625-CD-616-001

EOSDIS Core System Project

ECS Project Training Material Volume 16: Science Software Integration and Test

March 2001

Raytheon Company
Upper Marlboro, Maryland

ECS Project Training Material Volume 16: Science Software Integration and Test

March 2001

Prepared Under Contract NAS5-60000
CDRL Item 129

RESPONSIBLE ENGINEER

Elory R. McLeod /s/ 3/8/2001

Elroy R. McLeod Date
EOSDIS Core System Project

SUBMITTED BY

Gary W. Sloan /s/ 3/8/2001

Gary Sloan, M&O Manager Date
EOSDIS Core System Project

Raytheon Company
Upper Marlboro, Maryland

This page intentionally left blank.

Preface

This document is a contract deliverable with an approval code of 3. As such, it does not require formal Government approval. This document is delivered for information only, but is subject to approval as meeting contractual requirements.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Raytheon Systems Company
1616 McCormick Dr.
Upper Marlboro, MD 20774-5301

Note: This document contains change bars to indicate the addition or revision of material since the issuance of the predecessor document containing training material for Release 5B of the Earth Observing System Data and Information System (EOSDIS) Core System (ECS).

This page intentionally left blank.

Abstract

This is Volume 16 of a series of lessons containing training material for Release 6A of the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). This lesson provides a detailed description of the process required to install and integrate science software Product Generation Executives (PGEs) into the EOSDIS environment, and once installed, test the new science software to verify its operability.

***Keywords**:* training, instructional design, course objective, SSI&T

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Original
iii through xx	Original
1 through 444	Original
SSI&T Slide Presentation Part One 1 through 227	Original
ESDT Slide Presentation Part Two 1 through 118	Original

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
625-CD-616-001	Original	March 2001	

This page intentionally left blank.

Contents

Preface

Abstract

Introduction

Identification 1

Scope 1

Purpose 1

Status and Schedule 1

Organization 1

Related Documentation

Parent Document 3

Applicable Documents 3

Information Documents 3

Information Documents Referenced 3

Information Documents Not Referenced 3

Overview

Lesson Overview 7

Lesson Objectives 7

Importance 11

DAAC Science Software I&T Support Engineer 11

ECS Iteration 6A

Architecture: Overview 13
ECS Subsystems 13
Client Subsystem (CLS) 14
Interoperability Subsystem (IOS) 14
Data Management Subsystem (DMS) 14
Data Server Subsystem (DSS) 14
Ingest Subsystem (INS) 15
Data Processing Subsystem (DPS) 15
Planning Subsystem (PLS) 15
Communications Subsystem (CSS) 16
Management Subsystem (MSS) 16
Internetworking Subsystem (ISS) 17
Release 6A Capabilities 19

Science Software Configuration Management

ClearCase Overview 21
Creating a View in ClearCase 22
Setting a View in ClearCase 25
Creating a New Directory 26
Importing files into ClearCase 28
Importing Multiple Files into ClearCase 29
Checking Out a File From ClearCase 32
Checking a Modified Element into ClearCase 33
Checking a Modified Element/File into ClearCase 33

Science Software Integration and Test (SSI&T) Preparation, Processes and Setup

Key Operator Roles 35
COTS Software Tools 35

General Process 36
GENERAL 37
Science Office Project Instructions 37
Pre-SSI&T Activities 38
Formal SSI&T Activity 39
Science Software Integration & Test Flow: 42
Descriptions of the various Ids that SSIT creates 45
The Science Server Archive Package (SSAP): 45
The Current list of File List types in the SSAP code: 46
ODL File(s)/PGE Metadata: 49

Science Software Integration and Test (SSIT) Manager

SSIT Manager Overview 51
General Set Up of the SSIT Manager 59
SSIT Manager Tools 60
Using the SSIT Manager: 60

Acquiring and Unpacking the Delivered Algorithm Package (DAP)

Acquiring the Algorithm Package via FTP 63
Unpacking a DAP 65
SSIT Software Operating Instructions 66
Subscribing to the DAP: 66
DAP Insert 67
Performing a DAP Insert 67
An Example of a DAP Metadata File 68
Mail Template 73
Example of a PGE Executables Tar File Insertion Script 76

DAP Acquire

Performing a DAP Acquire Using SSIT Manager 77
An Alternative Way to Acquire a DAP 80

Inserting a Science Software Archive Package (SSAP)

Inserting a SSAP into PDPS 84

Updating a Science Software Archive Package (SSAP)

Updating a SSAP 87

Standards Checking of Science Software

Standards Checking Overview 89
Checking FORTRAN 77 ESDIS Standards Compliance 89
Checking for ESDIS Standards Compliance in Fortran 90 91
Checking for ESDIS Standards Compliance in C 93
Checking for ESDIS Standards Compliance in Ada 95
    Checking for ESDIS Standards Compliance in Ada: Verdix COTS 95
    Checking for ESDIS Standards Compliance in Ada: GNU *gcc* Compiler 96
Prohibited Function Checker 98
Checking Process Control Files 102
    Checking Process Control Files: Command-Line Version 103
Extracting Prologs 105

Compiling and Linking Science Software

Updating the Process Control Files (PCFs) 109
Setting up a SDP Toolkit Environment 112
    Setting Up the SDP Toolkit Environment 116
    An example of Compile procedures used to produce a PGE.exe: 117
Compiling Status Message Facility (SMF) Files 118

Building Science Software with the SCF Version of the SDP Toolkit

Building Science Software with the DAAC Version of the SDP Toolkit 123

Running a PGE in a Simulated SCF Environment

Setting Up the Environment for Running the PGE 127
Running and Profiling the PGE 128
Checking the PGE for Memory Leaks 131

The ECS Assistant Functionality Replaced in Part by Scripts and Monitor GUI Whazzup

Using Scripts to Start Up / Shut Down Servers 134
All Servers Start Up/Shut Down 134
Individual Servers Start Up/Shut Down 134
Monitoring ECS using WHAZZUP Web GUI 137
Using ECS Assistant to View ECS Science Data Server Database 140

Preparation of Earth Science Data Types (ESDT's)

Comparing Granule Level Metadata 145
Installing/Removing (ESDT/DLL) using the Science Data Server Operator GUI 146
Validating Successful ESDT Installation 148
Science Data Server 149
Browser to View ECS Science SDSRV Database 159
Using isql commands in the SDSRV 159
wisqlite is an isql Browser to View ECS Data Bases 160
Removing ESDT's from Archive Area using Command Line Scripts 160
Staging ESDT's for Installation 161
EcDsScienceDataServerStart TS1 StartTemperature cold 162
EcDsSdSrvGuiStart TS1 162
Re-install or Update ESDT's using the Science Data Server GUI 162
EcDsScienceDataServerStart TS1 StartTemperature maintenance 163
EcDsSdSrvGuiStart TS1 163

Production Rules

Purpose and Scope....165
Overview of Production Rules....165
Data Processing in ECS....165
Production Rule Definitions....166
Introduction....166
Input Data Specification....166
Basic Temporal....167
Example use of the Basic Temporal Rule....167
Production_Rules_Syntax....168

Production Requests

Science Software and Production Requests....169
Types of Processing....170
Production Rules Describe a PGE to ECS....171
Syntax of Production Rules....173
Basic Temporal Production Rule....176
Advanced Temporal Production Rule....184
Alternate Input and Optional Input Production Rules....187
Minimum/Maximum Number of Granules Production Rule....196
Optional DPRs Production Rule....200
Intermittent Activation Production Rule....203
Metadata Checks and Metadata Query Production Rules....205
Data Day Production Rule....213
Spatial Query Production Rule....214
Tiling Production Rule....217
Closest Granule Production Rule....222
Orbital Processing Production Rule....225
Production Planning Considerations....230
DPREP Considerations....231
Production Rules Technical Information Sources....232

DPREP

Introduction 233
SSI&T Activity for DPREP 233
DPREP Processes and Procedures Used in Testing 235
PGE Name-STEP ONE 237
PGE Name-STEP TWO.....238
PGE Name-STEP THREE 238
HowToRunDPREP.....239
DPREP Step1 profile1 PCF 252
Setups for DPREP 256
DPREP Considerations.....257
Updating the Orbit Model 261
Introduction to Updating the Orbit Model 261
PGE Chaining 262

PGE Registration and Test Data Preparation

PGE Registration 263
PGE ODL Preparation 263
ESDT ODL Preparation.....265
Example of ESDT.odl files being established in ECS.....269
AlternativeTool for SSIT Metadata Update:.....269
Example of a successful PDPS Science Metadata Update: 269
SSIT Operational Metadata Update GUI 270
Operational Metadata.....271
Test Data Preparation and Insertion of Data Granules.....274
Generating a Metadata Configuration File (Source MCF) 274
Example of a successful installation of a Source MCF:.....275
Creating a Target MCF (.met) for a Dynamic/Static Granule 276
Inserting Static Data Granules into the Science Data Server.....277
Inserting Dynamic Data Granules to the Science Data Server 279

Example of a successful insertion of a Dynamic Input Data Granule into the Data Servers: 281
Placing the Science Software Executable (SSEP) onto Science Data Server 286
Assembling a Science Software Executable Package 286
Inserting Science Software Executable Package onto Science Data Server 288
Example of a successful insertion of a SSEP EXE TAR:.....290

PGE Planning Processing and Product Retrieval

Using the Production Request Editor 293
Defining a new Production Request 296
Processing 299
Viewing a New Production Request.....299
Viewing Data Processing Requests 301
Using the Production Planning Workbench 302
Creating and Activating a Production Plan 302
Using the Planning Workbench to Run One PGE.....303
Monitoring Production 306
Using the Q/A Monitor 308

