A java-based stand-alone utility which can verify the integrity of files in the Data Pool using checksums. The utility could be set up as a background process as well as run on-demand by the DAAC operator to verify checksum values for a particular set of files.
17. **Data Pool Checksum Verification Server**: A C++ based stand-alone utility which can verify the integrity of files in the Data Pool using checksums. It provides resource management and load balancing to achieve optimal checksum throughput when the system is under load and normal operation. It replaces "Data Pool Checksum Verification Utility".
18. **Data Pool Restore On-line Archive From Tape Utility**: A java-based stand-alone utility which performs bulk repairs of the on-line archive, especially in the case of serious disk errors or a loss of a Data Pool file system. It can also be used to restore the integrity of granules which have files missing or corrupted, or missing links.
19. **Data Pool Restore AIM Tape Archive From On-line Archive Utility**: A java-based stand-alone utility which provides bulk repair as well as individual science granules in the AIM tape archive by replacing science granules with their copy from Data Pool On-line archive.
20. **Data Pool Publish Utility**: A java-based utility which allows operations staff to submit ECS insert actions for publication or registration, optionally with load control. The utility has been enhanced to include theme option and nonECS insert. It includes all the capability from "Batch Insert Utility" and has more of its own.
21. **Data Pool Unpublish Utility**: A java-based utility which moves granules from the public Data Pool on-line archive to the hidden Data Pool on-line archive.
22. **Data Pool Cloud Cover Utility**: A java-based utility which enables the DAAC to update/populate/remove the cloud cover information stored in the DPL database for public science granules, based on the cloud cover source that is associated to the desired collection via the DPL maintenance GUI.
23. **Data Pool Ftp Service (EcDlFtpService)**: A C++ based stand-alone SOAP web service running on the ECS service hosts. It accepts Ftp service requests, processes the requests and sends the results back to the web service client. This service has ability to keep the FTP connections in the pool and reuse them later. It is used by DPL Ingest, OMS and BMGT to perform Ftp Transfer operations.

4.10.1 Data Pool Subsystem Context

Figure 4.10-1 is the Data Pool Subsystem context diagram. The diagram shows the interaction of the Data Pool Subsystem with other EED subsystems. Table 4.10-1 provides descriptions of the interface events shown in the Data Pool Subsystem context diagram.

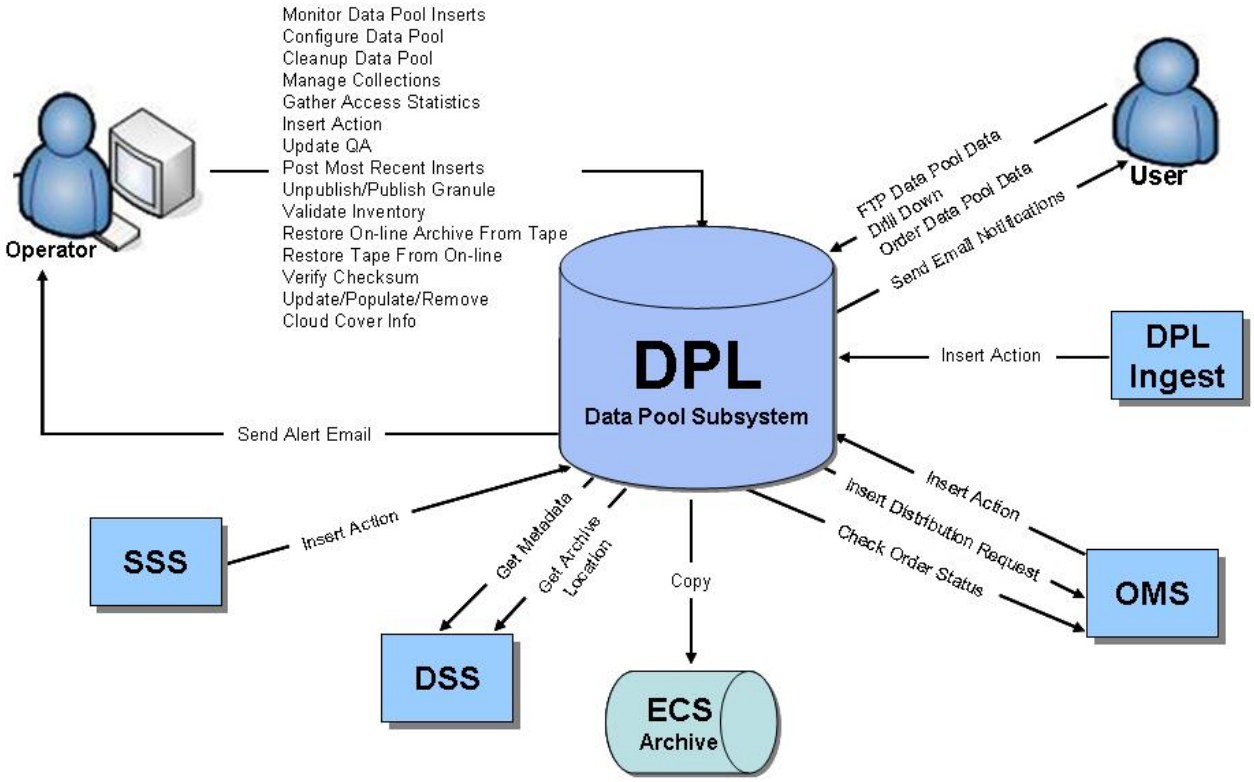

Figure 4.10-1. Data Pool Subsystem Context Diagram

Table 4.10-1. Data Pool Subsystem Interface Events (1 of 3)

Interface Event	Interface Event Description
Send Alert Email	The Data Pool Action Driver sends an alert email to a configured email address to notify operators of problems with an ECS Service Host, an archive file system, or a Data Pool file system.
Monitor Data Pool Inserts	The operator uses the Data Pool Maintenance GUI to monitor the queue of Data Pool inserts and to monitor the active insert processes.
Configure Data Pool	The operator uses the Data Pool Maintenance GUI to set values of Data Pool configuration parameters, and to define Data Pool entities such as themes and compression algorithms.

Table 4.10-1. Data Pool Subsystem Interface Events (2 of 3)

Interface Event	Interface Event Description
Cleanup Data Pool	The operator runs the Data Pool Cleanup utilities to clean specified granules out of the Data Pool on-line archive, and to identify and cleanup granules, which are orphans (on Data Pool disk but not in the database) or phantoms (in the Data Pool database but not on disk), or invalid links(soft links point to invalid files).
Manage Collections	The operator uses the Data Pool Maintenance GUI to add, remove, or change specifications for Data Pool collections. The operator uses the Remap Collection utility to map a collection from one collection group to another. The operator uses the Move Collection utility to move a collection from one file system to another.
Gather Access Statistics	The operator uses the access statistics rollup scripts for the firewall ftp and web access logs to gather statistics about end user access to data pool files, and to store those statistics in the Data Pool database.
Insert Action	The operator uses the Batch Insert utility or Data Pool Publish utility to insert historical data from the ECS archive into the Data Pool.
Update Granule Expiration	The operator uses the Update Granule Expiration utility to update the expiration date or retention priority for a Data Pool granule or set of granules.
Update QA	The operator uses the QA Update utility to propagate updates of QA information from the Inventory to the Data Pool.
Post Most Recent Inserts	The operator uses the Most Recent Inserts utility to post information about recent Data Pool Inserts to the Data Pool ftp directories.
Unpublish Granule	The operator uses the Data Pool Unpublish Utility to move granules from public Data Pool on-line archive to hidden Data Pool on-line archive.
Validate Inventory	The operator uses the Data Pool Inventory Validation Utility to identify discrepancies between AIM and DPL inventories.
Restore On-line Archive From Tape	The operator uses the Restore On-line Archive From Tape Utility to restore the integrity of granules to on-line archive from tape.
Restore Tape From On-line Archive	The operator uses the Restore Tape From On-line Archive Utility to restore the integrity of granules to tape from on-line archive.
Verify Checksum	The operator uses the Data Pool Checksum Verification Server to verify the integrity of the granules in Data Pool on-line archive.
Update/Populate/Remove Cloud Cover Information	The operator uses DataPool Cloud Cover Utility to update/populate/remove the cloud cover information for DPL granules.
FTP Data Pool data	The end user uses the customized WU-FTP service to download Data Pool data.
Drill Down	The end user uses the Web Access web pages to perform searches for Data Pool data.
Order Data Pool data	The end user uses the Web Access web pages to order Data Pool data for ftp or media distribution. The end user may choose to convert, reformat, or subset the data using the HDF-EOS to GeoTiff Conversion Tool (HEG).

Table 4.10-1. Data Pool Subsystem Interface Events (3 of 3)

Interface Event	Interface Event Description
Send Email Notification	The DPL subsystem (Web Access component) sends email to the end user indicating that the user's Data Pool order has been submitted. Email is sent by the WebAccess component only for downloads without HEG conversion, and only if the user requests email. (OMS sends order acknowledgement and distribution notice emails).
Insert Action	The Data Pool Ingest subsystem inserts a Data Pool insert action into the Data Pool Insert Action Queue (DlInsertActionQueue) for granules which are configured to be published in the Data Pool.
Insert Action	The OMS subsystem inserts a Data Pool insert action into the Data Pool Insert Action Queue (DlInsertActionQueue) for granules to be staged to the Data Pool for ECS distribution requests.
Insert Distribution Request	The DPL subsystem (WebAccess component) inserts distribution requests in the OMS database for Data Pool orders placed using the Data Pool Web Access web pages.
Check Order Status	The DPL subsystem (WebAccess component) checks status of orders in the MSS database.
Copy	The DPL subsystem copies data from the ECS Archive to the appropriate Data Pool file system.
Get Archive Location	The DPL subsystem looks up archive location information in the Inventory database, for granules which will be copied from the ECS archive to the Data Pool.
Get Metadata	The DPL subsystem gets metadata about ECS granules (QA, PH, etc.,) from the Inventory database, and uses this metadata to store corresponding metadata in the Data Pool database and to create an xml metadata file on Data Pool disk. The DPL subsystem gets metadata path about ECS granules (SCIENCE) from Inventory database, and uses this path to get xml files from small archive to DataPool filesystem.
Insert Action	The Spatial Subscription Server subsystem inserts Data Pool insert actions in the Data Pool Insert Action Queue (DlInsertActionQueue) for granules which are being inserted into the ECS inventory for which a Data Pool insert subscription is placed. Data Pool insert subscriptions are qualified subscriptions (unqualified Data Pool insert subscriptions have been replaced by DPL Ingest configuration of ESDTs for public Data Pool insert.)

4.10.2 Data Pool Hardware Context

Figure 4.10-2 is the Data Pool hardware context diagram. The diagram shows the interaction of the Data Pool custom code and COTS (in *italics*) with EED hardware components.

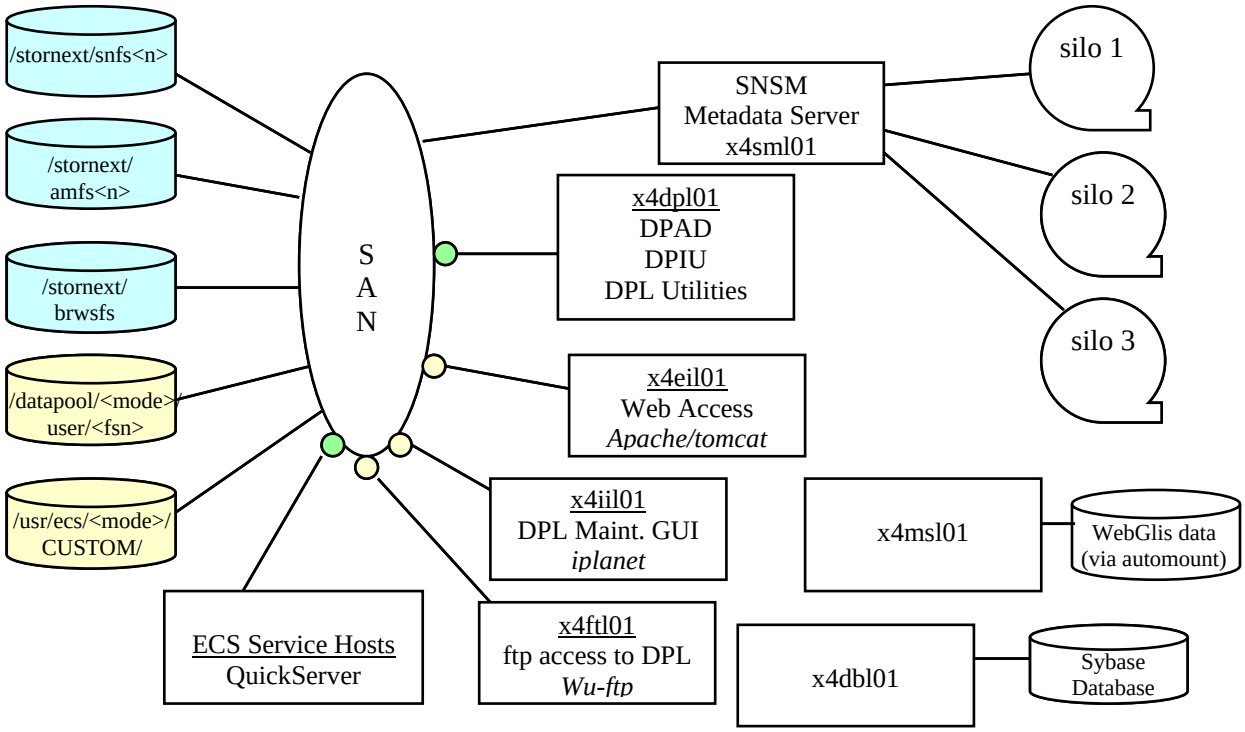

Figure 4.10-2. Data Pool Hardware Context

4.10.3 Data Pool Insert CSCI Functional Overview

ECS granules are inserted into the Data Pool via a two-step process. The first step, registration, involves storing basic inventory information about the granule, needed by EED custom code applications, in the Data Pool database, and copying the granule to a "hidden" directory structure (datapool/<mode>/user/<filesystem>/.orderdata) in the Data Pool. The second step, publication, occurs only for granules which belong to collections configured to be placed in the public Data Pool, where they are available for web access and anonymous ftp download. During the publication step, additional inventory information needed to support web access to public granules is stored in the Data Pool database, and the granule is moved from the Data Pool hidden directory structure to the public directory structure, where it can be accessed via anonymous ftp.