Postprocessing and General Investigation

Examining PGE Log Files.....311
Examining PDPS-Related Scripts and Message Files.....311
ECS Mechanisms for Capturing Information about a PGE's Execution.....312
ECS Production History (PH) 312
ECS Failed PGE Tar File.....313
Examining AutoSys JIL Scripts 314
Examining Application Log Files (ALOG) 315

File Comparison and Data Visualization

Using the GUI HDF File Comparison GUI....317
Using the hdiff HDF File Comparison Tool....318
Using the ASCII File Comparison Tool....319
Using the Binary File Difference Assistant....320
Data Visualization....322
Viewing Product Metadata with the EOSView Tool....322
Viewing HDF Image Objects....324
Viewing HDF-EOS Grid Objects....325
Viewing HDF-EOS Swath Objects....326
Viewing HDF SDS Objects....328
Viewing Product Data with the IDL Tool....328
Creating an Image Display Using IDL....329
Saving an Image Display Using IDL....331
Setting axis titles for a plot:....333
Saving a Plot Display Using IDL....334
Raster Mapping Fundamentals....334

Miscellaneous

Setting Up Environment for Direct PDPS Database Access....337

Examples of PGE and ESDT ODL Files for Each Instrument Team

Template ODL Files....339

Practical Exercise

Science Software Integration and Test -....341

Slide Presentation

Slide Presentation Description.....343

Appendix A. Troubleshooting and General Investigation

An Alternative Way to Install and Remove an ESDT.....345
Handling an ESDT Installation Failure.....346
Insert File Failure.....347
Handling an Insert Failure.....348
Viewing the Advertising Database Using SQL Commands - Quick-Step Procedures.....349
Acquire Failure.....350
Handling an Acquire failure:.....350
Failure During DPR Generation.....353
Handling DPR Generation Failures:.....353
Failure Scheduling a DPR.....353
Handling a DPR Scheduling failure.....353
Failures During Execution.....355
Handling a Failures During Execution.....355
PDPS Troubleshooting - The PGE Job has Failed.....356
The PGE Job has failed.....356
The Post-Execute Job has failed.....357
The PGE Job and Post-Execute Job have both failed.....357
The PGE Job has failed and the DPR has gone into "Failed-PGE" processing.....357

PDPS Troubleshooting - A single DPS job has failed or is hanging

The entire Job Box is hung.....359
The entire Job Box is Hung.....359
A DPS Allocation job is hanging.....359
A DPS Allocation job has failed.....360
A DPS Staging job is hanging.....361

A DPS Staging job has failed 361
A DPS PreProcess job has failed 361
A DPS PGE job is hung 361
A DPS PGE job has failed.....362
A DPS De-staging job has failed 362
All known problems are corrected and the re-start fails again 363
Non-PDPS servers are down 363
PDPS Troubleshooting - Job Activation Fails from the Planning Workbench 363
PDPS Troubleshooting - Input Data Problems.....364
PDPS Troubleshooting - Jobs are activated, but do not get started in AutoSys.....365
PDPS Troubleshooting - SDSRV Troubleshooting.....369

Using IQ Software to Create Reports

Creating Reports Using IQ Software 371
Formatting IQ Software Reports.....384
[TBS] 384

Deleting Granules

Deletion Capability and Features.....385
Deletion Sequence.....386
Granule Deletion Procedures.....387
Granule Deletion Client Steps.....389
Run SDSRV Deletion Cleanup Utility.....395
Delete files from the archive using the STMGT GUI.....414

Appendix B.

Learn more about SSI & T 427
ESDT Basics - http://dmserver.gsfc.nasa.gov/esdt/EsdtSection1/index.html 428
GDAAC Directory for SSI&T:http://gsfcsrvr8.gsfcmo.ecs.nasa.gov/SSIT/.....429

The ESDT Process (updated for drop 4) by Karl W. Cox, 22 December 1997 429
MODIS - Science Data Processing Software Version 2 System Description,.....429
ECSINFO: http://ecsinfo.hitc.com/iteams/Science/science.html 429
PDPS how to are located on the EDF machines at: /home/PDPS/docs/.....429
PDPS Web Page: http//dmserver.gsfc,nasa.gov/ecsdev/relb/pdps/index.html 429
PDPS Troubleshooting Techniques are located on the EDF machines at : /home/PDPS/troubleshooting/ 429
DPREP README files located at :/usr/ecs/TS1/CUSTOM/data/DPS/ 429
DPREP binary located: :/usr/ecs/TS1/CUSTOM/bin/DPS/.....429
Updating the Leap Seconds and the Earth Motions files.....429
Script Name: update_leapsec.sh.....430
Server Node Names Convention: 432
HOWTO_SSIT HELPFUL NOTES 433
Xterm format to bring up xterm windows for servers all at once 433
howto_setup_orbits_and_pathmaps.....435
howto_register_pge- prior to 6A. 436
howto_register_dpr-prior to 6A 436
Using Production Request Editor and Planning Workbench.....437
howto_Make a Production Request.....437
Abbreviations and Acronyms.....439
Glossary 443

Introduction

Identification

Training Material Volume 16 is part of a series of training material that will be used to teach M&O concepts to the M&O staff at the GSFC, LaRC, NSIDC and EDC DAACs.

Scope

Training Material Volume 16 defines the tasks required to perform Science Software Integration & Test (SSI&T). This lesson is designed to provide the operations staff with sufficient knowledge and information to satisfy all lesson objectives.

Purpose

The purpose of this training Material is to provide a detailed course of instruction that forms the basis for understanding SSI&T. Lesson objectives are developed and will be used to guide the flow of instruction for this lesson. The lesson objectives will serve as the basis for verifying that all lesson topics are contained within this Student Guide and slide presentation material.

Status and Schedule

This lesson module provides detailed information about training for **Release 6A.** Subsequent revisions will be submitted as needed.

Organization

This document is organized as follows:

Introduction:	The Introduction presents the document identification, scope, purpose, and organization.
Related Documentation:	Related Documentation identifies parent, applicable and information documents associated with this document.
Student Guide:	The Student Guide identifies the core elements of this lesson. All Lesson Objectives and associated topics are included.
Slide Presentation:	Slide Presentation is reserved for all slides used by the instructor during the presentation of this lesson.

This page intentionally left blank.

Related Documentation

Parent Document

The parent document is the document from which this ECS Training Material's scope and content are derived.

423-41-01	Goddard Space Flight Center, EOSDIS Core System (ECS) Statement of Work

Applicable Documents

The following documents are referenced within this ECS Training Material, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document:

420-05-03	Goddard Space Flight Center, Earth Observing System (EOS) Performance Assurance Requirements for the EOSDIS Core System (ECS)
423-41-02	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS)

Information Documents

Information Documents Referenced

The following documents are referenced herein and amplify or clarify the information presented in this document. These documents are not binding on the content of the ECS Training Material.609-CD-600 Release 6A Operations Tools Manual for the ECS Project

611-CD-600	Mission Operation Procedures for the ECS Project
910-TDA-022	Custom Configuration Parameters for ECS Release 6A

Information Documents Not Referenced

The following documents, although not referenced herein and/or not directly applicable, do amplify or clarify the information presented in this document. These documents are not binding on the content of the ECS Training Material.

305-CD-600	Release 6A Segment/Design Specification for the ECS Project
311-CD-600	Release 6A Data Management Subsystem Database Design and Database Schema Specifications for the ECS Project
311-CD-601	Release 6A Ingest Database Design and Database Schema Specifications for the ECS Project

311-CD-602	Release 6A Interoperability Subsystem (IOS) Database Design and Database Schema Specifications for the ECS Project
311-CD-603	Release 6A Planning and Data Processing Subsystem Database Design and Schema Specifications for the ECS Project
311-CD-604	Release 6A Science Data Server Database Design and Schema Specifications for the ECS Project
311-CD-605	Release 6A Storage Management and Data Distribution Subsystems Database Design and Database Schema Specifications for the ECS Project
311-CD-606	Release 6A Subscription Server Database Design and Schema Specifications for the ECS Project
311-CD-607	Release 6A Systems Management Subsystem Database Design and Schema Specifications for the ECS Project
311-CD-608	Release 6A Registry Database Design and Schema Specifications for the ECS Project
313-CD-600	Release 6A ECS Internal Interface Control Document for the ECS Project
334-CD-600	6A Science System Release Plan for the ECS Project
601-CD-001	Maintenance and Operations Management Plan for the ECS Project
603-CD-003	ECS Operational Readiness Plan for Release 2.0
604-CD-001	Operations Concept for the ECS Project: Part 1-- ECS Overview
604-CD-002	Operations Concept for the ECS Project: Part 2B -- ECS Release B
605-CD-002	Release B SDPS/CSMS Operations Scenarios for the ECS Project
607-CD-001	ECS Maintenance and Operations Position Descriptions
152-TP-001	ACRONYMS for the EOSDIS Core System (ECS) Project
152-TP-003	Glossary of Terms for the EOSDIS Core System (ECS) Project
211-TP-005	Transition Plan 4PX to 4PY, 4PY to 5A, and 5A to 5B for the ECS Project
220-TP-001	Operations Scenarios - ECS Release B.0 Impacts
500-1002	Goddard Space Flight Center, Network and Mission Operations Support (NMOS) Certification Program, 1/90
535-TIP-CPT-001	Goddard Space Flight Center, Mission Operations and Data Systems Directorate (MO&DSD) Technical Information Program Networks Technical Training Facility, Contractor-Provided Training Specification
220-TP-001	Operations Scenarios - ECS Release B.0 Impacts

500-1002	Goddard Space Flight Center, Network and Mission Operations Support (NMOS) Certification Program, 1/90
535-TIP-CPT-001	Goddard Space Flight Center, Mission Operations and Data Systems Directorate (MO&DSD) Technical Information Program Networks Technical Training Facility, Contractor-Provided Training Specification
170-WP-004-002	ECS Browse Granule Description
625-CD-506-002	ECS Training Volume 6, Production Planning and Processing
814-RD-600-001	Release 6A SCF Toolkit 5.2.7 Version Description Document (VDD)
910-TDA-011	ECS Custom Software File Permissions and Special Account Standard

This page intentionally left blank.

Overview

Lesson Overview

This lesson will provide you with the complete process, by which a new science algorithm from the science community is received, integrated with ECS, tested and distributed. While the basic concept for accomplishing these tasks is fairly straight forward, many staff members are utilized in order to accomplish the process.

Lesson Objectives

Overall Objective - The objective of the SSI&T lesson is to provide a detailed description of the process required to install and test Product Generation Executives (PGEs) in the EOSDIS environment.

Condition - The student will be given an operational system with supporting equipment, and the Student Guide.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 1 - The student will be given a review of Release 6A. The review will cover an overview of Architecture and the Implications for SSI&T processing.

Condition - The student will be given an operational system with supporting equipment, the Student Guide.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 2 - The student will perform routine running of Scripts to Start Up /Shut Down Servers. The using the Whazzup (GUI) will be used to monitor ECS. Selected features of ECS Assistant will be exercised as they apply to SSI&T objectives.

Condition - The student will be given an operational system with supporting equipment, the Student Guide.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 3 - The student will put text files from the DAP under configuration management using ClearCase to create a view, create a new directory , import files, check files in, and check files out.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 4 - The student will review the SSIT Preparation and Setup Procedures.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 5 - The student will perform routine running of the SSIT Manager Graphical User Interface(GUI)

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 6 - The student will acquire the delivered algorithm package (DAP) by using File Transfer Protocol (FTP) and unpack if the DAP is delivered as a tar file.

The student will be directed to review the README Text File to ascertain information that is directed to a particular PGE.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 7- The student will verify that FORTRAN 77, Fortran 90, C and Ada source files are compliant with the ESDIS Data Production Software and Science Computing Facility (SCF) standards and guidelines document and will search source files for prohibited functions. The student will also run the Process Control File (PCF) syntax checker and extract the prologs from the source code.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 8- The student will compile Product Generation Executives (PGEs) and link SCF and DAAC version of Science Data Processing (SDP) toolkits and update files by configuring the SDP Toolkit environment, updating the process control file and compiling status message facility files.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 9- The student will run a PGE executable in a simulated SCF environment and determine resource requirements for the PGE.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 10- The student will examine the PGE produced log files from the run in the simulated SCF environment. Refer to the procedures in the Postprocessing and General Investigation section.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 11 - The student will perform Hierarchical Data Format (HDF), ASCII, and binary file comparison using the GUI and/or hdiff and will use EOSView to examine data products. Refer to the procedures in the File Comparison and Data Visualization section.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 12- The student will be able to identify and register Earth Science Data Types (ESDTs) for use on ECS.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 13 - The student will insert a Science Software algorithm package (SSAP) into the Planning and Data Processing System (PDPS). Review also the procedure for updating a SSAP.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 14- The student will review the Production Rules and Requests for compliance with the particular PGE being worked on and also DPREP as an additional PGE effort where necessary.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 15- The student will update the PGE, ESDT, and Orbit ODL Templates as necessary to comply with the specific PGE being prepared for. Production Rules identified for this particular PGE should also be provided while updating the ODL files. An adjunk section, earth Science Data Types (ESDTs) slide presentation is provided for those needing additional ESDT understanding. This is the final section of the SSI&T slide presentation.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 16- The student will update the PDPS Database by inserting science ESDT metadata, science PGE metadata and operational metadata into it and will update the data server by inserting into science data granules, static files and executables.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 17- The student will subscribe to input and output data granules; submit a Production Request (PR) to run a PGE; use the Planning Workbench to plan, schedule and activate the PR; and monitor the processing under AutoSys.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 18- The student will use the procedures including the QA Monitor (GUI) covered in the Post Processing and General Investigation Section, to locate and examine data products and the Production History File. The student will learn how to perform problem tracking on science software and ECS software using DDTS.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 19- The student will review the procedures including Appendix A., Troubleshooting and General Investigation.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Specific Objective 20- The student will review the procedures included in Appendix B. This section includes many aspects of SSI&T that may apply in carrying out the work.

Condition - The student will be given an operational system with supporting equipment.

Standard - The student will use the tools in accordance with prescribed methods and complete required procedures without error.

Importance

The Science Software Integration and Test lesson will provide a review of the process that allows the SSIT team, Production Planner and Production Monitor to retrieve, install, check, compile, test and monitor science software.

DAAC Science Software I&T Support Engineer

Provide DAAC SSI&T execution support, ECS tool and system expertise and science S/W processing problem support. Provide support to scientists in the development and integration of science software for both updates and new science software into the DAAC ECS system.

Interfaces

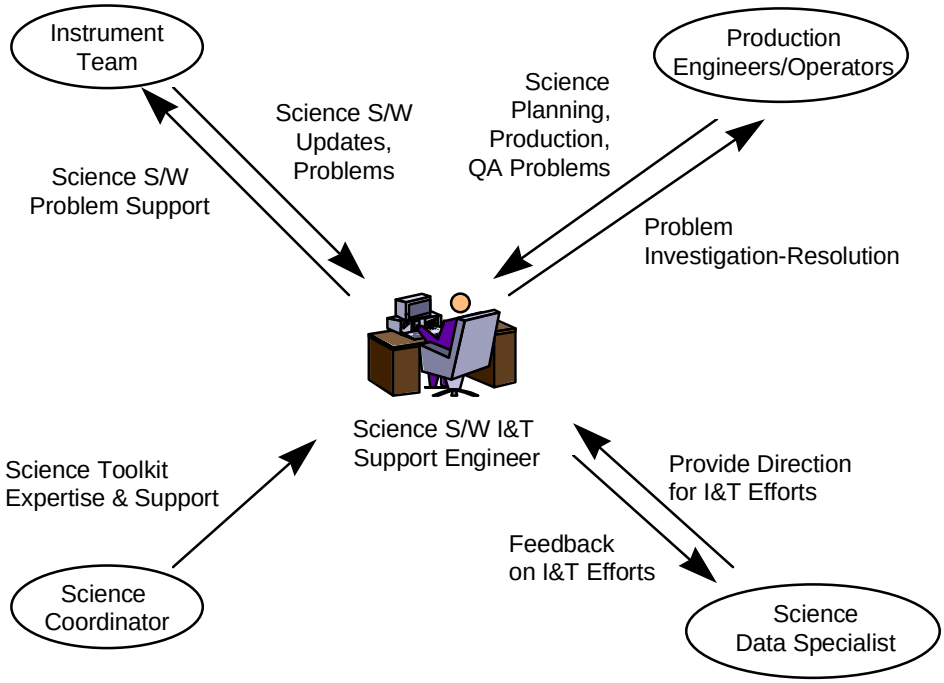


Figure 1. DAAC Science Software I&T Support Engineer Interfaces

Roles and Responsibilities

1. Provide support to Instrument Teams for the development and integration of science software into the DAAC ECS system.
2. Perform standards checking on all delivered software, including source code, scripts, process control files and related documentation.
3. Provide support for metadata updates and additions for science data products.
4. Support science processing problem investigation and resolution.
5. Recommend - assess, develop, implement changes to science toolkit software.
6. Provide support for integration and test of new and modified toolkit functions into the science software.
7. Support and track configuration of delta changes to science software, calibration coefficient files, relevant input files and documentation.
8. Facilitate migration and integration of major and minor modified versions of the science software into production versions.
9. Assess impacts and support integration and test for production planning and processing software changes. **(Planning Workbench GUI)**
10. Provide feedback and receive direction on I&T efforts.
11. Recommend-assess improvements for automated tools for SSI&T activities, such as file comparison and viewing data.
12. Write and implement procedures to examine non-standard auxiliary files and files not in HDF EOS format.

ECS Iteration 6A

The process of SSI&T or integration of EOS Instrument Science Software into the ECS has been developed and refined over three iterations of ECS. IR1, the Pre-Release B Testbed and Release 6A, Verson 2.0. The latter will be the at-launch system supporting EOS-AM1 instruments. Although every attempt has been made to keep integration procedures for science software consistent through succeeding releases, basis architectural differences have led to significant variances. This section describes the architecture of the last iteration of ECS.

Architecture: Overview

The ECS architecture can be grouped into the following four categories:

Data storage and management is provided by the Data Server Subsystem (DSS), with the functions needed to archive science data, search for and retrieve archived data, manage the archives, and stage data resources needed as input to science software or resulting as output from their execution. The Data Server Subsystem provides access to earth science data in an integrated fashion through an Application Programming Interface that is common to all layers.

Information search and data retrieval is provided by the science user interface functions in the Client Subsystem (CLS), by information search support functions in the Data Management Subsystem (DMS), and by capabilities in the Interoperability Subsystem (IOS) which assist users in locating services and data.

Data processing is provided by the Data Processing Subsystem (DPS) for the science software; and by capabilities for long and short term planning of science data processing, as well as by management of the production environment provided by the Planning Subsystem (PLS). Routine data processing and re-processing will occur in accordance with the established production plans. In addition ECS will provide “on-demand processing”, where higher level products are produced only when there is explicit demand for their creation.

Data ingest is provided by the Ingest Subsystem (INS), which interfaces with external applications and provides data staging capabilities and storage for an approximately 1-year buffer of Level 0 data (so that reprocessing can be serviced from local storage).The number of external interfaces which ECS will have is potentially very large, and the interfaces can serve very diverse functions, such as high-volume ingest of level 0 data and low-volume ingest of data from field campaigns.