A functional overview of the two-step Data Pool Insert process for ECS granules is shown below in two diagrams. The first diagram (Figure 4.10-3) shows the process for registration of a granule in the Data Pool. The second diagram (Figure 4.10-4) shows the process for publication of a granule in the Data Pool.

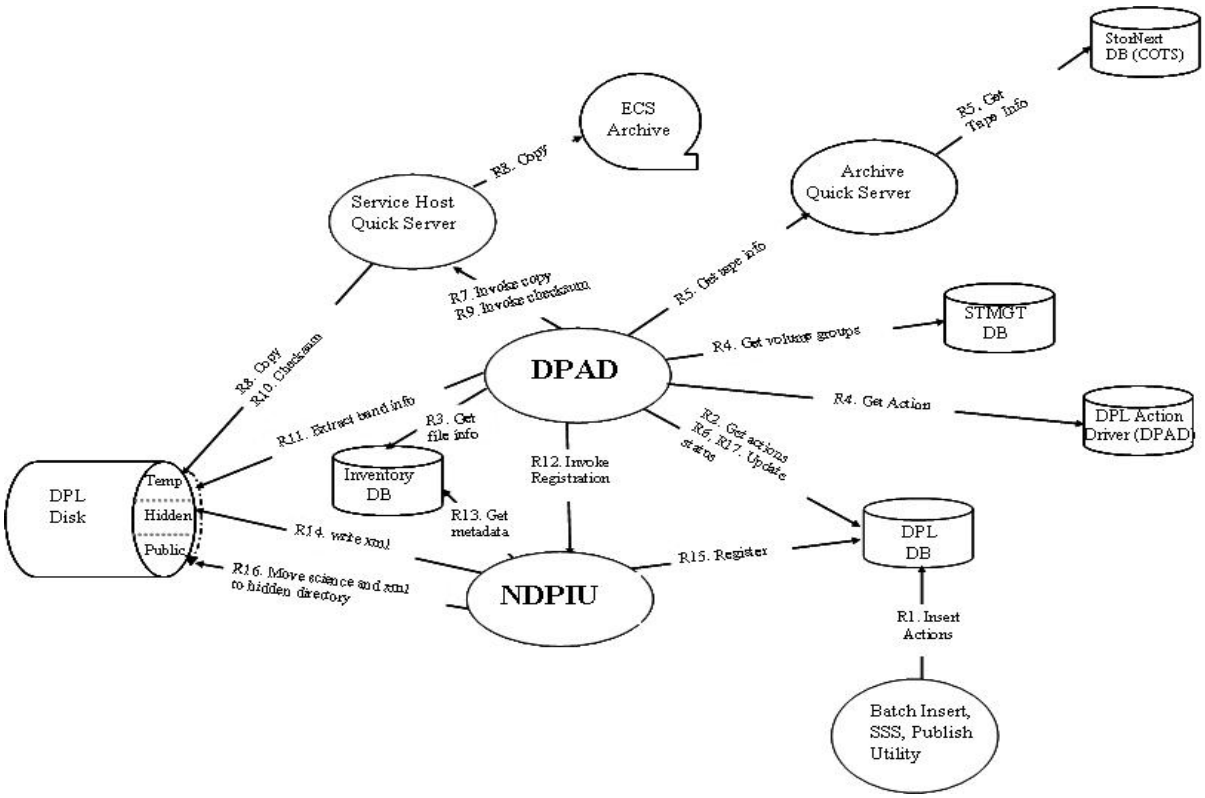

Figure 4.10-3. Data Pool Insert CSCI Architecture Diagram – Registration

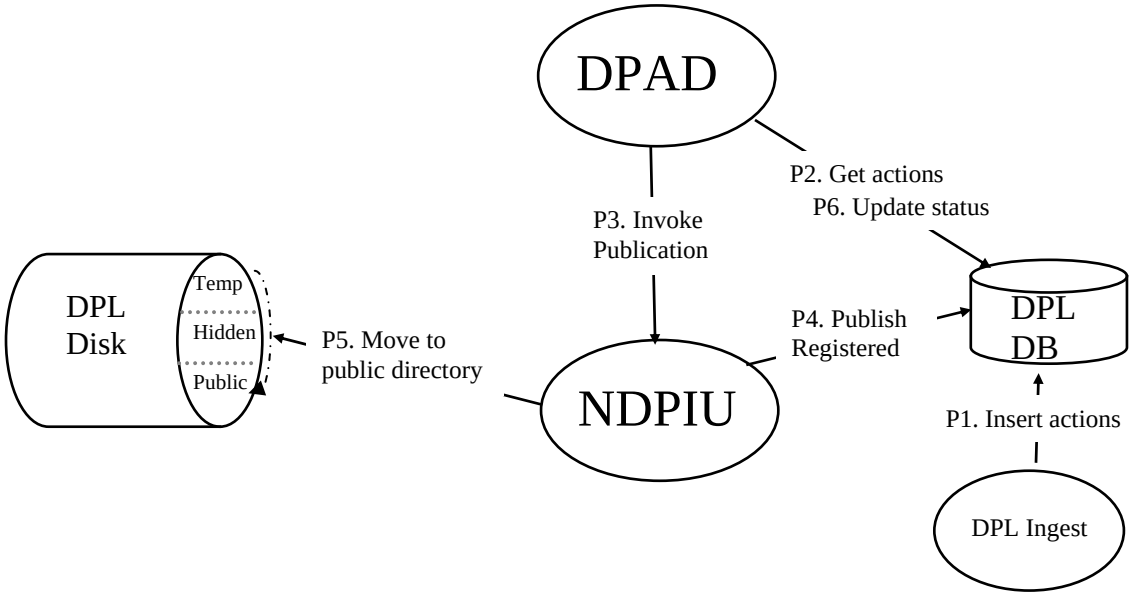

Figure 4.10-4. Data Pool Insert CSCI Architecture Diagram - Publication

There are five use cases for the Data Pool Insert process, one for each ECS process or component which requests insertion of a granule into the Data Pool. These use cases, and their relationship to the registration and publication steps, is shown in Table 4.10-2.

Table 4.10-2. Use Cases for Data Pool Insert

Requestor	Context	Data Pool Insert processes
OMS	stages granules in the hidden Data Pool for distribution	Registration (events R1 – R17 in Table 4.10-3 for data type such as QA, PH, DAP, etc; events R1-R12 and R15-R17 in Table 4.10-3 for science granules).
Data Pool Ingest	requests publication of a granule in the Data Pool after the granule has been ingested and archived	Publication (events P1 – P6 in Table 4.10-4).
Batch Insert Utility	queues existing ECS and non-ECS granules for insertion into the public Data Pool	Registration Publication (events R1-R16 in Table 4.10-3 for data type such as QA, PH, DAP, etc; events R1-R12 and R15-R16 in Table 4.10-3 for science granules, and P3-P6 in Table 4.10-4)
Publish Utility	Queues existing ECS granules for publication or registration in Data Pool on-line archive	Registration Publication (events R1-R16 in Table 4.10-3 for data type such as QA, PH, DAP, etc; events R1-R12 and R15-R16 in Table 4.10-3 for science granules, and P3-P6 in Table 4.10-4)
Spatial Subscription Server	queues granules for insertion into the public Data Pool as a result of a qualified subscription with a Data Pool insert action	Registration Publication (events R1-R16 in Table 4.10-3 for data type such as QA, PH, DAP, etc; events R1-R12 and R15-R17 in Table 4.10-4 for science granules, and P3-P6 in Table 4.10-4)

Note that Data Pool Ingest also stages granules in the hidden Data Pool during granule ingest. That process is somewhat different than the Registration process described below, in that it uses a different invocation of the NDPIU and does not involve the DPAD. Data Pool Ingest staging of granules in the hidden Data Pool is documented in the Data Pool Ingest chapter of this document.

Table 4.10-3 provides a process description for each of the major custom code components of the Data Pool insert process. Table 4.10-4 describes the interface events among the Data Pool insert process components for registration and publication.

Table 4.10-3. Data Pool Insert CSCI Process Description

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcDlActionDriver	Server	DPLHW	Developed	EcDlActionDriver (DPAD)is a C++ server that is responsible for dispatching Data Pool insert actions for ECS granules from the insert action queue in the Data Pool database (DlInsertActionQueue) as well as performing registration or publication of ECS granules(possibly involving copy, checksum, band extraction operations).
EcDlInsertUtility	Java utility	DPLHW	Developed	EcDlInsertUtility (NDPIU) is a java executable that is invoked by the EcDlActionDriver to register and publish ECS granules in the Data Pool (populate DPL database and move files to hidden or public Data Pool). One copy of the EcDlInsertUtility is invoked for each ECS granule to be inserted in the Data Pool.
EcDlQuickServer	Server	ACMHW, DIPHW, DRPHW, SPRHW, MSSHW, AITHW, CSSHW, DMGHW, DPSHW, INTHW, DPLHW, OMSHW	Developed	The EcDlQuickServer (Service Host Quick Server) is a C++ server which performs CPU-intensive operations, such as copy and checksum, on ECS service hosts.
hdf2jpeg	Utility	DPLHW	Developed	Java utility that extracts jpgs from an HDFEOS granule.
bandtool	Utility	DPLHW	Developed	C utility that extracts band information from an HDFEOS granule.

Table 4.10-4. Data Pool ECS Insert CSCI Process Interface Events (1 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
R1. Insert Action	One per granule inserted in Inventory which qualifies for existing subscription with Data Pool Insert action	Database: DataPool (DlInsertActionQueue)	Trigger: TrigInsEcNbDpEventDetails.sql	When a granule is inserted into Inventory which matches an existing subscription with Data Pool Insert action, the trigger inserts a row into the DlInsertActionQueue in the Data Pool database with actionSource = null.
R1. Insert Action	One per granule in Syn IV order placed through Order Manager	Database: DataPool	Process: OmServer Stored Proc: OmInsDPLAction	When a granule is ordered in Syn IV mode via the Order Manager, OMS inserts a row into the DlInsertActionQueue in the Data Pool database, with actionSource = O.
R1. Insert Action	One per granule in input file for the Batch Insert utility.	Database: DataPool	Utility: EcDlBatchInsert.pl	For each valid granule in its input file, the Batch Insert Utility inserts a row into the DlInsertActionQueue in the Data Pool database, with actionSource = B or R.
R1.Insert Action	One per granule in input file, or on the command line for the Batch Insert utility.	Database: DataPool	Process: EcDlPublishUtility	For each valid granule in its input file or on the command line, the Publish Utility inserts a row into the DlInsertActionQueue in the Data Pool database, with actionSource = B or R.
R2. Get Action	Continuously, as long as there are actions in DlInsertActionQueue with status = null or status = RETRY. If no actions, once per configured time interval (IdleSleep in DlConfig)	Database: DataPool	Process: EcDlActionDriver (DPAD)	The DPAD gets batches of actions (with status = null or status = RETRY) from the DlInsertActionQueue.

Table 4.10-4. Data Pool ECS Insert CSCI Process Interface Events (2 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
R3. Get file info	Once per file per ECS granule to be inserted	Database: Inventory (EcInDb) (DsMdUserDataFile)	Process: EcDlActionDriver (DPAD)	The DPAD gets file name information for each file in the granule from the Inventory database.
R4. Get volume groups	Once per ECS granule to be inserted	Database: Inventory (DsStVolumeGroup)	Process: EcDlActionDriver (DPAD)	The DPAD gets the name of the open volume group for the granule's collection (shortname, versionid) from the Inventory database.
R5. Get tape info	Once per file per ECS granule to be inserted	Process: Java Quick Server Database: COTS StorNext metadata database	Process: EcDlActionDriver (DPAD)	The DPAD calls the Java Quick Server, which runs on the StorNext (COTS) metadata server, to retrieve tape label information for the granule files, based on the volume group information from the Inventory database.
R6. Update status	Once per ECS granule to be inserted	Database: Data Pool (DlInsertActionQueue, DlActiveInsertProcesses)	Process: EcDlActionDriver (DPAD)	The DPAD updates the status of the insert in the Data Pool database.
R7. Invoke copy	Once per file per ECS granule	Process: EcDlQuickServer	Process: EcDlActionDriver (DPAD)	The DPAD chooses a QuickServer on an ECS Service Host to perform the file copy operation. The Service Host is chosen based on availability.
R8. Copy	Once per file per ECS granule	Storage Device: Data Pool disk (managed by COTS StorNext Storage Area Network) Storage Device: ECS Archive tape or cache (managed by COTS StorNext)	Process: EcDlQuickServer DlAdCopy	The QuickServer, running on an ECS Service Host, uses the DlAdCopy to copy the science file and its associated metadata xml file from the ECS Archive (tape or cache) to the Data Pool file system associated with the granule's collection.
R9. Invoke checksum	Once per file per ECS granule	Process: EcDlQuickServer	Process: EcDlActionDriver (DPAD)	The DPAD chooses a QuickServer on an ECS Service Host to perform the file checksum operation. The Service Host is chosen based on availability.

Table 4.10-4. Data Pool ECS Insert CSCI Process Interface Events (3 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
R10. Checksum	Once per file per ECS granule	Storage Device: Data Pool disk (managed by COTS StorNext Storage Area Network) Storage Device: ECS Archive tape or cache (managed by COTS StorNext)	Process: EcDlQuickServer	The QuickServer, running on an ECS Service Host, checksums the science file and the associated metadata xml file if needed on the Data Pool file system (temp directory).
R11. Extract band info	Once per ECS science granule, where the collection is enabled for HEG conversion (i.e., convertEnabledFlag is on for the collection)	Storage Device: temp directory in Data Pool file system	Process: EcDlActionDriver (DPAD) bandtool	The DPAD uses the bandtool utility to extract band information from the science granule, and writes band information to a temporary file in the Data Pool file system
R12. Invoke registration	Once per ECS science granule	Process: EcDlInsertUtility (NDPIU)	Process: EcDlActionDriver (DPAD)	The DPAD invokes an instance of the NDPIU from a pool to perform the granule registration.
R13. Get metadata	Once per ECS QA, PH, DAP and browse granule	Database: Inventory (EcInDb)	Process: EcDlInsertUtility (NDPIU),	The NDPIU gets metadata about the ECS QA, PH, DAP, brose granule from the Inventory database.
R14. Write xml	Once per ECS QA, PH, DAP and browse granule	Storage Device: temp directories in Data Pool file system	Process: EcDlInsertUtility (NDPIU)	The NDPIU writes the xml metadata file for the granule to the temp directory on the Data Pool file system.
R15. Register	Once per ECS granule	Database: Data Pool	Process: EcDlInsertUtility (NDPIU)	The NDPIU populates basic tables in the Data Pool database with inventory information about the granule.
R16. Move science and xml to hidden directory	Once per data file and xml file per ECS granule	Storage Device: temp and hidden directories in Data Pool file system	Process: EcDlInsertUtility (NDPIU)	The NDPIU moves the science file(s) and xml file for the granule from the temp directory in the Data Pool file system to the appropriate hidden directory (under /.orderdata).