ECS Subsystems

The following sub-sections provide brief overviews for each of these subsystems. More detailed discussions of their design breakdown can be found in 305-CD-020-002.

Client Subsystem (CLS)

The Client provides users with an interface through which they can access ECS services and data. It also gives science software access to the ECS services, as well as direct access to ECS data. Access is provided through graphic user interface (GUI) application tools for displaying the various kinds of ECS data (e.g., images, documents, tables), and libraries representing the client APIs to ECS services. The client subsystem follows an object-oriented design. The design is built around a core set of 'root' objects from which all other software will inherit its behavior.

Interoperability Subsystem (IOS)

The Interoperability subsystem provides an advertising service. It maintains a database of information about the services and data offered by ECS, and provides interfaces for searching this database and for browsing through related information items. For example, ESDTs are made visible through the advertising service. The Client Subsystem provides the user interface which enables access to the IOS.

Data Management Subsystem (DMS)

The Data Management subsystem provides three main functions:

Provide end-users with a consolidated logical view of a distributed set of data repositories.

Allow end-users to obtain descriptions for the data offered by these repositories. This also includes descriptions of attributes about the data and the valid values for those attributes.

Provide data search and access gateways between ECS and external information systems.

Data Server Subsystem (DSS)

The Data Server subsystem provides the management, cataloging, access, physical storage, and distribution functions for the ECS earth science data repositories, consisting of science data and their documentation. The Data Server provides interfaces for other ECS subsystems, which require access to data server services. The Data Server Subsystem consists of the following principal design components:

Database Management System - The Data Server subsystem will use database technology to manage its catalog of earth science data, and for the persistence of its system administrative and operational data.

Document Management System - Web server and database technology are used to implement a document management system to provide storage and information retrieval for guide documents, science software documentation, and ECS earth science related documents.

Data Type Libraries - The Data Server will use custom dynamic linked libraries (DLLs) to provide an extensible means of implementing the variety of ECS earth science data types and services, and will provide a consistent interface for use by other ECS subsystems requiring access to those services and data.

File Storage Management System - This component provides archival and staging storage for data.

Distribution System - The Data Server provides the capabilities needed to distribute bulk data via electronic file transfer or physical media.

Ingest Subsystem (INS)

This subsystem deals with the initial reception of all data received at an EOSDIS facility and triggers subsequent archiving and processing of the data. The ingest subsystem is organized into a collection of software components (e.g., ingest management software, translation tools, media handling software) from which those required in a specific situation can be readily configured. The resultant configuration is called an ingest client. Ingest clients can operate on a continuous basis to serve a routine external interface; or they may exist only for the duration of a specific ad-hoc ingest task. The ingest subsystem also standardizes on a number of possible application protocols for negotiating an ingest operation, either in response to an external notification, or by polling known data locations for requests and data.

Data Processing Subsystem (DPS)

The main components of the data processing subsystem - the science algorithms or Product Generation Executives (PGEs) - will be provided by the science teams. The data processing subsystem provides the necessary hardware resources, as well as a software environment for queuing, dispatching and managing the execution of these algorithms. The processing environment will be highly distributed and will consist of heterogeneous computing platforms. The AutoSys COTS tool is used as the scheduling engine. The tool is designed to manage production in a distributed UNIX environment. The DPS also interacts with the DSS to cause the staging and de-staging of data resources in synchronization with processing requirements.

Planning Subsystem (PLS)

The Planning Subsystem provides the functions needed to plan routine data processing, schedule on-demand processing, and dispatch and manage processing requests. The subsystem provides access to the data production schedules at each site, and provides management functions for handling deviations from the schedule to operations and science users. The Planning subsystem provides several functions to account for:

A processing environment which eventually will be highly distributed and consist of heterogeneous computing platforms

Existence of inter-site and external data dependencies

Dynamic nature of the data and processing requirements of science algorithms

Need for high availability

Providing a resource scheduling function which can accommodate hardware technology upgrades

Support for on-demand processing (as an alternative to predominantly routine processing)

Ability to provide longer-term (e.g., monthly) processing predictions as well as short term (e.g., daily) planning and scheduling

Communications Subsystem (CSS)

The CSS helps manage the operation of distributed objects in ECS, by providing a communications environment. The environment allows software objects to communicate with each other reliably, synchronously as well as asynchronously, via interfaces that make the location of a software object and the specifics of the communications mechanisms transparent to the application.

In addition, CSS provides the infra-structural services for the distributed object environment. They are based on the Distributed Computing Environment (DCE) from the Open Software Foundation (OSF). DCE includes a number of basic services needed to develop distributed applications, such as remote procedure calls (rpc), distributed file services (DFS), directory and naming services, security services, and time services.

Note: For Release 6A, DCE has been replaced with a Socket Scripts. These have been development to replicate those functions that DCE provided.

Finally, CSS provides a set of common facilities, which include legacy communications services required within the ECS infrastructure and at the external interfaces for file transfer, electronic mail, bulletin board and remote terminal support. The Object Services support all ECS applications with inter-process communication and specialized infra- structural services such as security, directory, time, asynchronous message passing, event logging, lifecycle service, transaction processing and World Wide Web (WWW) service.

Management Subsystem (MSS)

The Management Subsystem (MSS) provides enterprise management (network and system management) for all ECS resources: commercial hardware (including computers, peripherals, and network routing devices), commercial software, and custom applications. With few exceptions, the management services will be fully decentralized, such that no single point of failure exists.

MSS provides two levels of an ECS management view: the local (site/DAAC specific) view, provided by Local System Management (LSM), and the enterprise view, provided by the Enterprise Monitoring and Coordination (EMC) at the SMC. Enterprise management relies on the collection of information about the managed resources, and the ability to send notifications to those resources. For network devices, computing platforms, and some commercial of the shelf software, MSS relies on software called “agents” which are usually located on the same device/platform and interact with the device’s or platform’s control and application software, or the commercial software product. However, a large portion of the ECS applications software is custom developed, and some of this software - the science software - is externally supplied. For these components, MSS provides a set of interfaces via which these components can provide

information to MSS (e.g., about events which are of interest to system management such as the receipt of a user request or the detection of a software failure). These interfaces also allow applications to accept commands from MSS, provided to MSS from M&O consoles (e.g., an instruction to shut down a particular component). Applications that do not interact with MSS directly will be monitored by software, which acts as their "proxies". For example, the Data Processing Subsystem (DPS) acts as the proxy for the science software it executes. DPS notifies MSS of events such as the dispatching or completion of a PGE, or its abnormal termination.

MSS uses HP OpenView as the centerpiece of its system management solution. The information collected via the MSS interfaces from the various ECS resources is consolidated into an event history database, some on a near real-time basis, some on a regular polling basis (every 15-to 30 minutes) as well as on demand, when necessitated by an operator inquiry. The database is managed by Sybase, and Sybase query and report-writing capabilities will be used to extract regular and ad-hoc reports from it. Extracts and summaries of this information will be further consolidated on a system wide basis by forwarding it to the SMC (also on a regular basis).

MSS provides fault and performance management and other general system management functions such as security management (providing administration of identifications, passwords, and profiles); configuration management for ECS software, hardware, and documents; Billing and Accounting; report generation; trending; request tracking; and mode management (operational, test, simulation, etc.).

Internetworking Subsystem (ISS)

The ISS provides local area networking (LAN) services at ECS installations to interconnect and transport data among ECS resources. The ISS includes all components associated with LAN services including routing, switching, and cabling as well as network interface units and communications protocols within ECS resources.

The ISS also provides access services to link the ECS LAN services to Government-furnished wide-area networks (WANs), point-to-point links and institutional network services. Examples include the NASA Science Internet (NSI), Program Support Communications Network (PSCN), and various campus networks "adjoining" ECS installations. See Figure 2, which depicts an ECS Communications/Internetworking Subsystems

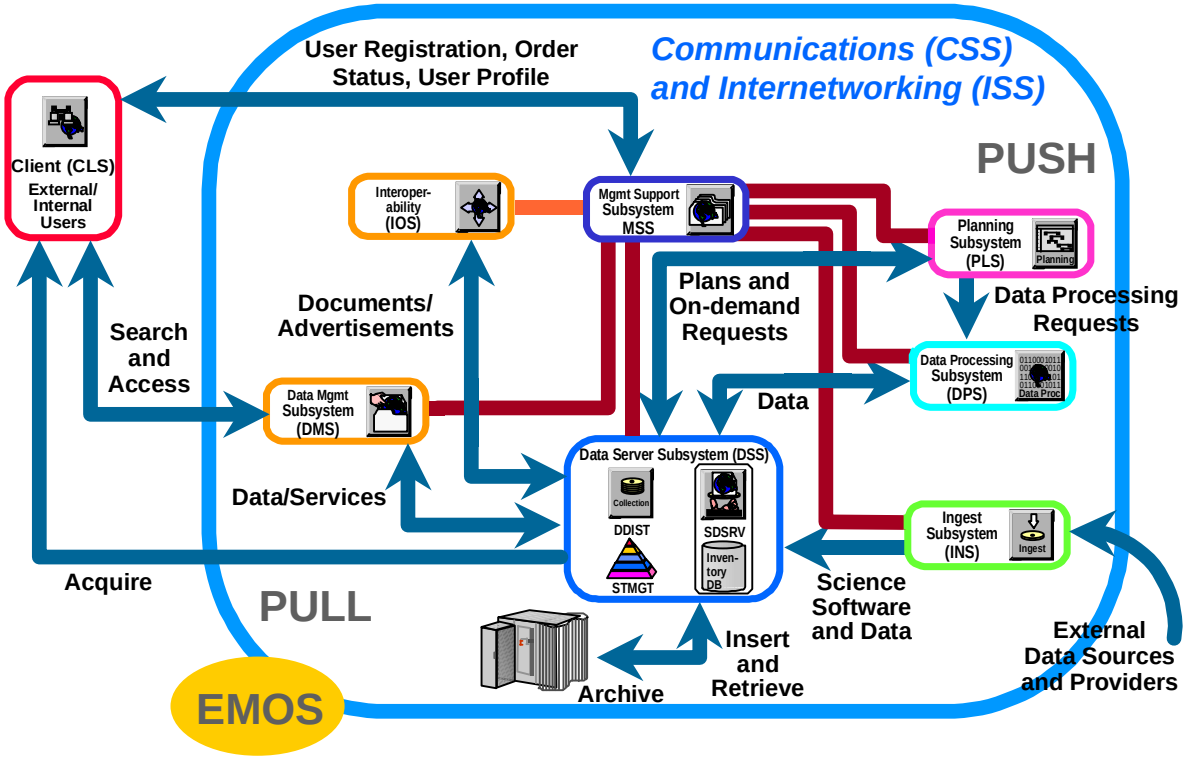


Figure 2. Diagram of ECS Communications /Internetworking Subsystems

SSI&T Functions

Table 1. List the major Functions that will be encountered by SSI&T staff. These architectural functions characterize the systems.

Table 1. Major SSI&T Functions

Function	Release 6A
System Operation	All servers must run and communicate with each other; bring up manually, or use ECS Assist tool.
Ingest Ancillary Data Granules	Ingest GUI, ESDTs must be visible to ADV server.
ESDT Insert	Use Ingest
ESDT Verification	verify through ADV
DAP, SSAP Insert	Use Ingest
PDPS Database Population	More attributes, production rules
PGE Operation	When all data is available; DPR activated. Granule Deletion Reprocessing Complex chaining through production rules.
File Access	verify presence through ADV; ftp from SDSRV; access to multiple sites
Multi-file Granule Support	Files inserted together, accessed as a single granule.
Subscription Management	Subscription Manager

Release 6A Capabilities

The Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS) capabilities are developed in terms of formal releases. Release 6A, which is controlled by Configuration Management, provides capabilities to support the ingest and archive of raw data obtained from the EOS AM 1 mission spacecraft, morning equator crossing spacecraft series (Terra (AM-1)), EOS PM 1 mission spacecraft, afternoon equator crossing spacecraft series (Aqua (PM-1)) and the Land Remote-Sensing Satellite (Landsat 7). Other capabilities provided by Release 6A 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 6A unique capabilities include:

- support the regeneration and archive of products previously produced and archived
- support ingest, production, storage, and distribution of Terra and Aqua instrument data
- support writing files to CD-ROMs and DLT tape drives for distribution
- support data compression (Unix and Gzip) for distribution
- provide access to non-science data collections by a limited number of attributes and values
- provide Science Data Server scalability/operability improvements
- support FTP Pull Subscription
- provide machine-to-machine gateway support between SIPS and ECS for data orders
- provide capability for operator deletion of granules
- support EOS Data and Operations System (EDOS) backup by distribution of replacement data to EDOS on D3 tapes and ingesting of backup data on D3 tapes from EDOS
- archive and distribution request management by multi-host scheduling
- ingest browse and metadata from IGS tapes
- allow users to request a data processing request associated with a DAR at the time of submittal

A more detailed overview of the Release 6A ECS may be found in the Release 6A Segment/Design Specifications for the ECS Project, 305-CD-600-001.

Science Software Configuration Management

The CM Administrator and System Administrator are key players in the SSI&T process. The CM Administrator receives the science software from the Science Data Specialist, places these files into a directory and request that the System Administrator place the files under configuration control by using the ClearCase tool. The science software is then tested by the SSI&T team and once the science software has successfully been tested, and upon direction from the CCB, the files are distributed to the Production Planner for placement on production server.

The CM and System Administrator need a good understanding of the ClearCase tool. ClearCase will be used to create a view, create a new directory, import files into the temporary subdirectories, and check-in and check-out files.

ClearCase Overview

All data managed under ClearCase are stored in Versioned Object Bases (VOBs), which are the "public" storage areas and Views, which are the "private storage areas. VOBs are data structures that can only be created by the CM administrator using the mkvob ("make vob") command. A VOBs is mounted as a file system and when viewed through a view, it appears as a standard UNIX directory tree structure. This file system, accessed through its mount point, has a version-control dimension that contains file elements and versions of file elements. Once reviewed, the System Administrator will place these files under configuration control. In order to accomplish this task, a view must be created in ClearCase. A view is necessary in order to make visible and accessible files and directories that have been checked in to a VOB.

Data that are under configuration management in ClearCase are said to be "checked in". In order to alter a checked-in data element (e.g. a file) to make a newer version of it, the data element must first be "checked out". Once the change has been made to the checked- out version, it is checked in again. The VOB will then contain both versions of the data element and either can be retrieved at a later date.

In general, executable binary files, object files, and data files should not be checked into ClearCase. Binary and object files are not stored efficiently in ClearCase; data files for software may be extremely large and a VOB is typically not sized for this.

Files that should be checked into ClearCase include source code, scripts, makefiles, assorted build and run scripts, documentation and other ASCII files.

A Versioned Object Base is defined by the following characteristics:

A mountable file system which stores version-controlled data, such as source files, binary files, object libraries, WYSIWYG documents, spreadsheets and anything that can be stored in the UNIX file system.

Can be mounted on some or all workstations

Several VOBs may exist on a machine or on different machines on a network.

When mounted as a file system of type MFS, a VOB can be accessed with standard UNIX and ClearCase tools.

The ClearCase file system is transparent.

Created by the CM administrator

A VOB is comprised of:

Storage area for versioned files, derived objects and cleartext files.

Database (live, shadow and log file).

Creating a View in ClearCase

In order to make files and directories that are in a ClearCase VOB visible and accessible a ClearCase view must set. A ClearCase view need only be created once. Once created, the view can be set at the beginning of each user session. Multiple views for a single user may be created.

In order for the SSI&T tools under the SSIT Manager to have access to the ClearCase VOB, the ClearCase view must be set before the SSIT Manager is run.

A view is defined by the following characteristics:

A working context for an individual developer or closely coordinated group.

Can be used to access any VOB or multiple VOBs.

Selects versions of VOB directories and files to display.

Allows developer to work without interfering with other developers.

Not a set of files but a way of seeing shared elements.

Each user may have multiple views for new development, bug fixing or porting activities.

A view is comprised of:

View storage area (typically in a local machine) - private storage for checked-out files, derived objects and private files.

Configuration Specification - set of rules that determine the version of a file the view will see.

View-tag - Name given to the view (ex. angies_view), view-tags are registered in /urs/adm/atria/view_tags.

Objects stored in a view:

Checked-out versions of file elements.

Unshared derived objects.

The ClearCase procedures can either be run from the UNIX command line or from the File Browser Screen. The SSI&T Training will only cover the UNIX command line procedures. The corresponding GUI procedures are included in the Training Material for future reference.

The following procedure not only will create a view, but will also allow creation of a subdirectory where new science software files may be stored.

Assumptions:

ClearCase is available.

A Versioned Object Base (VOB) has been created.

Creating a View in ClearCase Using Command Lines

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

2 Enter the **password** then press the **Enter** key.

3 At a UNIX prompt type **cleartool lsview**, then press the **Enter** key.

The lsview command displays the pathname to the storage location of the views.

4 At a UNIX prompt type **cleartool mkview -tag *ViewName ViewPath/ViewName.*vws**, then press the **Enter** key.

The ***ViewPath*** is the full path to the directory where views are stored.

The ***ViewName*** is the user selected name for the view. The file name for the view must end in **".vws"**.

5 Displays the directory name of the current VOB, just below the toolbar.

6 Displays the content of the directory in the space below the directory's name.

For future reference, the corresponding ClearCase GUI procedures are included in the following section. Selecting a view listed in the View Tag Browser screen brings up the File Browser, or main screen, shown in Figure 3.

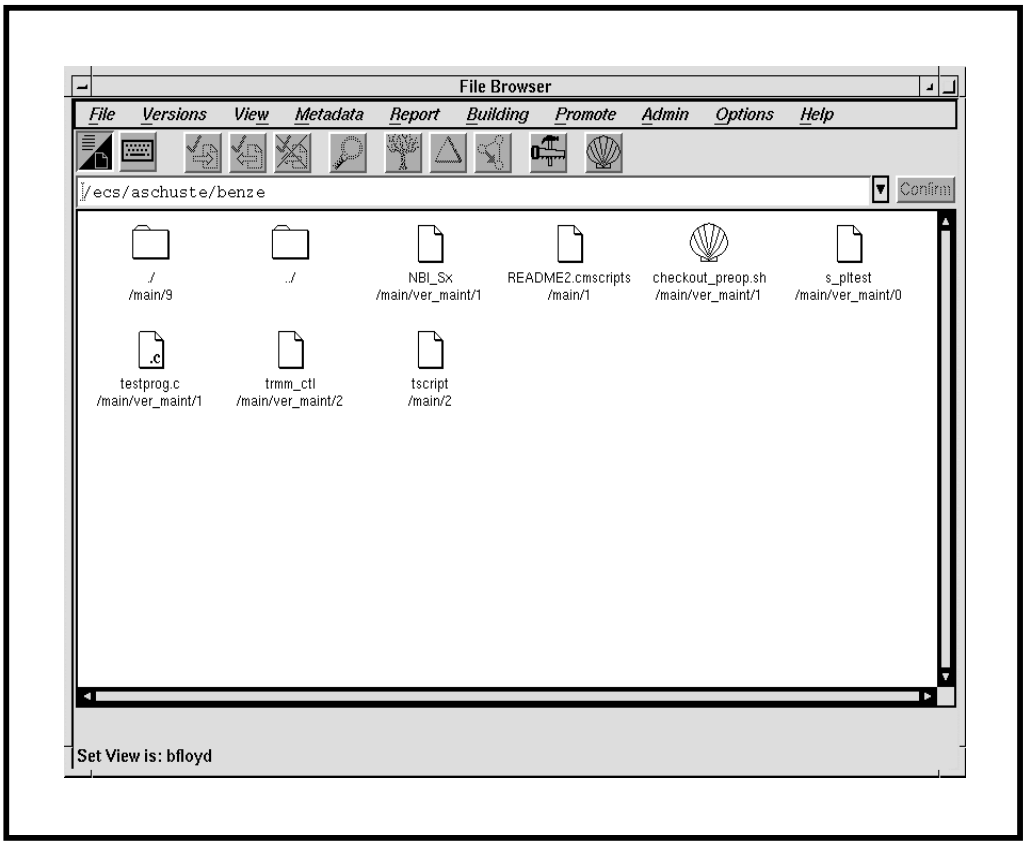


Figure 3. ClearCase File Browser Screen (Main Screen)

Creating a View in ClearCase using the File Browser Screen

1 The user should log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

Cursor moves to the Password field.