Table 4.10-4. Data Pool ECS Insert CSCI Process Interface Events (4 of 4)

Event	Event Frequency	Interface	Initiated By	Event Description
[R17. Update status]	Once per insert request	Database: Data Pool	Process: EcDlActionDriver (DPAD)	The DPAD updates the insert request status in the DlInsertActionQueue.
[P1. Insert action]	Once per publication request	Database: Data Pool (DlInsertActionQueue)	Process: Data Pool Ingest Processing	The DPL Ingest Processing server places a request for granule publication in the DlInsertActionQueue.
[P2. Get actions]	Continuously, as long as there are actions in DlInsertActionQueue with status = null or status = RETRY. If no actions, once per configured time interval (IdleSleep in DlConfig)	Database: DataPool	Process: EcDlActionDriver (DPAD)	The DPAD gets batches of actions (with status = null or status = RETRY) from the DlInsertActionQueue.
P3. Invoke publication	Once per ECS granule to be published in the Data Pool	Process: EcDlInsertUtility (NDPIU)	Process: EcDlActionDriver (DPAD)	The DPAD invokes an instance of the NDPIU from a pool to perform the granule publication.
P4. Publish registered granule	Once per ECS granule to be published in the Data Pool	Database: Data Pool	Process: EcDlInsertUtility (NDPIU)	The NDPIU populates additional tables in the Data Pool database with inventory information needed to support web access to the granule.
P5. Move to public directory	Once per ECS granule to be published in the Data Pool	Storage Device: hidden and public directories in Data Pool file system	Process: EcDlInsertUtility (NDPIU)	The NDPIU moves the data file(s) and xml file for the granule from the hidden directory in the Data Pool file system to the appropriate public directory
P6. Update status	Once per ECS granule to be published in the Data Pool	Database: Data Pool	Process: EcDlActionDriver (DPAD)	The DPAD updates the insert request status in the DlInsertActionQueue to the final request state, and removes the insert request from the DlActiveInsertProcesses table.

4.10.4 WebAccess CSCI Functional Overview

Data Pool Web Access (EcDlWebAccess) is a java-based Web application that runs on a Web application server with related COTS software. The Data Pool Web Access application allows end-users and operators to interface with Data Pool data on the Sybase server. It allows end-users to perform drill-down searches for Data Pool data, to view metadata and browse images online, to request conversion of Data Pool data and to order data via ftp pull, ftp push or physical media orders.

Figure 4.10-5 is the WebAccess CSCI architecture diagram. The diagram shows the events sent to the WebAccess CSCI processes and the events the WebAccess CSCI processes send to other processes.

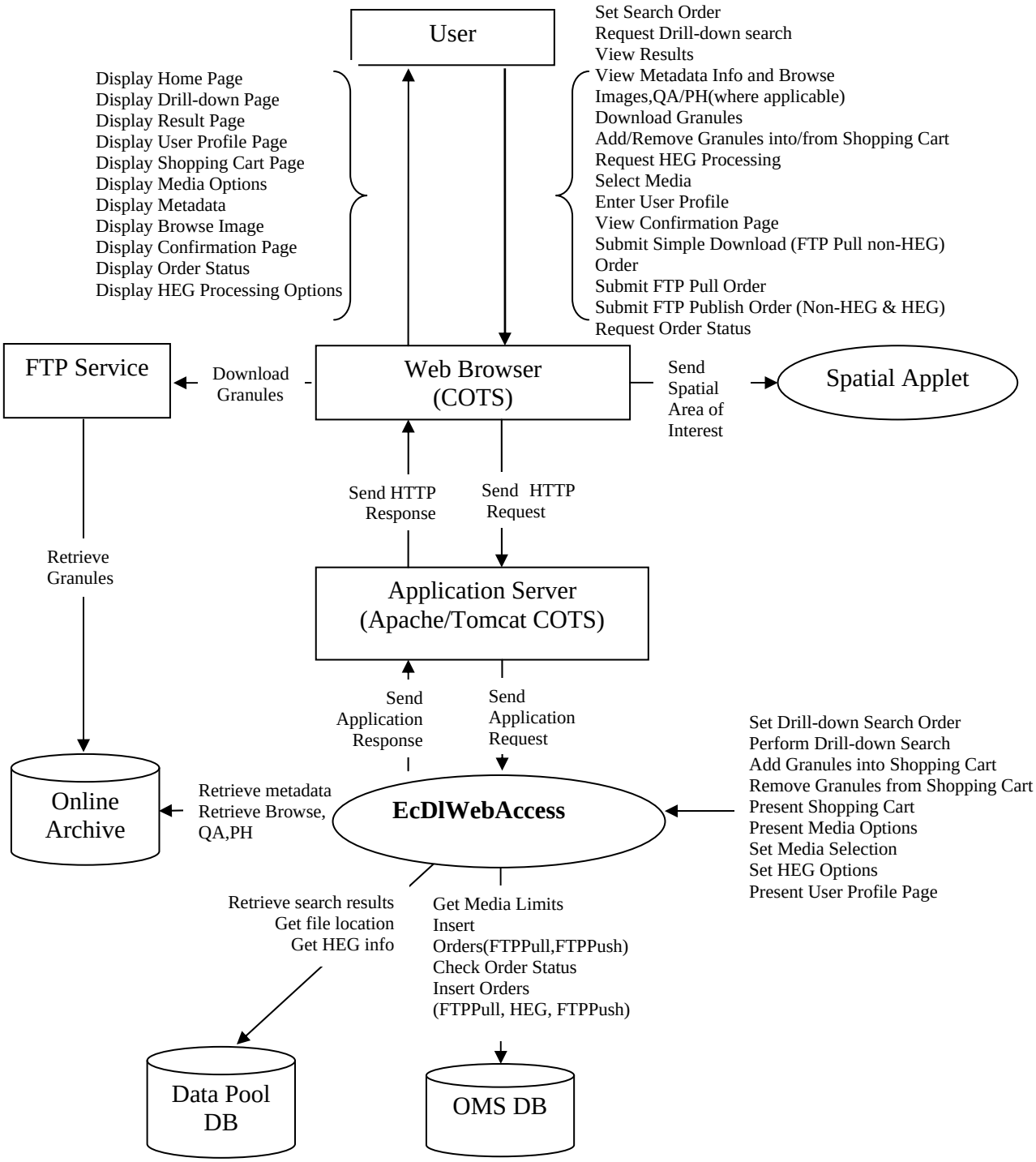

Figure 4.10-5. WebAccess CSCI Architecture Diagram

4.10.4.1 WebAccess Process Descriptions

Table 4.10-5 provides descriptions of the processes shown in the WebAccess architecture diagram.

Table 4.10-5. WebAccess CSCI Process Description

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcDlWebAccess	Web App	INTHW	Developed	EcDlWebAccess is a web application that runs inside a web server. It provides user friendly web pages which allow users to search and retrieve data from Data Pool, view granules and related granule products. Once the list of granules has been retrieved, the user can then order granules through ftp-pull, ftp-push or physical media orders. Users may also request HEG conversions for granules in an order.
Application Server	Server	INTHW	COTS	Application Server hosts the EcDlWebAccess web application.
Spatial Applet	Applet	INTHW	Developed	The spatial applet allows users to interactively select desired area of interest on a map of the earth. The spatial applet uses WebGlis, a COTS product from USGS, to produce the map.
Web Browser	Browser	ACMHW, DIPHW, DRPHW, SPRHW, MSSHW, AITHW, CSSHW, DMGHW, DPSHW, INTHW, DPLHW, OMSHW	COTS	The Web Browser loads and displays EcDlWebAccess web pages.

4.10.4.2 WebAccess Process Interface Descriptions

Table 4.10-6 describes the interface events among the WebAccess CSCI processes.

Table 4.10-6. WebAccess CSCI Process Interface Events (1 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Set Search Order	One Per Search Sequence	Process: Web Browser (COTS)	User	The user configures the presentation of drill-down sequence following certain rules. The parameters are: Data Group, Theme, Data Set, Date, Time, Spatial, Cloud Cover, Day/Night Flag and Science QA.
Request Drill-down Search	One Per Request	Process: Web Browser (COTS)	User	The User specifies search criteria on each Drill-Down page (Theme/Group/ESDT, Temporal, TimeOfDay, AreaOfInterest, Cloud Cover etc.).
View Results	One Per Request	Process: Web Browser (COTS)	User	The User Clicks “Get the Granules” link on the drill-down page, or the drill-down search attributes have been exhausted.
View Metadata Info and Browse Images	One Per Request	Process: Web Browser (COTS)	User	The User chooses to view the metadata information and/or browse image of a granule.
Download Granules	One Per Request	Process: Web Browser (COTS) Wu-FTP (COTS)	User	The User downloads the granules by initializing a ftp request.
Add/Remove Granules into/from Shopping Cart	One or Many Per Order Request	Process: Web Browser (COTS)	User	The User adds/removes granules into/from shopping cart.
Request HEG Processing	One or Many Per Order Request	Process: Web Browser (COTS)	User	The User selects format, projection, projection parameters, spatial subsetting or band subsetting for the granules in the shopping cart.

Table 4.10-6. WebAccess CSCI Process Interface Events (2 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Select Media	One Per Order Request	Process: Web Browser (COTS)	User	The User selects one media option from the following: ftp-pull, ftp-push, CDROM and DVD.
Enter User Profile	One Per Order Request	Process: Web Browser (COTS)	User	The User enters user profile: name, email address, contact address, shipping address for a physical media order, ftp push related info for an ftp-push order.
View Confirmation Page	One Per Order Request	Process: Web Browser (COTS)	User	The User performs checkout and is presented with the detail of the orders.
Submit Simple Download (FTP-Pull Non-HEG, with ordering option of QA/PH where applicable) order	One Per Order Request	Process: Web Browser (COTS)	User	The User submits a simple download order that does not require HEG processing and is presented with data pool order Id, download links.
Submit FTP Pull HEG Order	One Per Order Request	Process: Web Browser (COTS)	User	The User submits an ftp pull HEG order and is presented with an OMS order id and order status.
Submit FTP Push Order (nonHEG and HEG, with ordering option of QA/PH where applicable)	One Per Order Request	Process: Web Browser (COTS)	User	The User submits an ftppush order, either with or without HEG processing, and is presented with an OMS order id and order status.

Table 4.10-6. WebAccess CSCI Process Interface Events (3 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Request Order Status	One or Many Per Request	Process: Web Browser (COTS)	User	The User requests the order status.
Send HTTP Request	One Per User Request	Process: Application Server (COTS)	Process: Web Browser (COTS)	The Web Browser sends HTTP request on behalf of the user to the Application Server.
Send Application Request	One Per User Request	Process: EcDlWebAccess	Process: Application Server (COTS)	The Application Server sends the request to EcDlWebAccess.
Set Drill-down Search Order	One Per Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: SearchOrderServlet.java SearchOrderBean.java SearchOrderAction.java	The EcDlWebAccess sets the sequence of the searching parameters: Data Group, Theme, Data Set, Date, Time, Spatial, Cloud Cover, Day/Night Flag and Science QA.
Perform Drill-down Search	One Per Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: DrilldownServlet.java SearchRequestBean.java AbstractDataBean.java ISummaryData.java	The EcDlWebAccess performs search based on the current drill-down searching parameters.

Table 4.10-6. WebAccess CSCI Process Interface Events (4 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Select Spatial Area of Interest	One Per Request	Process: Web Browser (COTS)	Process: Web Browser (COTS) Spatial Applet Library: EcDlSpatial.jar	The Web Browser hosts the Spatial Applet, handles user's interaction with a data coverage map of the earth and converts User's selection of spatial area of interest to HTTP request.
Retrieve Search Results	One or Many Per Order Request	Database: DataPool	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: GranuleRetrieverServlet.java GranuleDataBean.java	The EcDlWebAccess retrieves the search results from the Data Pool database, including notable data set level information such as average granule size, source parameter for cloud cover, product quality summary link or whether the data for a data set is typically compressed.
Get File Location	One per file in result set	Database: DataPool	Process: EcDlWebAccess	EcDlWebAccess gets Data Pool disk location for metadata and browse files from the Data Pool database.
Retrieve Metadata	One Per Request	Storage Device: Data Pool disk	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: XMLServlet.java	The EcDlWebAccess retrieves the metadata of a granule from Data Pool disk.

Table 4.10-6. WebAccess CSCI Process Interface Events (5 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Retrieve Browse Image	One Per Request	Storage Device: Data Pool disk	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: BrowseServlet.java	The EcDlWebAccess retrieves the browse image of a granule from Data Pool disk.
Retrieve QA/PH	One Per Request	Storage Device: Data Pool disk	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar	The EcDlWebAccess retrieves the QA/PH of a granule from Data Pool disk.
Add Granules into Shopping Cart	One or Many Per Order Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: GranuleRetrieverServlet.java CartBean.java	The EcDlWebAccess adds one or more granules into a shopping cart.
Remove Granules from Shopping Cart	One or Many Per Order Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: CartServlet.java CartBean.java SetCartInfoAction.java EmptyCartAction.java	The EcDlWebAccess removes granule(s) from a shopping cart.
Present Shopping Cart	One or Many Per Order Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: CartServlet.java CartBean.java DisplayCartAction.java	The EcDlWebAccess returns a shopping cart along with some data set level information and HEG processing options.

Table 4.10-6. WebAccess CSCI Process Interface Events (6 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Get Media Limits	One Per Order Request	Database: Order Manager DB	Process: EcDlWebAccess	EcDlWebAccess gets media limit information from the OMS DB.
Present Media Options	One Set Per order request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: CartServlet.java MediaAction.java	The EcDlWebAccess returns a media option list based on the configured media limits.
Set Media Selection	One Per Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: CartServlet.java MediaAction.java	The EcDlWebAccess saves the media selection for the order.
Get HEG info	One per Request, where convertEnabledFlag is set for one or more collections in cart	Database: DataPool	Process: EcDlWebAccess	EcDlWebAccess gets information from the Data Pool database to determine which, if any, HEG processing options to present.
Set HEG Options	One per Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: CartServlet.java	The EcDlWebAccess saves the HEG processing options for the order.

Table 4.10-6. WebAccess CSCI Process Interface Events (7 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Present User Profile Page	One Per Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: CartServlet.java ProfileAction.java	The EcDlWebAccess returns a user profile page associated with the media via Application Server.
Process FTP Pull NON HEG (Simple Download) Order	One Per Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: CartServlet.java SubmitOrderAction.java DownloadOrderImpl.java	The EcDlWebAccess saves the simple download order into DPL DB via Sybase Server and presents an order acknowledgement page for user to view the order it and download the data.
Insert Orders (FTPPull HEG, FTPPush, Physical Media)	One Per Order Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: CartServlet.java SetCartInfoAction.java OmOrderImpl.java OmHEGOrderImpl.java	The EcDlWebAccess saves the order into OM DB and MSS DB via Sybase Server and presents an order acknowledgement page for user to view the order it and order status.

Table 4.10-6. WebAccess CSCI Process Interface Events (8 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Check Order Status	One or Many Per Order Request	Process: EcDlWebAccess	Application Server (COTS) Process: EcDlWebAccess Library: EcDlWaDrill.jar Class: OrderTrackingServlet.java OrderTrackingBean.java	The EcDlWebAccess tracks the order status in the MSS database, with the email address and order id.
Send Applicatio n Response	One Per User Request	Process: Application Server (COTS)	Process: EcDlWebAccess	EcDlWebAccess sends the response to the Application Server.
Send HTTP Response	One Per User Request	Process: Web Browser (COTS)	Process: Application Server (COTS)	The Application Server sends the response from EcDlWebAccess back to Web Browser.
Display Home Page	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays the home page where user can set the drill-down order or start drill-down search with data set, data group or theme.
Display Drill-down Page	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays the Drill-down page with the values of drill-down parameter.
Display Result Page	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays the Result Page the captures notable data set level information such as average granule size, source parameter for cloud cover, product quality summary link or whether the data for a data set is typically compressed.

Table 4.10-6. WebAccess CSCI Process Interface Events (9 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Display User Profile Page	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays the user profile page during the final step in the order submission process.
Display Shopping Cart Page	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays a shopping cart along with some data set level information and HEG processing options.
Display Media Options	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays the media options available for the current order.
Display Browse Image	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays browse image associated with a granule.
Display QA/PH	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays QA/PH associated with a granule.
Display Metadata	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays the full hierarchy metadata information of a granule.
Display Confirmation Page	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays the confirmation page of the order.
Display Order Status	One Per Request	User	Process: Web Browser (COTS)	The Web Browser displays the order status upon submission of a ftp pull HEG order, ftp push (HEG & NON-HEG) and physical media order (HEG & NON-HEG)
Display HEG Processing Options	One Per Request	User	Process: Web Browser (COTS)	Besides the shopping cart page, the Web browser also displays HEG processing options in the input projection parameter page, band subsetting page and spatial subsetting page.

Table 4.10-6. WebAccess CSCI Process Interface Events (10 of 10)

Event	Event Frequency	Interface	Initiated by	Event Description
Download Granules	One Per User Request for Granule on Results Page	Process: FTP Service	User Process: Web Browser (COTS)	The user downloads granules from the results page using the Data Pool FTP Service.
Retrieve Granules	One Per User Request for Granule on Results Page	Storage Device: Data Pool Disk	Process: FTP Service	The Data Pool FTP Service retrieves the granule from Data Pool disk and downloads it to the user via ftp protocol.

4.10.5 Data Stores

There are two data stores associated with the Data Pool subsystem. They are the Data Pool database (DPL DB) and the Order Manager database (OMS DB). Table 4.10-7 provides a description of these data stores.

Table 4.10-7. Data Pool Data Stores

Data Store	Type	Description
DPL DB	Sybase	The Data Pool (DPL) database implements the large majority of the persistent data requirements for the Data Pool subsystem. The Data Pool database contains: a) inventory data for the Data Pool granules, including data warehousing (Dimension and Fact) data which support Web Access drill down; b) configuration data for the Data Pool; c) interim processing data for the Data Pool utilities; d) data for monitoring Data Pool insert queues and processing; e) Data Pool access statistics; and f) information about data pool entities such as collection groups, collections, file systems, compression algorithms, and themes.
OMS DB	Sybase	The Order Manager (OMS) database stores persistent information about orders placed using the Data Pool WebAccess web pages.

4.11 Bulk Metadata Generation Tool Subsystem Overview

The ECS Bulk Metadata Generator Tool (BMGT) was created to support the development of value-added providers and external search and order tools by providing them with detailed metadata for the collections and granules archived at a DAAC. Currently, the EOS Clearinghouse (ECHO) is the only consumer of this capability. Since ECHO and WIST (soon to be replaced by Reverb) are the primary search and order interface for ECS holdings, BMGT has a vital role in the ECS system. BMGT automatically exports metadata to ECHO as changes occur to the ECS data. Metadata can also be generated and exported on demand. There is currently no supported mechanism for the export of BMGT generated metadata to value added providers other than ECHO. Value added providers interested in this metadata should use ECHO's client APIs to access it.

Generally, BMGT metadata export is initiated at a regular interval as a UNIX cron job to export any changes to DAAC holdings in a timely manner. At the end of each interval, metadata reflecting added, removed, or changed granules and collections in the DAAC archive is generated in XML format. The generated products are exported to ECHO via FTP where they are ingested into the ECHO catalog. In addition to automatically exporting metadata reflecting changes to the archive holdings, the BMGT can be manually executed to generate metadata for specific granules and collections. The output of these 'Manual' exports includes the products requested by the operator. Manual exports can be used to reconcile any discrepancies with ECHO or produce targeted sets of metadata for a particular task. Manually produced BMGT products are generally exported to ECHO, but it is possible to generate metadata products for internal use as well without exporting to ECHO.

In addition to core granule metadata, retrieved from the XML met files in the ECS XML archive, BMGT exports additional metadata which may be useful to ECHO. This additional data includes the visibility of the granule, the URLs (if any) to immediately download the data from the datapool FTP server, and linkages to browse files. The browse image files themselves are also sent to ECHO as part of the metadata package, and can be immediately viewed via the WIST client as well as acquired via FTP links from the ECS datapool.

NOTE: The BMGT uses the terms 'cycle', 'package', and 'export cycle/package' interchangeably, and they are used as such in this document.

4.11.1 BMGT Subsystem Context

Figure 4.11-1 is the BMGT Subsystem context diagram. The diagram shows the high level events generated between BMGT and other subsystems.

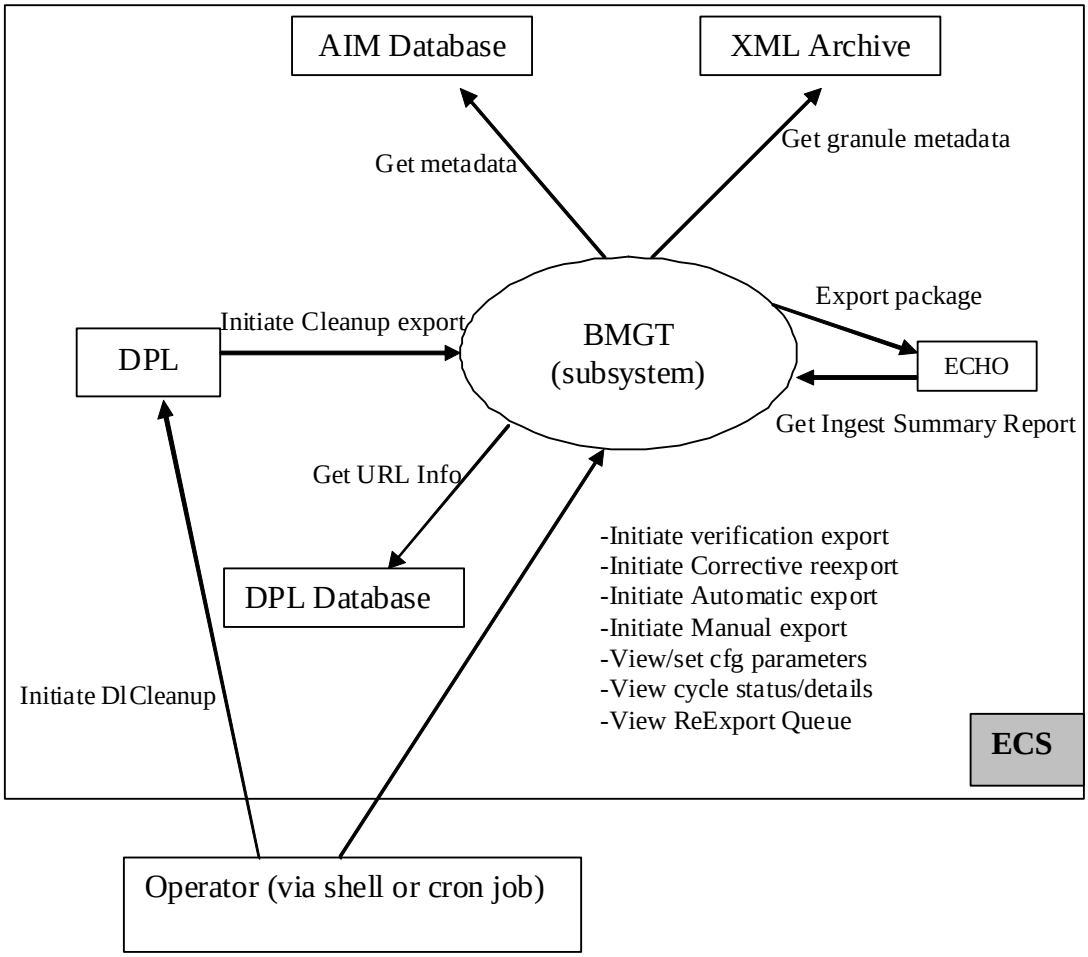

Figure 4.11-1. BMGT Subsystem High Level Context Diagram

Table 4.11-1 provides descriptions of the interface events in the BMGT Subsystem context diagram.

Table 4.11-1. BMGT Subsystem High Level Interface Events

Event	Interface Event Description
Initiate Corrective re-export	The operator initiates a script, which will initiate the generation of a BMGT package which contains the metadata for granules and collections which were queued for re-export due to an error returned from ECHO.
Initiate Verification export	The operator initiates a script, which will initiate the generation of verification metadata for the granules and collections specified by the operator. This metadata will be used to verify that the metadata in ECHO's catalog matches that in ECS.
Initiate Automatic export	The operator initiates a script, usually as a cron job, and the BMGT ensures that the automatic export cycles for the current day have been pre-populated, and initiates the generation of any cycles for which the time range has ended.
Initiate Manual Export	The operator runs a script, which will initiate the generation of an export package containing the metadata requested by the operator i.e., based on a list of granules/collections/groups and desired products, not an event time range.
Get URL Info	BMGT reads necessary information from Data Pool database to support the export of FTP URLs and/or their change or removal due to granule inserts and deletes and collection moves in the public datapool.
Initiate DlCleanup	The operator initiates the DataPool cleanup script to remove some granules from the DataPool, logically and/or physically.
Initiate Cleanup export	Data Pool Cleanup utility initiates a BMGT Cleanup export cycle to convey Data Pool granule deletes to ECHO (See 'Get URL Info' event above). The cleanup cycle is initiated independent of the Automatic export schedule and only exports URL deletes.
Get metadata	BMGT reads collection, granule, browse, and other metadata from the AIM database.
Get granule metadata	BMGT reads granule metadata from the XML metadata files located on the small file archive.
Export package	BMGT uses FTP to push a zip'd copy of the generated products plus associated browse to ECHO.
Get Ingest Summary Report	BMGT uses FTP to pull Ingest Summary Reports from ECHO. These detail the result of ECHO ingest for the packages exported by BMGT.
View/set cfg parameters	The operator uses the BMGT GUI to view the current values for various configuration parameters that affect the behavior of BMGT. If the operator is logged in to the GUI as an administrative user, he/she can modify these values. The Operator can also view, but not modify, the configured error handlers for all errors which may be returned from ECHO in an Ingest Summary report and also view the group configuration and verification status.
View cycle status/details	The operator uses the BMGT GUI to view a list of recent export cycles and their current status. The operator also views details on each cycle or a list of cycles which have failed.
View ReExport Queue	The operator uses the BMGT GUI to view a list of granules currently queued for re-export due to a previous error returned from ECHO.

4.11.2 BMGT/ECHO Interface

The interface from BMGT to ECHO ('Export package' in Figure 4.11-1) is in the form of metadata 'packages'. A package consists of a zip file and zero or more binary browse image files. The zip file contains zero or more XML files which contain metadata on ECS archive holdings (see Table 4.11-2). The zip file always contains at the very least a single manifest file (see Table 4.11-3) which lists the contents of the package (which could be empty), and provides some additional control data.

The interface from ECHO to BMGT('Get ingest summary report' in Figure 4.11-1) is in the form of an Ingest Summary Report (ISR) which contains a summary of the outcome of ECHO's attempted processing of a package sent by BMGT.

Table 4.11-2 and 4.11-3 describe the types of files which make up the ECHO/BMGT interface. *Note: all schema files referenced below can be found at http://www.echo.nasa.gov/ingest/schemas/operations/*

Table 4.11-2. BMGT Metadata Product File Types

File Type	Schema	Description
METC	Collection.xsd	Collection metadata.
METG	Granule.xsd	Granule metadata.
BBR	Browse.xsd	Browse granule metadata.
Browse Data	*N/A*	Binary Browse image file.
METU	Granule.xsd	Granule QA, URL, visibility, restriction, and browse linkage updates.

Table 4.11-3. BMGT/ECHO Interface Control File Types

File Type	Schema	Description
Manifest	PackageManifest.xsd	List of files contained in a package as well as additional control data.
Ingest Summary Report	IngestReport.xsd	Summary of the outcome of ECHO's attempted processing of a metadata package.

4.11.3 ECS Events and BMGT products

In general, BMGT is run automatically on a set schedule, splitting a 24 hour day into 1 or more equally sized intervals. At the end of each interval, BMGT will export to ECHO (via the interface described in 1.1.2) metadata reflecting any relevant changes to the ECS holdings since the end of the last interval. Relevant changes include inserts, deletes, and updates to collections and granules which are configured to be exported to ECHO. Such events cause database triggers to be fired which record the events in database tables where they are then picked up by BMGT. BMGT will evaluate all relevant events in a given time frame, and export the new state of the affected object as an add, delete, or update to the ECS metadata stored at ECHO. If any event

has been superseded by a later event, it will not be exported, or will be modified to reflect the actual state of the granule or collection in ECS. Table 4.11-4 describes the types of events which are relevant to BMGT/ECHO, and what type of product is exported to reflect each event during an automatic export cycle. In a Manual BMGT export cycle, the same types of metadata are generated and exported, but for objects and products specified by operator input rather than by events. A Manual BMGT cycle will only generate inserts and deletes, not updates (with a few exceptions).