2 Type the **password** then press the **Enter** key.

3 Invoke ClearCase by typing **xclearcase &** on the UNIX command line then press the **Enter** key.

The ClearCase Transcript screen is displayed as the View Tag Browser loads.

The ClearCase View Tag Browser screen is displayed listing available views.

4 To create a view for checking in the software change package, select a known View and press the **Enter** key.

The File Browser window is displayed.

5 Select **File→Execute→Single Command.**

The String Browser window is displayed.

The prompt Enter shell command to run is displayed.

6 Invoke the make view command by typing **mkview [filename]** on the UNIX command line and press the **Enter** key.

The tempdisp window appears.

The View [filename] Created Successfully and the Cache Updated for View [filename] prompts are displayed.

7 Close the **tempdisp** window by clicking on the window and press the **Enter** key.

The tempdisp window closes.

8 Select **View →List** from the menu.

The View Tag Browser is displayed.

9 Find the new view by scrolling through the list until the new view is observed.

Setting a View in ClearCase

In order to make files and directories that are in a ClearCase VOB visible and accessible, a ClearCase view must be set. Only one view can be set (active) at a time.

Setting a View in ClearCase Using Command Lines

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

2 Enter the **password** then press the **Enter** key.

3 At a UNIX prompt type **cleartool setview *ViewName*** where ***ViewName*** is the user's view created in the previous section, then press the **Enter** key.

Setting a View Using the File Browser Screen in ClearCase

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

Cursor moves to the Password field.

2 Type the **password** then press the **Enter** key.

3 Invoke ClearCase by typing **xclearcase &** on the UNIX command line then press the **Enter** key.

The ClearCase Transcript screen is displayed as the View Tag Browser loads.

The ClearCase View Tag Browser screen is displayed listing available views.

4 To set a view, select a known View and press the **Enter** key.

The File Browser window is displayed.

5 Select **File→Execute→Single Command.**

The String Browser window is displayed.

The prompt Enter shell command to run is displayed.

6 Invoke the set view command by typing **setview ViewName** on the UNIX command line and press the **Enter** key.

ViewName is the name of the view to set.

Creating a New Directory

In cases where a new directory needs to be created and placed in ClearCase, the user will activate ClearCase and create a new directory. This type of procedure is necessary only if a new directory is required.

The following is a list of tools, and or assumptions:

1. A VOB has been created at the UNIX directory.
2. A view has been created.

Creating a New Directory in ClearCase Using Command Lines

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

2 Enter the **password** then press the **Enter** key.

3 At a UNIX prompt type **cleartool setview *ViewName***, then press the **Enter** key.

The ***ViewName*** is the user's view.

4 At a UNIX prompt type **cleartool lsvob,** then press the **Enter** key.

This command lists all the VOBs and allows the identification of the SSI&T VOB.

5 At a UNIX prompt type **cd *pathname***, then press the **Enter** key.

The ***pathname*** is the full path name of the parent directory in the VOB in which the new directory is to be added.

6 At a UNIX prompt type **cleartool checkout -nc .** then press the **Enter** key.

This command checks out the current directory. Note the dot for the directory.

The **-nc** is a keyword used when no comments are to be made for this action.

7 At a UNIX prompt type **cleartool mkdir -nc *dirname,*** then press the **Enter** key.

The *dirname* is the name of the new directory being created.

8 At a UNIX prompt type **cleartool checkin -nc *dirname*,** then press the **Enter** key.

This command checks in the new directory named ***dirname*.**

9 At a UNIX prompt type **cleartool checkin -nc .** then press the **Enter** key.

This command checks in the current directory.

Entering a New Directory Using the File Screen Browser into ClearCase

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

Cursor moves to the Password field.

2 Type the **password** then press the **Enter** key.

3 Invoke ClearCase by typing **xclearcase &** on the UNIX command line then press the **Enter** key.

The ClearCase Transcript screen is displayed as the View Tag Browser loads.

The ClearCase View Tag Browser screen is displayed listing available views.

4 Select **File→Execute→Single Command.**

The String Browser window is displayed.

The prompt Enter shell command to run is displayed.

5 Invoke the make directory element by typing **mkdir [filename]** on the UNIX command line and press the **Enter** key.

6 Invoke the make element command by typing **mkelem [directory name]** on the UNIX command line and press the **Enter** key.

7 Type into the directory input box of the **File Browser** the name of the directory in the VOB to be checked out, press the **Enter** key, then follow the menu path **Version→Checkout→Reserved: no comment**.

In order to add new files to ClearCase, the directory in which the files are to be added must be checked out first.

ClearCase forces the checkout onto a maintenance branch to isolate the maintenance activity.

If someone else has already checked out the directory, permission to check out the directory is denied.

A separate shell window is displayed.

8 Cancel the checkout of the element if it is decided that no changes are to be made by typing into the directory input box of the **File Browser** the name of the directory to be checked in, press the **Enter** key, then follow the menu path **Version→Uncheckout→Unreserved: no comment**,

9 On the **File Browser** screen, follow the menu path **File→Exit**.

The ClearCase Graphical User Interface session is closed.

Importing files into ClearCase

Once the user has created a directory to place the science software files, ClearCase can be used to place a single file or multiple files in a UNIX directory structure under CM.

The following is a list of tools, and or assumptions:

A VOB and subdirectory are created to hold these files.

No object files or executables exist in the source code directory.

The PGE was received with a directory structure that contains various types of files.

These files will be entered into ClearCase and will maintain the same directory structure as the delivery structure.

The first procedure to review will be importing a single file. The following procedure review covers importing multiple files in a UNIX directory structure.

Importing a Single File into ClearCase

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

Cursor moves to the Password field.

2 Type the **password** then press the **Enter** key.

3 At a UNIX prompt, type **cleartool setview *ViewName***, press **Enter**

The ***ViewName*** is the name of the ClearCase View.

4 At the UNIX prompt, type **cd *pathname***, then press the **Enter** key.

The ***pathname*** is the full path name of the subdirectory in the VOB into which the file is to be checked in.

If the desired directory cannot be seen, it could mean that the view has not been set or the properties of the view do not allow the directory to be seen; check with the CM Administrator.

5 At a UNIX prompt, type **cp *pathname/filename* .**, press **Enter** (note the space and then "**dot**" at the end of the command)**.**

The ***pathname*** is the full path name to the directory where the file to be checked in exists and ***filename*** is the file name of the file to be checked in.

This command copies a file over into the VOB area in preparation for checking it in.

6 At the UNIX prompt, type **cleartool checkout -nc .**, press **Enter** (note the space and then **"dot**" at the end of the command).

This command checks out the current directory (represented by the **"dot"**) from ClearCase.

Adding a new file (or element) to a directory represents a modification of the directory. Hence, the directory must be checked out before a file can be checked in.

7 At a UNIX prompt, type **cleartool mkelem -nc *filename***, then press the **Enter** key.

The ***filename*** is the name of the file that was copied over in step 5 and is the file that will be checked into ClearCase.

This command creates a ClearCase element from the file in preparation for checking it in.

The **-nc flag** means **"no comment";** it suppresses ClearCase from prompting for a comment to be associated with the make element step.

8 At the UNIX prompt, type **cleartool checkin -nc *filename***, then press the **Enter** key.

The ***filename*** is the name of the file to be checked into ClearCase.

This command performs the check in of the file.

The **-nc flag** means **"no comment**"; it suppresses ClearCase from prompting for a comment to be associated with the checkin step.

9 At the UNIX prompt, type **cleartool checkin -nc .**, press **Enter** (note the space and then "**dot**" at the end of the command).

This command checks in the current directory (represented by the **"dot"**) into ClearCase.

The adding of an element (**here, a file**) represents a modification to the directory and hence, the new version of the directory must be checked back in.

The **-nc flag** means **"no comment"**; it suppresses ClearCase from prompting for a comment to be associated with the checkin step.

Importing Multiple Files into ClearCase

The importing of multiple files into ClearCase will not be performed during the SSI&T Training Lesson. The DAP for the synthetic PGE contains only one source code module and a minimal number of other files. A real PGE will generally contain many source files, header files, and multiple other types of files stored in a standard type of directory structure that is retained when the PGE is packed into the tar file. The script provided by ClearCase is used for the purpose of making another load script to enter all of the DAP files along with the directory structure at one time. The final step of running the load script can only be performed by the DAAC Administrator.

The following procedure explains how to place the entire contents of a UNIX directory structure under ClearCase. A UNIX directory structure refers to all the files and subdirectories under some top-level directory.