Table 4.11-4 also describes the root schema element for each type of event (in the "Products" column) as well as the location of that element in the document tree (under the "Notes" column). Note that most granule metadata updates are processed as a delete followed by an add.

Most BMGT products are grouped together by product type (METG, URL, METC, etc) and by ESDT grouping (defined in site specific configuration). This will result in one (or more, depending on volume) files in a package per product and ESDT group combination for which there is metadata to be produced.

Table 4.11-4. ECS Event to BMGT Product Mapping (1 of 2)

Event Type	Cause/location	Products [<FileType> <Schema element>]	Notes
GRINSERT	Granule inserted into AIM database, or logically undeleted.	[METG Granule] If associated browse (if any) not already exported, export it: ***See BRINSERT***	METG Contains all metadata associated with the granule such as URLs, Browse linkages, etc. Under GranuleMetaDataFile/Granules
GRDELETE	Granule deleted from AIM either logically or physically.	[METG GranuleDelete] If Associated Browse has no more links, remove it: ***See BRDELETE***	Deletes all metadata associated with the granule. Under GranuleMetaDataFile/GranuleDeletes
GRUPDATE	Granule core metadata updated in AIM/XML file.	[METG Granule]	Overwrites old granule metadata with new. Under GranuleMetaDataFile/Granules
GRHIDE	Granule hidden in AIM DB, and not available for order.	[METU Visible]	Under GranuleMetaDataFile/GranulePartialAdds/ GranulePartialAdd/Fields/Field/
GRUNHIDE	Granule unhidden in AIM DB, and now available for order.	[METU Visible]	Under GranuleMetaDataFile/GranulePartialAdds/ GranulePartialAdd/Fields/Field/
CLINSERT	Collection inserted into AIM DB.	[METC Collection]	Under CollectionmetaDataFile/Collections
CLDELETE	Collection deleted from AIM DB.	[METC CollectionDelete]	Under CollectionMetaDataFile/CollectionDeletes

Table 4.11-4. ECS Event to BMGT Product Mapping (2 of 2)

Event Type	Cause/location	Products [<FileType> <Schema element>]	Notes
CLUPDATE	Collection modified in AIM DB.	[METC Collection]	Overwrites old collection metadata with new. Under CollectionMetaDataFile/Collections
BRINSERT	Browse file inserted in AIM DB	[BBR BrowseImage + Browse binary file]	Exports actual browse data + its metadata. Under BrowseMetaDataFile/BrowseImages
BRDELETE	Last link to browse file deleted in AIM DB	[BBR BrowseImageDelete]	Under BrowseMetaDataFile/BrtowseImageDeletes
BRLINK	Science/Browse link added to AIM DB.	[METU BrowseImage, AllBrowseImages]	Adds browse link to affected granule (removing any existing link first). Under GranuleMetaDataFile/GranulePartial[Adds|Deletes] / GranulePartial[Add|Delete]/Fields/Field/
BRUNLINK	Science/Browse link removed from AIM DB.	[METU AllBrowseImages] If Browse has no more links, remove it: ***See BRDELETE.***	Instructs ECHO to remove any Browse linkages for specified granule. Under GranuleMetaDataFile/GranulePartial[Adds|Deletes] / GranulePartial[Add|Delete]/Fields/Field/
QAUPDATE	QA parameters modified in AIM DB.	[METU MeasuredParameter]	Under GranuleMetaDataFile/GranulePartial[Adds|Deletes] / GranulePartial[Add|Delete]/Fields/Field/
INSERT (datapool)	Granule Added to public datapool.	[METU OnlineAccessURL, AllOnlineAccessURLs, OnlineResource, AllOnlineResources]	Under GranuleMetaDataFile/GranulePartial[Adds|Deletes] / GranulePartial[Add|Delete]/Fields/Field/
DELETE (datapool)	Granule logically or physically removed from public datapool.	[METU OnlineAccessURL, AllOnlineAccessURLs, OnlineResource, AllOnlineResources]	Under GranuleMetaDataFile/GranulePartial[Adds|Deletes] / GranulePartial[Add|Delete]/Fields/Field/
CLMOVED (datapool)	Collection metadata updated in DPL DB.	[METU OnlineAccessURL, AllOnlineAccessURLs, OnlineResource, AllOnlineResources]	Replaces URL(s) of granules in affected collection & in public DPL. Under GranuleMetaDataFile/GranulePartial[Adds|Deletes] / GranulePartial[Add|Delete]/Fields/Field/

4.11.4 BMGT Architecture

Figure 4.11-2 displays the BMGT Architecture diagram.

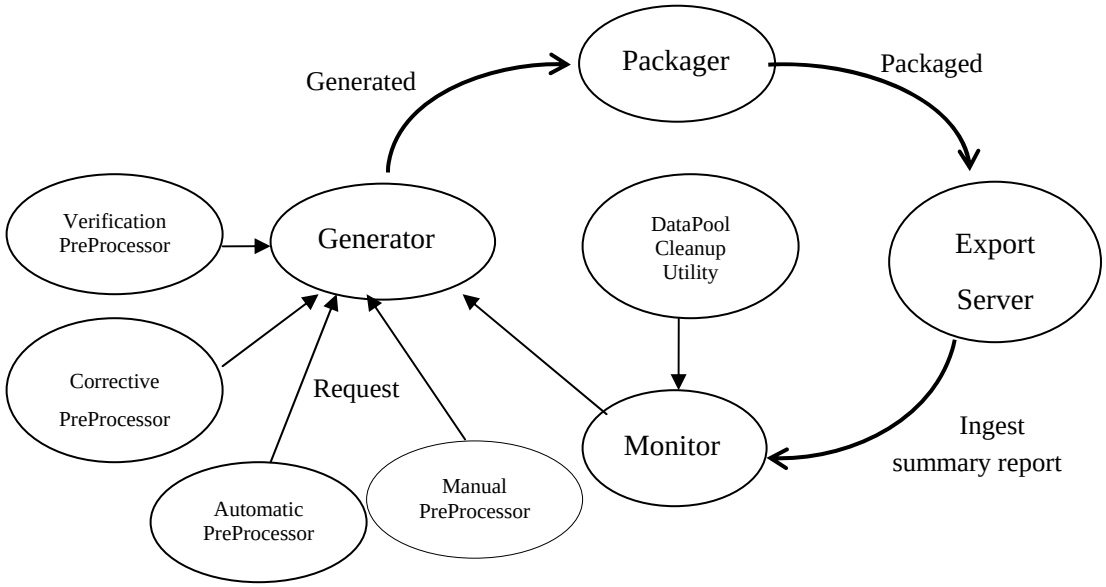

Figure 4.11-2. BMGT Architecture Diagram

Table 4.11-5 provides descriptions of processes shown in the architecture diagram.

Table 4.11-5. BMGT Processes (1 of 3)

Process	Type	Hardware CI	COTS/ Developed	Functionality
Automatic Pre Processor	Application	OMSHW	Developed	Script is kicked off by operator or cron job. Based on BMGT configuration, the BMGT determines whether any automatic export cycles are ready to be generated/exported. If so, they are prepared and flagged for generation. If necessary, will also prepopulate the export schedule for the current day.
Manual Pre Processor	Application	OMSHW	Developed	The operator runs a script to explicitly tell the BMGT to initiate an export cycle. The Manual start script provides a large number of options for generating the export package to fit the operator's needs.
Corrective Pre Processor	Application	OMSHW	Developed	The operator runs a script to initiate the re export of full metadata for granules which were queued for re export after an error was returned from ECHO indicating a problem ingesting metadata for the granule.
Verification Pre Processor	Application	OMSHW	Developed	The Operator runs a script to initiate the export of full metadata for a specified list of granules and collections for the purpose of verifying and reconciling any differences between ECS and ECHO catalogs.
DataPool Cleanup Utility	Application	DPLHW	Developed	When Data Pool Cleanup is run, it triggers the BMGT to produce an export package containing METU products in order to notify ECHO as quickly as possible of the removal of granules from the public Data Pool. The BMGT Monitor is also involved in this process as it prepares and flags the cleanup cycle for generation after being notified by DPL cleanup that a cleanup cycle is needed.

Table 4.11-5. BMGT Processes (2 of 3)

Process	Type	Hardware CI	COTS/ Developed	Functionality
Generator	Server	OMSHW	Developed	Once a BMGT cycle has been flagged for generation, the task of creating the required output products (ECSMETC, ECSMETG, ECSBBR, and **ECSMETU**[*Qa/BrLink, Visibility metadata, URL metadata*]) falls to the Generator. For performance reasons, the Generator is a multi-threaded application. It uses the information assembled by the preprocessor to create one or more output products. It also manages dependencies between products. Once the products have been created, the work passes to the Packager.
Packager	Server	OMSHW	Developed	The Packager is responsible for 'assembling' the export package. This includes renaming files to include file counts and creating manifest and zip files for export to ECHO. The Packager will also generate synchronization packages where necessary to ensure that BMGT and ECHO keep their unique package identifiers in sync. Sync packages are generated when a sequenced manual package fails to generate. The empty sync package takes the place of the failed package and allows other cycles to proceed.
Export Server	Server	OMSHW	Developed	Once the products have been packaged, the Export Server takes over. It is responsible for FTP'ing the package to ECHO, as well as pulling Ingest Summary Reports from ECHO to the local system where they can be picked up by the Monitor.

Table 4.11-5. BMGT Processes (3 of 3)

Process	Type	Hardware CI	COTS/ Developed	Functionality
Monitor	Server	OMSHW	Developed	The Monitor performs a number of general purpose 'monitoring' tasks, such as making sure the export cycles do not get 'stalled' and cleaning up old packages. It also performs some of the preprocessor work for cleanup export packages requested by DataPool cleanup and processes the Ingest Summary Reports generated by ECHO, handling any errors reported within.

EBIS document 920-TDx-001 (HW Design Diagram) provides descriptions of the HWCIs and document 920-TDx-002 (Hardware-Software Map) provides site-specific hardware/software mapping.

4.11.5 Use of COTS in the BMGT Subsystem

- JRE

  The JRE constitutes the Java virtual machine and the Java platform core libraries. It provides applications with the Java platform. Included with it is JAXP (Java API for XML Processing), which is also used by BMGT.

- jConnect

  jConnect implements the JDBC interface and provides Java applications with drivers to access Sybase database SQL server.

- JDOM

  JDOM libraries allow java applications to create and edit xml documents.

- Sybase Server

  The BMGT accesses the Inventory database to read inventory metadata, Data Pool database to get URL information for granules in Datapool, and Ingest database to monitor ingest of generated products into ECS.

- JAF / Javamail

  Java Activation Framework (JAF) and Javamail provide BMGT the capability to send email messages.

- Protomatter

  Used to maintain a pool of Database connections for both Datapool and Inventory databases.

- Perl

  Interpreted scripting language used to implement the Manual Start Script (EcBmBMGTManualStart.pl).

- Jaxen

  An open source java XPath library.

4.11.6 BMGT Subsystem Software Description

4.11.6.1 BMGT CSCI Functional Overview

The BMGT servers shown in Table 4.11-2 are always running, waiting for work to do. However, metadata generation will only be initiated when one of the five 'preprocessor' utilities is run, either by the operator or as a cron job. There is one preprocessor for each BMGT cycle type (AUTOMATIC, MANUAL, CORRECTIVE, verification [VER_LONG, VER_SHORT, and VER_INC], and CLEANUP). Each preprocessor is responsible for deciding which products need to be generated for the cycle, as well as which granules or events go into those products. Each preprocessor will ensure that the appropriate events (or granules/collections for manual, verification, or corrective export) are flagged to be processed in the new cycle, assign the cycle a unique and sequential package ID (if needed), and flag the cycle as ready to generate. Verification, CORRECTIVE, and MANUAL pre processors are actually the same executable, the Manual Preprocessor, but are called with different options in each case.

In general, the use case for each BMGT cycle type is as follows:

AUTOMATIC. Preprocessor is kicked off as cron job, configured to run at a set interval (could also be run by operator). Based on BMGT configuration, the BMGT determines whether any automatic export cycles are ready to be generated/exported. If so, they are prepared and flagged for generation. If the pre processor has not yet been run during the current day, then all cycles for the current day are pre populated with placeholders.

MANUAL. The operator runs the manual preprocessor script to explicitly tell the BMGT to initiate an export cycle. This script provides a large number of options for generating the export package to fit the operator's needs.

CORRECTIVE. The operator runs the corrective preprocessor script to initiate the re export of granules queued for re export due to ECHO ingest error.

VER_SHORT. The operator runs the verification preprocessor to initiate the export of a listing of all granules, collections, or browse in a particular collection (or the entire catalog in the case of collections) for the purpose of comparing with the equivalent listing in ECHO.

VER_LONG. The operator runs the verification preprocessor to initiate the full metadata export of the specified granules and/or collections for the purpose of full comparison with the metadata on record in ECHO.

VER_INC. The operator runs the verification preprocessor (or it is run via a cron) to initiate the full metadata export granules and collections automatically selected by the preprocessor as the least recently updated granules (and their associated collections) which have not yet been incrementally verified in this manner since their last update. This metadata is compared against that on record in ECHO.

CLEANUP. When Data Pool Cleanup is run, it triggers the BMGT to produce an export package containing METU products in order to notify ECHO as quickly as possible of the removal of granules from the Data Pool. The BMGT Monitor is also involved in this process as it prepares and flags the cleanup cycle for generation after being notified by cleanup that a cleanup package is required. DlCleanup and BMGT Monitor together act as the Cleanup preprocessor.

Once a cycle has been preprocessed and flagged for generation by the preprocessor, the BMGT servers handle the remainder of the cycle's processing. A nominal export cycle will be picked up by the Generator server, which will generate the appropriate products. The cycle will then be picked up by the Packager server, which will rename the products to include a file count, create a manifest file, and create a zip file containing the products. The Export server will then FTP push the zip file, along with any browse files to ECHO and then wait for an Ingest Summary Report to be generated by ECHO, which it will then pull via FTP to a local directory. The Monitor server will ensure that an Ingest Summary Report is received within a configured time limit, and when it does, will read it, handle any errors contained within, and update the cycle status accordingly. Assuming no errors in the report, the Monitor will clean up the files generated by the cycle, as well as some of the cycle's database records. If there is an error, the monitor will handle the error as detailed in Section 4.11.6.8. After a configured amount of time, the Monitor will completely remove the cycle from the BMGT records.

The following sections provide more detailed descriptions of the various components briefly described above.

4.11.6.2 Automatic Export Process

The Automatic Export Process is a command line tool, usually spawned by a cron task. Its purpose is to initiate automatic export cycles at configured intervals. Since the process may be executed more frequently than the configure interval, it must perform checks to make sure it really is time to initiate an export cycle (i.e., there is a complete export interval which has not yet been flagged for generation), and exit otherwise. When the automatic preprocessor is run for the first time for a given day, a complete days worth of export cycles are 'pre-populated' in the database as placeholders to be initiated at a later time.

When an export cycle is to be started, the Automatic Preprocessor selects the events that will be used to generate the output products. These events, generated automatically by the database during normal operations, effectively document the changes that have occurred to the metadata

holdings, such as granule or collection inserts, updates, or deletions. The Automatic Preprocessor copies the events relevant to the time interval covered by the export cycle into special BMGT working tables for use by the Generator.

Once the events have been selected, the Automatic Preprocessor sets up the collection group configuration. This configuration, initially stored in a file, is used to group the data in the output products. It must first be loaded into the database, and then checks are performed in order to detect configuration changes, such as the recent enabling of a collection for collection or granule metadata export. These changes require additional data to be exported, and the Automatic Preprocessor converts them into one or more events, which are added to the BMGT working tables for processing. The final step in collection group configuration is to make a copy of the current configuration (so that a subsequent run can perform a comparison in order to detect the changes described above).

Once the collection group configuration has been loaded, the audit trail can be created, recording statistics and status for each product in a cycle. This is used later by the Generator to know what products to generate and track progress, and the Monitor to verify the Ingest Summary Reports.

Next, the Automatic Preprocessor checks for collection deletion events. If a collection is deleted, then the BMGT must not export any granules for that collection. So if it finds a collection deletion event, the Automatic Preprocessor removes all other events for granules belonging to that collection from the working tables.

At this point, the cycle is ready to be generated, and the Automatic Preprocessor updates the database so that the cycle will be picked up by the Generator. It also assigns a package sequence number to the export cycle. In order to prevent possible recovery problems and gaps in the sequence number, these two steps are performed atomically.

The Automatic Export Process has now completed its task, and will exit. Figure 4.11-3 below summarizes the sequence of events that occur in the database.

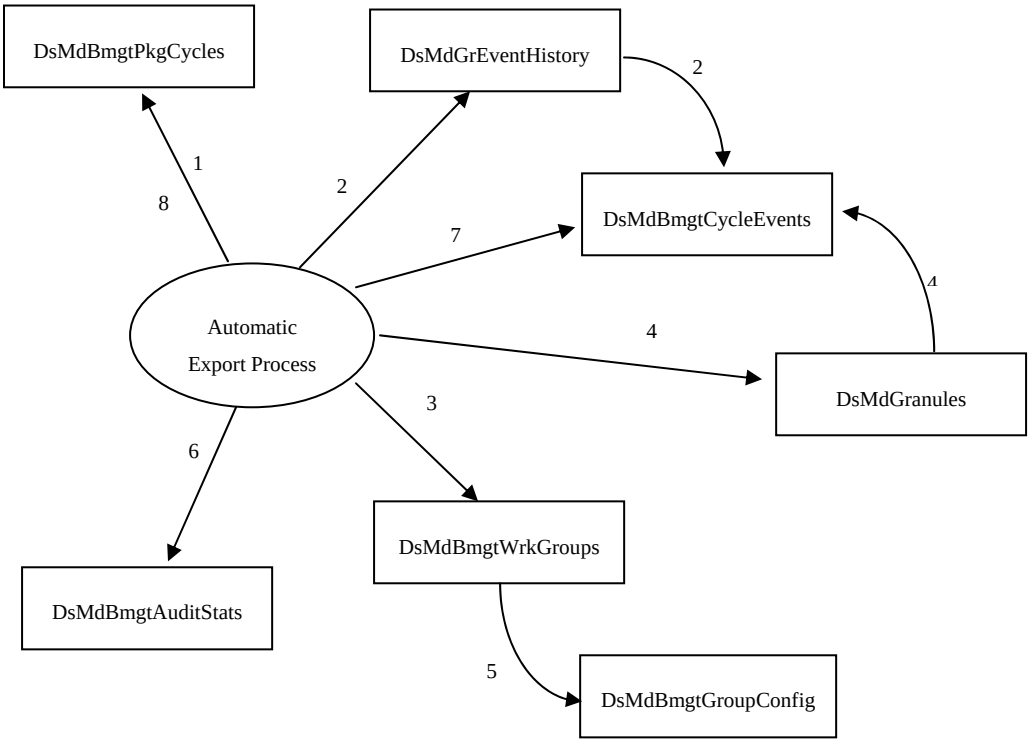

Figure 4.11-3. Automatic Preprocessor database sequence

The Automatic Preprocessor performs the following database interactions (diagrammed in Figure 4.11-3) to prepare a cycle for generation:

1. Retrieve export cycle details
2. Select events for export cycle into working table
3. Load collection group configuration
4. Add events related to configuration changes
5. Make permanent copy of collection group configuration
6. Create audit trail
7. Remove events related to collection deletions
8. Update export cycle to trigger generation

4.11.6.3 Manual Export Process

The Manual Preprocessor is a command line tool used by the operator (called from the command line or a cron). Its purpose is to allow the operator to generate custom export cycles. For example, a targeted package containing only a few specific granules and products may be required to correct synchronization issues between ECS and ECHO. The Manual preprocessor

also allows the initiation of verification and corrective export packages to find and repair discrepancies with ECHO. The Manual Preprocessor provides a wide range of command line options that allow the selection of granules by ID, collection, collection group, or a combination of collection and an insert or update time range. It also allows the operator to select the export type, output products that should be generated, and whether or not they will have a sequence number or be exported to ECHO.

The first step performed by the Manual Preprocessor is to verify the command line options to eliminate any incompatible options, or return an error if required options are omitted. The Preprocessor will then check whether another manual export start script is currently running and exit if so. In addition, it will check if another cycle is currently generating and if so, it must verify with the operator that this is acceptable. At this time, the Preprocessor will also check for any raised alerts and will exit if any are found.

Once the command line options have been verified, it creates an export cycle entry in the database in order to create a unique identifier for this export cycle (note that this entry also records other package information, such as whether it should exclude subsequent automatic runs from processing concurrently, the products that are to be generated, whether it is a deletion run, and whether it is being exported). It then uses the command line options to build a list of granules and/or collections in the database (while product generation for an automatic export cycle works exclusively from event lists, in the case of a manual export cycle, it works exclusively from granule and collection lists).

If the command line options provide explicitly granule IDs, either directly, or via a file, these are loaded into a special working table. If one or more collections are provided, these are loaded into another working table, and then used to select their associated granules (if necessary). If one or more groups are provided, then the collections associated with the specified group(s) are loaded into the collection working table mentioned above, and used to populate the granule working table if necessary.

If the command line options specify that this is to be an incremental verification or corrective export, then the granule/collection tables are populated automatically based on the contents of other database tables.

Once the granule and collection lists have been populated, the Manual Export Process loads the collection group configuration. This is slightly simpler than the automatic export case, and simply involves loading the configuration into the database (no configuration changes need to be detected). The Manual Export Process will attempt to load configuration for only those groups required by the command line options (this is not always possible, so sometimes unnecessary groups will be loaded into the database tables).

Once the collection group configuration has been loaded, the audit trail can be created for the requested products and groups. This is used later by the Generator to know what to generate and to track progress, and the Monitor to verify the Ingest Summary Reports.

At this point, the cycle is ready to be generated, and the Manual Preprocessor updates the database to assign a package sequence number (if requested), and at the same time, update a

status column so that the cycle will be picked up by the Generator. To prevent race conditions, these operations are performed as one atomic operation.

The Manual Export Process has now completed its task, and will exit. Figure 4.11-4 below summarizes the sequence of events that occur in the database.

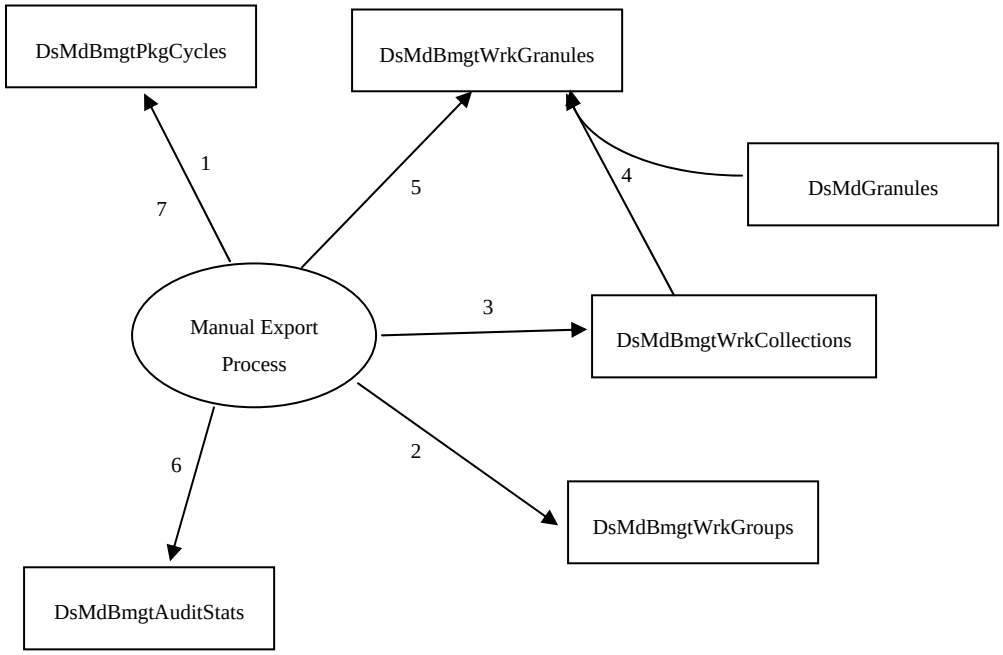

Figure 4.11-4. Manual Export Process database sequence

The Manual Preprocessor performs the following database interactions (diagrammed in Figure 4.11-4) to prepare a cycle for generation:

1. Create export cycle row
2. Load group configuration
3. Load collections into working table
4. Populate granules based on collection/insert time (if METG being produced)
5. Load granules into working table
6. Create audit trail
7. Update export cycle to trigger generation

4.11.6.4 Cleanup

The DataPool Cleanup Utility is part of the DataPool subsystem which has been modified to interface directly with BMGT. It is run periodically, either on a cron, or manually by an

operator, to remove granules from the DataPool and free disk space. If any of the removed granules were public in the DataPool, it must invoke BMGT to notify ECHO that the DataPool URLs it has for them are no longer valid.

When Cleanup is initiated, either in 'predelete' or 'deleteall' modes, it will create a 'CLEANUP' cycle entry in the database. Unlike automatic and manual cycles, there is very little flexibility in how a cleanup cycle is generated and exported – it will be assigned a package sequence ID, exported to ECHO, and it will generate only the METU product. Before exiting (or moving on to delete granules, in 'deleteall' mode), Cleanup will wait to make sure the cycle has been recognized by the BMGT Monitor server. The Monitor, in turn, will select the URL delete events that are to be processed by this export cycle. In this respect, a Cleanup export cycle behaves as an automatic export cycle (i.e. it works from an event list). Note that with the introduction of the Online archive, the Cleanup Utility is not used very often.

Once the events have been selected, the Monitor will create the audit trail for the export cycle. This is very simple, since it consists of only METU products.

Finally, the Monitor will assign a package sequence number and mark the export cycle to tell the Generator that it can be processed.

NOTE: The CLEANUP cycle functionality is no longer widely used and may be removed entirely in a subsequent release. Any users depending on this functionality should let SDPS developers know.

4.11.6.5 Generator

The Generator is responsible for generating the actual products. Products include collection metadata, granule metadata, browse metadata, and updates to granule metadata.

The Generator is a server process. It monitors the database export cycles table looking for packages that are ready for generation (as happens when the Manual or Automatic Preprocessor, or DataPool Cleanup runs). These packages will initially be in the NEW state. When it finds a package that is ready, it retrieves the export cycle details.

Before the Generator can start processing a package, it must first check to see what type of package it is, and whether there are any currently executing packages that may prevent it from executing, or if there is an Alert raised in the BMGT system preventing certain types of cycle from running. For example:

An automatic export cycle package cannot be processed if there is currently another automatic export cycle package being processed.

A cleanup export cycle package cannot be processed if there is currently another cleanup export cycle being processed.

An automatic or cleanup export cycle cannot be processed it there is a manual export cycle being processed, and the manual export cycle is marked as being 'exclusive'.

Only a non-sequenced cycle can be processed if there is an Automatic Alert in place.

Only Automatic and non-sequenced Manual cycles can be processed if there is a Verification Alert in place.

Assuming that the package can be processed, the Generator starts its work. First, it updates the package state to STARTED, to allow the GUI to display the fact that it is now being worked. It then determines what products are going to be created. It then creates a number of specialized product processor instances – once for each product that is to be generated. It also creates a specialized data source instance for each product. These data sources are abstracted classes which allow the differing sources of product data (events vs. granule lists) to be hidden from the product generation code.

Once all the product processors and data sources have been initialized, the Generator creates a number of threads to perform the actual work. The Generator is responsible for managing the dependencies between the different products. For example, the METU product cannot be generated until the METG product has completed. The generator will sequence these dependencies as follows.

All products requested:

- Collection and granule metadata are generated
- If granule metadata are successfully generated then
    - Browse and METU are generated
    - If browse generation is successful then
        - Browse link updates (METU) are generated

Only Browse products requested:

- Browse product is generated
- If Browse generation successful then
    - Generate browse links product

Only granule metadata product is requested:

- Granule metadata are generated
- If granule metadata generate successfully then
    - Generate browse and METU
    - If browse generated successful then
        - Generate browse links (METU)

While the different products have different algorithms for generating the output, they all follow a common theme. A product consists of one or more product groups. The processing thread will iterate through these product groups, one at a time. For each product group, it will select all the data for that product group from the data source, and generate zero or more output files. It repeats this process until there are no more product groups. The status of each product is recorded at the product group level, including the number of granules or collections (by insert, update, delete), as well as the number of skipped granules/collections.

Once all products have finished, the Generator reviews the status of all the product groups. If there were any errors, or the number of skipped granules exceeds a configurable limit, the product generation fails. In this case, the Generator will send an email, and set the status of the export cycle to PRODUCT GENERATE FAILED. If products were generated successfully, the status will be set to PRODUCT GENERATED.