This procedure is geared toward science software deliveries. In such cases, science software is delivered is in the form of a UNIX ***tar*** files. A ***tar*** file has been unpacked (**un*tar*-red)** and the contents are to be placed under ClearCase configuration management.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

The following is a list of tools, and or assumptions:

A VOB and subdirectory are created to hold these files.

A ClearCase view is not required to perform this procedure.

Importing Multiple Files Into ClearCase

1 At a UNIX prompt, type **cd *ParentPathname***, then press the **Enter** key.

The ***ParentPathname*** is the path name of the directory that ***contains*** the directory structure to be brought into ClearCase. This is ***not*** the VOB.

2 At the UNIX prompt, type **clearcvt_unix -r *DirName***, then press the **Enter** key.

The ***DirName*** is the name of the directory in which it and everything below it is to be brought into ClearCase.

A conversion script will be then be created. The **-r** causes all subdirectories to be recursively included in the script created.

3 Contact the VOB Administrator and request that the utility script cvt_script be run on the script created in step 2.

The VOB Administrator is the only one who can run the cvt_script because it modifies the VOB.

4 At this time the user logs out from this workstation. The VOB Administrator completes the procedure.

The remaining steps are accomplished by the VOB Administrator.

5 The VOB Administrator logs into the AIT Sun workstation by typing **username** then press the **Enter** key.

Cursor moves to the Password field.

6 Type the **password** then press the **Enter** key.

7 Invoke ClearCase by typing **xclearcase &** on the UNIX command line then press the **Enter** key.

The ClearCase Transcript screen is displayed as the View Tag Browser loads.

The ClearCase View Tag Browser screen is displayed listing available views.

8 To create a view for checking in the software change package, select a known View and press the **Enter** key. If you are using an existing view, select the desired existing view and proceed to step 14.

The File Browser window is displayed.

9 Select **File→Execute→Single Command.**

The String Browser window is displayed.

The prompt Enter shell command to run is displayed.

10 I nvoke the make view command by typing **mkview [filename]** on the UNIX command line and press the **Enter** key.

The tempdisp window appears.

The View [filename] Created Successfully and the Cache Updated for View [filename] prompts are displayed.

11 Close the **tempdisp** window by clicking on the window and press the **Enter** key.

The tempdisp window closes.

12 S elect the VOB where the software change package is to be imported then press the **Enter** key.

13 T o create a subdirectory for the software change package in that VOB, which is a modification to the parent directory (for the VOB) the parent directory must be checked out by following the menu path **Version→Checkout→Reserved: no comment**.

In order to add new files to ClearCase, the directory in which the files are to be added must be checked out first.

ClearCase forces the checkout onto a maintenance branch to isolate the maintenance activity.

If someone else has already checked out the directory, permission to check out the directory is denied.

A separate shell window is displayed.

14 Star t a shell process in a separate window by clicking on the shell icon button of the **File Browser** toolbar.

A separate shell window is displayed.

15 To run the script, type **cvt_script** then press the **Enter** key.

The VOB Administrator is the only person who can run the **cvt_script** because it modifies the VOB.

16 To check in the new directory, type into the directory input box of the **File Browser** screen: **path** [where **path** is the full path identification for the new directory (**directoryname**)], then press the **Enter** key. Then select **Versions→Checkin** from the menu.

17 T o check in the parent directory (for the VOB), type into the directory input box of the **File Browser** screen: **VOBpath** (where **VOBpath** is the full path identification for the parent directory), then press the **Enter** key. Then select **Versions→Checkin** from the menu.

18 On the **File Browser** screen, follow menu path **File→Exit**.

The ClearCase Graphical User Interface session is closed.

Checking Out a File From ClearCase

If a configured file requires modification, then the file needs to be checked out of the configured directory and placed in a user directory. This will allow the file(s) to be modified.

The following is a list of tools, and or assumptions:

The file or directory must be an element created in ClearCase.

The view should be configured to ensure the correct version of the file or directory is seen.

Checking Out an Element/File from the Command Line

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

2 Enter the **password** then press the **Enter** key.

3 At a UNIX prompt type **cleartool setview *ViewName***, then press the **Enter** key.

The ***ViewName*** is the name if the user's view.

4 At a UNIX prompt type **cleartool checkout -nc *element*** then press the **Enter** key.

The ***element*** is the name of the file or directory that is to be checked out.

The **-nc flag** means **"no comment"** which will suppress the ClearCase prompting for a comment to be associated with the check out step.

Checking Out an Element/File from the File Screen Browser

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

Cursor moves to the Password field.

2 Type the **password** then press the **Enter** key.

3 Invoke ClearCase GUI by typing **xclearcase &** on the UNIX command line then press the **Enter** key.

The ClearCase Transcript screen is displayed as the View Tag Browser loads.

The ClearCase View Tag Browser screen is displayed listing available views.

4 To check out the directory where the controlled files were place, type into the directory input box of the **File Browser** screen: **path** [where **path** is the full path identification for the directory (**directoryname**)], then press the **Enter** key. Then select **Versions→Checkout** from the menu.

5 Select **File→Execute→Single Command.**

The String Browser window is displayed.

The prompt Enter shell command to run is displayed.

6 To determine editing privileges, type **ls -l,** then press the **Enter** key.

A prompt displaying read/write/execute privileges will be displayed. There will be three groupings:

User Group Others

r=read, w=write, x=execute

7 If you have editing/execute privileges, you can revise the contents of the file with any text editor.

8 To checkin a controlled file, select **Versions→Checkin** from the menu.

The file/directory will be checked in to ClearCase and the version will be updated.

Checking a Modified Element into ClearCase

This procedure explains how to check in a modified element to ClearCase. An element refers to a directory or file in ClearCase, that is, under configuration management. Modifications made to a file or directory cannot be saved in ClearCase unless the file or directory had been checked out first.

The following is a list of tools, and or assumptions:

A VOB exits and is mounted at a known UNIX directory.

A ClearCase view exists for the SSI&T operator.

The element or file has been checked out and modified.

The modified file is now in the user's directory on the VOB from which it was checked out.

Checking a Modified Element/File into ClearCase

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

Cursor moves to the Password field.

2 Type the **password** then press the **Enter** key.

3 At a UNIX prompt, type **cleartool setview *ViewName***, then press the **Enter** key.

The ***ViewName*** is the name of the user's view.

4 At the UNIX prompt, type **cleartool checkin -nc *filename***, then press the **Enter** key.

The ***filename*** is the name of the file (full path name allowed) that is to be checked out (and later modified).

The **-nc flag** means **"no comment"**; it suppresses ClearCase from prompting for a comment to be associated with the check out step.

This command checks in the current directory.

5 This step is optional; it is performed when ClearCase does not accept a checkin because the element was not modified. In this case, the check out must be canceled. At a UNIX prompt, type **cleartool uncheckout -nc *filename***, then press the **Enter** key.

The ***filename*** is the name of the file or directory **(full path name allowed)** checked out.

This command cancels the check out of an element/file.

Science Software Integration and Test (SSI&T) Preparation, Processes and Setup

Key Operator Roles

Science Coordinator: Provide support to Instrument Teams for the integration and testing of science software in the ECS system at the DAAC. Perform standard checking on all delivered software including source code, scripts, process control files and related documentation.

Science Data Specialist: Serves as a point-of-contact for planning, integrating, testing, and operating science software.

CM Administrator: Record, report, manage and distribute new and updated science software.

Science Software I & T Support Engineer: Provide support to Instrument Teams for the development, integration, test and problem resolution of science software.

Production Planner: Populate, maintain and schedule the production planning database for science software.

COTS Software Tools

ClearCase: This tool is used as the ECS software configuration management tool. ClearCase provides a mountable file system that is used to store version-controlled data, such as source files, binary files, object libraries and spreadsheets.

Distributed Defect Tracking System (DDTS): This tool is used to electronically process configuration change requests **(CCRs).** DDTS will prompt the user for relevant information, identify the request and will mail these requests to pre-designated personnel.

MODIS Science Data Processing Software Version 2.0 System Description Manual

This manual should be referred to for more detailed information on how to perform the SSI&T operational procedures as they apply to MODIS PGE's. It covers the specific attributes for each individual PGE and setup criteria.

Operations Tools Manual

The SSI&T operational procedures are given in many of the section that follow. They are organized by activity. The order in which the procedures appear loosely follows the order in which they will usually be performed.

These procedures present the use of GUIs supplied in Release 6A. Some procedures may have a command line equivalent; these are documented in the corresponding GUI help screens but are

not presented here in the interest of simplicity. Version 2.0 Operations Tools Manual 609-CD-003-004 should be referred to for more detailed information on how to use GUI's and command line equivalent usage.

General Process

The SSI&T process consist of two activities:

Pre-SSI&T activity - During this activity the Delivered Algorithm Package (DAP) is inspected, and tested in a non-production environment.

Formal SSI&T activity - During this activity, the Product Generation Executives (PGEs) are integrated with the DAAC version of the SDP Toolkit and executed on the ECS PDPS platform.

- **Key Terms:**

Product Generation Executives (PGEs) - The smallest scheduled unit of science software.

Delivered Algorithm Package (DAP) - An ensemble of PGE source code, makefile, documentation, and other related files delivered in a package from the SCF to the DAAC for SSI&T.

Process Control File (PCF) - Relate logical identifiers to physical files and other parameters required by the PGE.

Strings - The processing hardware on which the science software runs.

Archive - A File Storage Type indicating that granules that will be inserted toData Server are intended for long term storage and acquisition for distribution.

Collection - A related group of data granules.

Granule - The smallest data element, which is identified in the inventory, tables.