All of the algorithms (source code and database) in the Generator are designed in such a way that they can be restarted – for example, if the Generator was abruptly terminated, or the machine died. Simply restarting the Generator will cause the package to be picked up and continued. This recovery works at the product group level, so that completed product groups would not be processed again, but incomplete product groups (partially processed, or completed product groups that contained skipped granules) would be restarted. This applies to all types of export cycles – manual, automatic, corrective, cleanup, etc.

4.11.6.6 Packager

The Packager server is responsible for:

- Renaming files to include file count values
- Creating manifest and ZIP file for ECHO export

The Packager periodically checks the database to look for export cycles that have reached the PRODUCT GENERATED state. These export cycles are now eligible for processing by the Packager.

The first task for the Packager is to rename the product files. Primarily, it must include the file count in some of the filenames (this is generally difficult to do in the generator, since the number of output files is not known until they have all been created).

Next, if the package is due for export to ECHO, the Packager creates a manifest file (a XML file listing the package contents). It then zips up the XML files and the manifest into a single ZIP file. Browse files, if any are not added to the zip file.

Packages that produced no outputs in the generation phase will still send an empty package to ECHO. This is done to maintain the sequence numbers of packages – for example it is possible for a manual run to be started using the next available sequence number, then an automatic run to be started. The manual run may produce no products, but the subsequent automatic run does produce products and its sequence number is one higher. If the Packager did not send the empty package, a sequence number would have been skipped and the ECHO would not then process the

outputs from the automatic run. For this purpose, in these situations, a SYNC package will be requested, and the Packager will generate this as an empty package.

Once the export cycle's product files have been packaged, the Packager updates the status of the export cycle to PACKAGE GENERATED.

4.11.6.7 Export Server

The Export Server is responsible for transferring an export package to ECHO via FTP, and for retrieving Ingest Summary Reports.

The Export Server periodically checks the export cycles table in the database for packages that are in the PACKAGE GENERATED state, and are configured for export to ECHO.

When it finds such a package, the Export Server updates the export cycle state to TRANSFERRING. It then establishes an FTP session with the ECHO FTP server, and starts transferring the files.

Due to the way in which ECHO works, the Export Server is required to transfer the package ZIP file last, after any BROWSE files. Since the zip file is likely to be smaller than the BROWSE files, and there is only one zip file, it is used to signal that the transfer is complete. ECHO will wait a configured amount of time after finding a new zip file before uncompressing and extracting the XML files and reading the Manifest.

Upon completing the file transfers, the Export Server will update the export cycle status to EXPORTED.

The Export Server will also periodically poll the ECHO FTP area for Ingest Summary Report files (It is assumed that ECHO will copy the complete Ingest Summary Report file to their FTP area rather than write the file in place to minimize the chance of the BMGT Export Server picking up an incomplete report). The Export Server will pull the Ingest Summary Report to a temporary area on the local machine, and then move it to a configured drop box directory (where the Monitor will find it).

The Export Server operation can be suspended or resumed by the operator via the GUI. While the Export Server is suspended, FTP operations to and from ECHO will be suspended.

By default, all FTP connections will be made through the DataPool Ftp Service which allows connections to be reused and provides a common FTP interface for ECS custom code. The operator is also able to configure the Exporter to use the legacy standalone FTP client (which was the only option prior to release 8.0), which relies on Apache FTP libraries. It is highly recommended that the DataPool FTP service be used, and the legacy client is only preserved for transition and backwards compatibility purposes. It will likely be removed in a subsequent release.

4.11.6.8 Monitor

The Monitor Server is responsible for processing Ingest Summary Reports, cleaning up export package files when required, looking for 'stalled' packages, and initiating CLEANUP cycles requested by DataPool Cleanup.

Once an Ingest Summary Report is found in the Ingest Summary Report drop box directory, the Monitor parses the report and inserts the statistics it contains into the Inventory Database audit trail. If the Ingest Summary Report indicates a complete success, the export cycle status is updated to COMPLETE. The Monitor then cleans up the package files, and some of the database tables (excluding the audit trail).

Alternatively, if the Ingest Summary Report indicates one or more errors, one of a set of error handling policies will be used to alert the operator and/or attempt to remedy the error situation. The possible error handling policies are outlined below, and the listing of which policy is used for which error can be found in the BMGT GUI on the "Configuration/Error Tuning" page. A further discussion of the error handling policies can be found in the ECS/ECHO ICD.

4.11.6.8.1 BMGT Monitor Error Handling Policies

The following is a list of the error handling policies used by the BMGT Monitor for handling of ECHO errors returned in the ISR. Each policy also lists the associated Response Code which represents the response in the BMGT GUI. For further information, and an error to response mapping, see the ECS/ECHO ICD.

4.11.6.8.1.1 Responses for Job and File Level Errors

Retry Package

The metadata will be repackaged and re-FTP'd to ECHO with the same sequence number. The metadata files will not be regenerated.

Response Code: RETRY_PACKAGE

Regenerate Package

The package metadata files will be generated again, re packaged, and FTP'd to ECHO with the same sequence number.

Response Code: REGENERATE_PACKAGE

In both cases, the regeneration and retransfer includes the browse files that were associated with this export. DAAC staff must investigate if continued re-export does not succeed. Although the BMGT will continue to generate subsequent automatic export packages, ECHO will stop processing them until the missing package is ingested. If re-export elicits the same (or equally serious) error from ECHO, and the number of such errors for that package exceeds the threshold defined by NUM_RETRIES_FOR_ALERT in the BMGT configuration, then an alert will be raised, restricting what can be exported by BMGT before the issue has been addressed and the alert cleared. If the errant package was of type VER_INC, then the alert will limit exports to AUTOMATIC and un sequenced MANUAL packages. If the errant package was of type

AUTOMATIC, then the alert will limit exports to only unsequenced MANUAL packages. Clearing the alert will return BMGT export to normal.

Any file level error which elicits one of these two responses will be accompanied by a job level error. This ensures that BMGT only re-exports a package if no file in that package was ingested. If an entire file in a package fails, the whole package will fail.

These error policies are only valid for AUTOMATIC and VER_INC packages. If an error configured for one of these policies is received for another type of package, then a SYNC package will simply be exported in place of the failed package, and the original package will have a status of COMPLETE_WITH_ERRORS.

4.11.6.8.1.2 Item Errors (automatically handled)

Many types of item level errors may be able to be automatically handled depending on the status of the objects in question. Such errors occur, for instance, when an attempt to export a browse link for a granule fails because ECHO cannot find the referenced browse granule; a granule is updated (e.g., by trying to add a URL) but ECHO does not have the granule in its inventory; or if a granule verification is exported for a granule that is not in the ECHO inventory. In these cases, either ECHO will attempt to repair the error, BMGT will attempt to re-export the affected object to ECHO, or BMGT will ignore the error. In all but one case (Ignore Error), these actions will result in the BMGT package being placed in the “COMPLETE_WITH_WARNINGS” state, indicating that there was an error, but that it does not require immediate DAAC staff attention.

Ignore Error

The error will not cause any email to be sent, or any state change to the package. The BMGT package will be placed in the "COMPLETE" state (barring any additional, more serious errors). An error can only be ignored if the error indicates that the ECS and ECHO inventories are in agreement. For instance, if a granule delete fails because the granule was not present in ECHO’s inventory. If the inventories are not in agreement, then the error will be handled by the Notify DAAC Staff Handler.

Response Code: IGNORE_ERROR

Ignore Error Completely

The same as Ignore Error, but the error will be ignored regardless of whether the ECS and ECHO inventories are in agreement.

Response Code: IGNORE_COMPLETELY

Error Handled by ECHO

ECHO attempted to repair the error. If the attempt failed, then there will be additional errors. No email will be sent, and the ID of the object for which the error occurred will be added to a list which can be used at a later point to verify that ECHO did indeed fix the problem. The BMGT package will be placed in the “COMPLETE_WITH_WARNINGS" state.

Response Code: VER_AUTO_HANDLER

ReExport Object

The object will be added to the ReExport queue so that the object can be re-inserted into ECHO in a future corrective re export. An email will be sent to the DAAC staff indicating the error that occurred and the number of items added to the reExport Queue. The BMGT package will be placed in the "COMPLETE_WITH_WARNINGS" state.

Response Code: REEXPORT_OBJECT

ReExport Associated Object

An object associated with the object that caused the error will be added to the ReExport queue so that the object can be re-inserted into ECHO in a future corrective re export. The ID of this associated object can be found in the error message text. An email will be sent to the DAAC staff indicating the error that occurred and the number of items added to the reExport Queue. The BMGT package will be placed in the "COMPLETE_WITH_WARNINGS" state.

Response Code: REEXPORT_ASSOCIATED_OBJECT

ReExport Object Delete

The object will be added to the ReExport queue so that the object can be removed from ECHO in a future corrective re export. An email will be sent to the DAAC staff indicating the error that occurred and the number of items added to the reExport Queue. The BMGT package will be placed in the "COMPLETE_WITH_WARNINGS" state.

Response Code: OBJECT_DELETE_REEXPORT

4.11.6.8.1.3 General Errors (not automatically handled)

The majority of errors cannot be automatically handled and will result in an alert to the DAAC staff as described by the responses below. These responses can occur for errors at the job, file, and item level.

Notify DAAC Staff

An alert email will be sent to DAAC staff indicating the error that occurred. The BMGT package will be placed in the “COMPLETE_WITH_ERRORS" state. This is the default error response, and will be used for any unrecognized errors.

Response Code: NO_OBJECT_REEXPORT, DUPLICATE_PACKAGE

Contact ECHO

An alert email will be sent to DAAC staff indicating the error that occurred, and that they should contact ECHO to resolve the issue. The BMGT package will be placed in the "COMPLETE_WITH_ERRORS" state.

Response Code: NO_OBJECT_REEXPORT_CONTACT_ECHO

The Retry Package and Ignore Error responses have variants that will instruct the operator to contact ECHO in addition to their normal responses. In the case of the Ignore Error response, this instruction will only be given if the error is not ignored. See the Retry Package and Ignore Error response descriptions for more info on these responses

Response Code: RETRY_PACKAGE_CONTACT_ECHO,
IGNORE_ERROR_CONTACT_ECHO

Send SYNC package

An empty Synchronization package will be sent to ECHO. An email will be sent to DAAC staff indicating the error that occurred and that a SYNC package was exported. The BMGT package will be placed in the "COMPLETE_WITH_ERRORS" state.

Response Code: NO_OBJECT_REEXPORT_SYNC

4.11.6.9 Additional Monitor Tasks

The Monitor performs additional (monitoring) tasks, as follows:

If the Monitor detects that an Ingest Summary Report for a package has not been received from ECHO within a configurable time frame, it will send a warning email to the operator. Additionally, it will update the export cycle state to indicate that the email has been sent (and thus prevent it from happening again).

If the Monitor detects that a package has not begun transfer to ECHO within a configurable time frame after being started, it will send a warning email to the operator. Additionally, it will update the export cycle state to indicate that the email has been sent (and thus prevent it from happening again).

The Monitor periodically runs a cleanup task in the database to remove the audit trail entries for completed export cycles older than a configurable number of days.

The Monitor polls the database to look for cleanup export cycles which have been requested by DataPool Cleanup, but not yet initiated. If one is found, Monitor will select datapool delete events to be exported, and flag the cycle for generation.

Refer to EED/SIPS ICD - 423-41-57, EED/ECHO Metadata Inventory ICD (this is still a work-in-progress artifact) and BMGT whitepaper - 170-WP-023-007 to better understand interfaces and OPS concept.

For reference, ECHO schemas are included in Section 4.11.2.13.

4.11.6.10 BMGT CSCI Context

BMGT consists of only one CSCI. Therefore, the Subsystem Context in Figure 4.11-1 represents the BMGT CSCI Context, and it will not be replicated here.

4.11.6.11 BMGT CSCI Process Interface Description

BMGT consists of only one CSCI. Therefore, the interface description in Table 4.11-1 represents the BMGT CSCI Process Interface, and it will not be replicated here.

4.11.6.12 Data Stores

BMGT uses AIM and Datapool databases as well as the StorNext XML archive to generate its products. Table 4.11-6 describes the Data Stores.

Table 4.11-6. Data Store

Data Store	Type	Description
Inventory DB	Sybase	BMGT reads required Metadata info from Inventory DB.
Data Pool DB	Sybase	BMGT reads required DPL URL and cloud cover Info from Data Pool DB.
Small file archive	StorNext file system	BMGT reads granule and collection metadata files from the XML archive, applies XSLT stylesheets to them, and inserts the resulting XML into its output products.

4.11.6.13 BMGT GUI Functional Overview

The BMGT GUI is a JSF based web GUI, designed along the lines of the DPL Ingest GUI. It will allow the operator to monitor the generation and export of BMGT packages (Automatic, Manual, Corrective and Cleanup).

The GUI provides DAAC staff with the following functions:

- Display BMGT export processes that are currently in progress
- Monitor the status of the BMGT FTP service function that FTPs products to ECHO

- Allow the operator to suspend / resume FTP of products to ECHO
- List the N most recent export packages and view detailed information about them, where N is configurable by the DAAC staff
- Cancel an export package;
- List the N most recently completed packages which resulted in errors and view detail information about them, where N is configurable by DAAC staff.
- View and change BMGT configuration parameters, except for configuration items such as collection group / collection mapping that must be specified in XML configuration files. Changing the BMGT runtime configuration parameters will be restricted to DAAC staff that is logged in as BMGT administrator
- View BMGT Monitor policies for handling errors returned from ECHO.
- View the contents of the ReExport Queue.
- Display global alerts upon a configured number of FTP to ECHO failures
- Display the status of incremental verification on a system, group, and collection level and reset the incremental verification of a particular collection.
- Display and remove alerts flagged by repeatedly failing BMGT packages.

4.11.6.14 ECHO Metadata Schemas

The ECHO schemas are very large and would not conveniently fit in to this document. The latest schema currently is use in the operational ECHO environment can be found at the URLs listed below.

4.11.6.14.1 Collection.xsd

See http://www.echo.nasa.gov/ingest/schemas/operations/Collection.xsd

4.11.6.14.2 Granule.xsd

See http://www.echo.nasa.gov/ingest/schemas/operations/Granule.xsd

4.11.6.14.3 Browse.xsd

See http://www.echo.nasa.gov/ingest/schemas/operations/Browse.xsd

4.11.6.14.4 MetadataCommon.xsd

See http://www.echo.nasa.gov/ingest/schemas/operations/MetadataCommon.xsd

Abbreviations and Acronyms

A

ABC++	Document Generator used to provide class level detail
ACL	Access Lists
ACMHW	Access and Control Management Hardware (Configuration Item)
AD	Advertisement
ADC	Affiliated Data Center (National Oceanic and Atmospheric Administration only)
AGS	ASTER Ground System
AIT	Algorithm Integration and Test
AIM	Archive Inventory Management
AITHW	Algorithm Integration and Test Hardware (Configuration Item)
AI&T	Algorithm Integration and Test
AITTL	Algorithm Integration and Test Tools (Computer Software Configuration Item)
ALOG	Applications Log
AM-1	See TERRA (spacecraft)
AOI	Area of Interest
AOS	ASTER Operations Segment
AP	Algorithm Package
APC	Access/Process Coordinators
API	Application Program Interface
AQA	Algorithm Quality Assurance
AQUA	PM-1 Satellite (AIRS, AMSR-E, AMSU, CERES, HSB, MODIS)
AR	Action Request
AS	Administration Stations
ASCII	American Standard Code for Information Interchange
ASE	Adaptive Server Enterprise

ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
ATM	Asynchronous Transfer Mode
AURA	NASA mission to study the earth's ozone, air quality and climate (formerly the CHEM mission)

B

BBR	Browse metadata (conforming to the ECS/ECHO Metadata Inventory ICD)
BCP	Bulk Copy Program
	Bulk Copy Procedure
BDS	Bulk Data Server
BMGT	ECS Bulk Metadata Generator Tool
BLM	Baseline Manager

C

CAD	Computer Aided Design
CCB	Change Control Board (Raytheon Convention)
	Configuration Control Board (NASA Convention)
CBLM	ClearCase Baseline Manager
CCDI	ClearCase DDTS Integration
CCR	Configuration Change Request
CDE	Common Desktop Environment
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CD-ROM	Compact Disk - Read Only Memory
CDS	Cell Directory Service
CFG	Configuration File
CGI	Common Gateway Interface
CHUI	Character-based User Interface
CI	Configuration Item
CLI	Command Line Interface
CLS	Client Subsystem
CM	Configuration Management

CMI	Cryptographic Management Interface
CMP	Configuration Management Plan
CN	Change Notice
CO	Contracting Officer
COTS	Commercial Off the Shelf (Software or Hardware)
CPF	Calibration Parameter File
CPU	Central Processing Unit
CRM	Change Request Manager
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSMS	Communications and Systems Management Segment (ECS)
CSS	Communications Subsystem

D

DAAC	Distributed Active Archive Center
DADS	Data Archive and Distribution System
DAO	Data Assimilation Office
DAP	Delivered Algorithm Package
DAS	Dual Attached Station
DB	Database
DBMS	Database Management System
DCCI	Distributed Computing Configuration Item
DCN	Document Change Notice
DDICT	Data Dictionary (Computer Software Configuration Item)
DDR	Detailed Design Review
	Data Delivery Record (same as a Product Delivery Record)
DDT	DAAC Distribution Technician
DDTS	Distributed Defect Tracking System (COTS)
DEM	Digital Elevation Model
DESKT	Desktop (Computer Software Configuration Item)
DEV	Custom Developed Code

DFS	Distributed File System
DID	Data Item Description
DIPHW	Distribution and Ingest Peripheral Hardware Configuration Item
DLL	Dynamic Link Library
DLT	Digital Linear Tape
DM	Data Management
DMGHW	Data Management Hardware (Configuration Item)
DMS	Data Management Subsystem
DNS	Domain Name Service
DOF	Distributed Object Framework
DORRAN	Distributed Ordering, Researching, Reporting, and Accounting Network (At EDC)
DP	Data Provider
DPAD	DataPool Action Driver
DPL	Data Pool Subsystem
DPR	Data Processing Request
DPRID	Data Processing Request Identifier
DPREP	Data Pre-Processing
DR	Data Repository
DRPHW	Data Repository Hardware (Configuration Item)
DSC	Development Solution for the C programming language
DSS	Data Server Subsystem
DTD	Document Type Definition
DTF	Sony DTF Tape cartridge system (replacement for the D3 tape cartridge system)
DTS	Distributed Time Service

E

EBIS	EMD Baseline Information System
ECHO	ECS ClearingHouse
ECN	Engineering Change Notice
ECS	Earth Observing System Data and Information Core System

EDC	Earth Resource Observation System (EROS) Data Center
EDF	ECS Development Facility
EDG	EOS Data Gateway
EDHS	ECS Data Handling System
EDN	Expedited Data Set Notification
EDOS	Earth Observing System Data and Operations System
EDR	Expedited Data Set Request
EDS	Expedited Data Set
EC	Error conditions (in tickets)
EED	EOSDIS Evaluation and Development
EGS	EOS Ground System
EISA	Enhanced Industry Standard Architecture
E-mail	Electronic Mail (also Email, e-mail, and email)
EMD	EOSDIS Maintenance and Development Project
EMOS	ECS Mission Operations Segment (formerly FOS)
EMSn	EOSDIS Mission Support network
EOC	Earth Observing System Operations Center
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
EPD	External Product Dispatcher
EROS	Earth Resource Observation System
ESDIS	Earth Science Data and Information System (GSFC Code 505)
ESDT	Earth Science Data Type
ESRI	Environmental Systems Research Institute
ETM+	Enhanced Thematic Mapper Plus (Landsat 7)
EWOC	ECHO WSDL Order Component

F

FC	Functional components (capabilities in tickets)
FCAPS	Fault, Configuration, Accountability, Performance, and Security services
FDS	Flight Dynamics System

FH	Fault Handling
FLDB	Fileset Location Database
F&PRS	Functional and Performance Requirements Specification
FSMS	File and Storage Management System
FTP	File Transfer Protocol
FTPD	File Transfer Protocol Daemon

G

GB	gigabyte (10^9)
Gb	gigabit (10^9)
GCDIS	Global Change Data and Information System
GCMD	Global Change Master Directory (not developed by ECS)
GFE	Government Furnished Equipment
GLAS	Geoscience Laser Altimeter System
GSFC	GODDARD Space Flight Center (NASA facility and DAAC)
GSMS	Ground System Management Subsystem (ASTER)
GTWAY	(ASTER) Gateway (Computer Software Configuration Item)
GUI	Graphical User Interface

H

HDF	Hierarchical Data Format
HDF-EOS	an EOS proposed standard for a specialized HDF data format
HEG	HDF-EOS To GeoTIFF
HMI	Human Machine Interface
HSB	Humidity Sounder for Brazil
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
HW	Hardware
HWCI	Hardware Configuration Item

I

IAS	Image Assessment System
IBM	International Business Management

ICESat	Ice, Cloud and Land Elevation Satellite
ICD	Interface Control Document
ID	User Identification (or Identifier)
IDG	Infrastructure Development Group
IDL	Interactive Data Language
I/F	Interface
IGS	International Ground Station (Landsat 7)
IHCI	Internetworking hardware configuration item
IIU	DSS Inventory Insert Utility
ILG	Infrastructure Library Group
ILM	Inventory, Logistics, Maintenance (ILM) Manager
IMS	Information Management System (ECS element name)
INHCI	Internetworking HWCI
I/O	Input/Output
IOS	Internetwork Operating System
IP	InterProcess Communication
IPC	Inter-Process Communication
IRD	Interface Requirements Document
IRR	Incremental Release Review
ISIPS	ICESat Science Investigator-Led Processing System
ISO	International Standards Organization
ISR	ECHO Ingest Summary Report
ISS	Internetworking Subsystem
I&T	Integration and Test

J

JAF	Java Activation Framework
JAXP	JAVA APL for XML Processing
JDBC	Java Database Connectivity
JDOM	Java Document Object Model
JESS	Java Earth Science Server

JEST	Java Earth Science Tool
JPL	Jet Propulsion Laboratory (DAAC)
JRE	Java Runtime Environment

K

KFTP	Kerberos File Transfer Protocol

L

L0 - L4	Level-0 through Level-4 data (ECS)
L0R	Landsat Reformatted Data
LAMS	Landsat 7 Archive Management System
LAN	Local Area Network
LaRC	Langley Research Center (DAAC)
LFS	Local File System
LZ77	Lampel-Ziv coding

M

M&O	Maintenance and Operations
MB	Megabyte (10^6)
Mbps	Megabits Per Second
MCF	Metadata Configuration File
MCI	Management Software Configuration Item (Computer Software Configuration Item)
METC	Collection Metadata (conforming to the ECS/ECHO Metadata Inventory ICD), also ECSMETC
METG	Granule Metadata (conforming to the ECS/ECHO Metadata Inventory ICD), also ECSMETG
METU	Update Metadata (conforming to the ECS/ECHO Metadata Inventory ICD), also ECSMETU
MISR	Multi-Imaging SpectroRadiometer
MLCI	Management Logistics Configuration Item (Computer Software Configuration Item)
MM	Mode Management
MMO	Mission Management Office

MMS	Mode Management Service
MOC	Mission Operations Center
MOPITT	Measurements of Pollution in the Troposphere
MP	Message Passing
MS	Mass Storage
MSCD	Mirror Scan Correction Data (file)
MSS	System Management Subsystem
MSSHW	(System) Management (Subsystem) Hardware Configuration Item
MTA	LAMS Metadata File
MTP	Distribution Product Metadata File Extension (<filename>.MTP)
MTPE	Mission to Planet Earth

N

NASA	National Aeronautics and Space Administration
NBSRV	Spatial Subscription Server
NCEP	National Centers for Environmental Predictions
NCR	Non-conformance Report
NESDIS	National Environmental Satellite, Data, and Information Service (NOAA)
NFS	Network File System
NIS	Network Information Service
NISN	NASA Integrated Services Network
NMC	National Meteorological Center (located at National Oceanic and Atmospheric Administration - NOAA)
NNTP	Network News Transfer Protocol
NOAA	National Oceanic and Atmospheric Administration
NSI	National Aeronautics and Space Administration Science Internet
NSIDC	National Snow and Ice Data Center (DAAC)

O

ODL	Object Description Language
OEA	OGC-ECHO Adaptor
OEM	Original Equipment Manufacturer

OMS	Order Manager Subsystem
OMSHW	Order Manager Subsystem Hardware
OMSRV	Order Manager Server
OMGUI	Order Manager Graphical User Interface
OOA	Object oriented analysis
OOD	Object oriented design
OODCE	Object Oriented distributed computing environment
OPS CON	Operations Concept
OS	Operating System
OSI	Open Systems Interconnect

P

PAN	Production Acceptance Notification
PC	Personal Computer
	Performance Constraints (in tickets)
PCD	Payload Correction Data (file)
PCFG	Process Configuration File
PDR	Product Delivery Record
PDRD	Product Delivery Record Discrepancy
PF	Process Framework
PI	Principal Investigator
PM-1	EOS Afternoon Equator Crossing Mission (See Aqua); Mission to study the land, oceans and the earth's radiation budget
PMPDR	Physical Media Product Delivery Record
PSA	Product Specific Attributes
PVC	Performance Verification Center

Q

QA	Quality Assurance
QDS	Quick Look Data Set (Same as Expedited Data Set)
QAUU	DSS Quality Assurance Update Utility

R

RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
RCS	Request Communications Support
RDBMS	Relational Database Management System
REL	Release
RFA	Remote File Access
RFC	Request For Comments
RIP	Routing Information Protocol
RMA	Reliability, Maintainability and Availability
RMS	Request Management Services
ROSE	Request Oriented Scheduling Engine
RPC	Remote Procedure Call
RSC	Raytheon Systems Company
RTU	Rights to Use

S

S4PM	Simple, Scalable, Script-based Science Processor for Missions
SAGE III	Stratospheric Aerosol and Gas Experiment III
SAN	Storage Area Network
SAS	Single Attached Station
SATAN	Security Administrator Tool for Analyzing Networks
SCF	Science Computing Facility
SCLI	SDSRV Command Line Interface
SCP	Secure Copy
SCSI	Small Computer System Interface
SDE	Software Development Environment
SDF	Software Development Folder
SDP	Science Data Processing
SDPTK	Science Data Processing Toolkit
	Science Data Processing Toolkit (Computer Software Configuration Item)
SDSRV	Science Data SeRVer (Computer Software Configuration Item)

SGI	Silicon Graphics, Inc.
SIM	Spectral Irradiance Monitor
SIPS	Science Investigator-Led Processing Systems
SMC	System Management Center
	System Monitoring and Coordination Center
SMP	Symmetric Multi-processor
SMTP	Simple Mail Transport Protocol
SNAC	StorNext Archive Cash
SOAP	Simple Object Access Protocol
SOLSTICE	Solar Stellar Irradiance Comparison Experiment
SORCE	Solar Radiation and Climate Experiment
SPARC	Single Processor Architecture
SPRHW	Science Processing Hardware (Configuration Item)
SQL	Structured Query Language
SQS	Spatial Query Server
SRF	Server Request Framework
SSAP	Science Software Archive Package
SSH	Secure Shell
SSIT	Science Software Integration and Test
SSS	Spatial Subscription Server Subsystem
STK	StorageTek
SW	Software
Sybase	(ECS) COTS database management product (ASE)
SYSLOG	System Log

T

TAR	Tape Archive
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TD	Technical Documents
TELNET	Telecommunications Network

TERRA	EOS AM Mission spacecraft 1, morning equator crossing spacecraft series -- ASTER, MISR, MODIS and MOPITT instruments; Mission to study the land, oceans and the earth's radiation budget
TIM	Total Irradiance Monitor
TM	Thematic Mapper (Landsat)
TT	Trouble Ticket
TTPro	TestTrack Pro

U

UDP	User Datagram Protocol
UML	Unified Modeling Language
UR	Universal Reference
URL	Universal Resource Locator
USGS	U. S. Geological Survey
UUID	Universal Unique Identifier

V

VT	Virtual terminal

W

WAN	Wide Area Network
WIST	Warehouse Inventory Search Tool
WKBCH	WorKBenCH (Computer Software Configuration Item)
WKSHW	Working Storage Hardware Configuration Item
WRS	Worldwide Reference System
WS	Working Storage
WSDL	Web Service Definition Language
WWW	World Wide Web

X

xAR	x Acquisition Request (where x is any kind of or generic acquisition request)
XBDS	Bulk Data Service Protocol
XDR	External Data Representation
XFS	Extended File System

XML	eXtensible Markup Language
XSD	XML Schema
XRU	DSS XML Replacement utility
XSLT	XML Stylesheet Language for Transformations.
XVU	DSS XML Validation Utility