Product - A set of output values generated by a single execution of a PGE for archival by ECS. A PGE may generate one or more products whose attributes are defined by the data provider.

Reliability - Software reliability means that the software runs to normal completion repeatedly over the normal range of data inputs and running conditions.

Safety - Software safety means that the software executes without interfering with other software or operations.

The science software in the DAPs will be integrated onto the PDPS and be used to produce the output data as determined by the algorithms. The refined and updated DAPs and data produced by the science software will eventually be provided to the subscribing user. Before the PGE is integrated into a production environment, extensive testing on the software must be performed.

The following list provides a suggested, logical "road map" for getting science software tested and integrated into the ECS. This list is not intended to cover every situation and variations may be required.

GENERAL

Science Software Integration and Test (SSI&T) is the process by which the science software is tested for production readiness in the DAACs in order to assure its (1) reliability and (2) safety. Prior to the delivery of the ECS software to the DAACs, SSI&T Checkout is conducted on early versions of the Products Generation Executives (PGEs) using separate system modes in the ECS Mini-DAAC, VATC (Verification and Acceptance Test Configuration), or the DAAC environments.

SSI&T activities can be broadly separated into two categories: pre-SSI&T and formal SSI&T. Pre-SSI&T activities are those which do not involve the ECS Planning and Data Processing (PDPS) or the Science Data Server (SDSRV), but the formal SSI&T activities do involve the full ECS including the PDPS and the SDSRV.

Most steps in the SSI&T process are inter-related and some steps may assume that another step has been completed. The ordering of the steps is very important but it cannot, however be interpreted as a detailed, step-by-step guide to SSI&T activities.

Science Software Integration and Test consists of the following activities most of which are fully detailed in Science Software Integration & Test Operational Procedures for the ECS Project (162-TD-001).

Science Office Project Instructions

The following subjects listed in Figure 4, are procedures from Science Office Project Instructions as they apply to the building of Science Data Products and SSI&T. A review of these documents is highly recommended at this time . They can be accessed from the Web using the following URL:

http://dmserver.gsfc.nasa.gov/proj_instr/sopi_index.html

Number	Subject	Issue Date
SO-1-002	Earth Science Data Type Generation Procedures PDF	5/11/98
SO-1-003	Science Software Integration and Test (SSI&T) PDF	9/14/98
SO-1-004	Science Office Science Support Internal Processes PDF,	
Science Office Science Support Internal Processes (161-IT-003-001)PDF		6/30/98
SO-1-005 Product Specific Attribute (PSA) Analysis PDF		7/23/98
SO-1-006	PGE Testing PDF	7/07/98
SO-1-007	Earth Science Data Types Testing and Integration PDF	7/24/98

Number	Subject	Issue Date
SO-1-008	QA Metadata Update Tool (QAMUT) PDF	7/23/98
SO-1-009	Metadata Works PDF	7/23/98
SO-1-010	ECS Science Metadata Valids Update Procedures PDF	7/24/98
SO-1-011	Metadata-Process Established with MODIS PDF	8/20/98
SO-1-013	Test Data Configuration Management Project Instruction PDF	3/8/00

Figure 4. Science Office ECS Project Instructions

Pre-SSI&T Activities

1 As the DAP is delivered to a DAAC by the Instrument Team for SSI&T, the PGE listing documentation is reviewed.

2 The DAP is acquired and unpack and the documentation (i.e., packing list, readme, etc.) checked. The DAP contents are further checked by the Science Data Specialist to verify that the contents match the packing list, agreed-upon directory structures are employed, location of files are correct, and all intended files and directories are present.

3 The Science Data Specialist requests that the CM Administrator place the DAP under Configuration Management control using ClearCase.

4 The SSI&T team checks the science software for standards compliance using the Process Control File Checker to check process control files (PCF), and the Prohibited Function Checker to check source files. Extract and check prologs.

5 The SSI&T team builds the science software into PGEs using the SCF version of the SDP Toolkit. Compile all source code. Link object code with appropriate libraries. . If the SMF files compile successfully, then proceed to Step 11 below; otherwise. the problem needs to be fixed and a successful compile must occur before proceeding further. This may require one or more of the following:

Note any error messages and review the included documentation to ensure a proper compile;

Check environmental variables;

Review the setup files for proper directories and variables;

Make corrections and recompile.

If the executable builds successfully, proceed to Step 12. If the build fails, it may be necessary to do one or more of the following before proceeding:

Check environmental variables;

Make corrections and repeat the build.

6 Run the PGE from the Command Line.

If it the execution is successful, then the output files (products) are checked using the SSIT Manager file comparison tools; otherwise, one or more of the following needs to be done before proceeding:

Check error logs;

Check environmental variables;

Review and source the setup files;

If necessary files are missing, then review the PCF file to ensure the existence of correct reference directories.

7 The SSI&T team runs and profiles the PGEs from the UNIX command line on the SGI, saving the profiling results. They will be used later when entering operational metadata into the PDPS.

8 The SSI&T team collects performance statistics for the PGEs.

9 The SSI&T team examines the output log files from the PGE runs for any anomalous message. The SSI&T team compares the output product data with the delivered test data using the file comparison tools. If the products do not match the delivered test outputs (expected outcome), the outputs should be analyzed and the PGE must be re-run. If the products match the delivered test outputs then

10 Steps 10 through 13 are repeated once using the DAAC Toolkit. If the products generated with the DAAC Toolkit match the delivered test output, formal SSI&T may begin.

11 SSI&T team reports any science software problems using the DDTS NCR process.

12 The SSI&T team reports any ECS problems using the DDTS NCR process.

13 The SSI&T team collects and logs all lessons learned.

Formal SSI&T Activity

1 For each ESDT used by the PGE, construct an ESDT ODL file for updating the PDPS or verify that they already exist. ESDT ODL files are also needed for all input and output data granules.

2 Construct a PGE ODL file for updating the PDPS database. This involves using the delivery PCF to construct an initial PGE ODL template file, which must then be hand edited to add required metadata. A mapping between logical IDs in the PCF and ESDT ShortNames must be known before this step is done.

3 Install ESDTs on the Science Data Server if verification indicates that they do not already exist. Installation links the PGE to all input and output ESDTs that allows the PGE to run within the PDPS. The Advertising Server must also receive notification of the update. If this fails then the ESDT's must be re-installed again after removing original ESDT's from the SDSRV. Note: While installing ESDT's the SDSRV intermittently coredumps. To clean-up you must remove the ESDT from ADSRV, SBSRV and DDICT and then try again.

4 The Science Metadata.met is updated (PGE & ESDT Object Description Language or ODLs are created). This supplies metadata to the PDPS database

- If the Science Metadata update is successful, then the Operational Metadata is updated; otherwise, the ESDT ODL files may have to be checked for correctness before updating the Operational Metadata.

5 Register the PGEs with associated data in the PDPS database. This step uses the PGE ODL from step 2 above.

6 For each input dynamic data granule needed by the PGE, construct a Target MCF and insert both granule and .met files into the Science Data Server.

7 For each input static granule needed by the PGE, construct a Target MCF.met file and insert both to the Science Data Server.

8 Assemble the SSEP (as a tar file) and Insert it to the Science Data Server.

9 Initiate a Production Request (PR) that will result in one or more DPRs.

10 Use the Planning Workbench to plan the PR and hence, run the PGE.

11 Monitor the PGE run using AutoSys. The PGE's progress is monitored using the AutoSys COTS. The distinct steps that are visible on the AutoSys GUI and whose success is evident are Resource Allocation (.Al), Staging (.St), Pre-Processing (.Pr), Execution of the PGE (.EX), Post-processing (.Ps), De-staging (.Ds), and De-Allocation of resources (.Da).

12 If any of the steps in the execution is not successful, then each failure must be identified and corrected before proceeding to the next step.

13 Examine the output Production History File from the PGE runs for any anomalous messages. Compare the output product data with the delivered test data using the file comparison tools. . If any of the steps in the execution is not successful, then each failure must be identified and corrected before proceeding to the next step.

14 If the output files match the test output files and they are in Hierarchical Data Format (HDF),

they are visualized using the EOSView tool, or the Interactive Display Language (IDL) tool. If the files are not HDF, then IDL is used.

15 Using the Planning subsystem, initiate more complex Production Requests if chaining is required.

16 Using electronic or hard media transfer methods, distribute the data products to the Instrument Teams for their review.

RECORDS

A weekly SSI&T status report is provided to NASA. This report contains the Performance Measurement Data.

PERFORMANCE MEASUREMENTS

SSI&T PGEs planned vs. actually delivered, pre-tested, and integrated is the metric used to monitor the effectiveness of the process described in the Procedure. Additionally, the Duration of Effort Required to Integrate in Work Days is used.

Preparation and Setup to use the SSI&T Manager Tool

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

2 Enter **setenv DISPLAY <local workstation IP address>:0.0** where the local workstation IP address represents the IP address where you are located.

3 You may need to setup the terminal so that the remote host is displayed on your screen. (Sun machine) This is done by clicking on the Application Manager icon (the file drawer located at the bottom of the screen), followed by the Desktop Tools icon, followed by the Terminal Console icon.

4 Perform a remote login by typing rlogin [p0ais01] then press the **Enter** key.

The Enter Password prompt is displayed.

5 Enter the **password** then press the **Enter** key.

6 Enter the directory where the setup script is located by typing **cd [**directory name**]** then press the **Enter** key.

7 Source the setup script by typing **source [**script name**]** then press the **Enter** key.

The setup script contains directory paths, sets of alias commands, and tools for SSI&T.

For example, source the SSI&T script: Type **source /usr/ecs/{MODE}/CUSTOM/utilities /.buildrc** <ENTER>

Note: This step only needs to be done once per login

8 To ensure access to the multi server environment when needed, the following generic login command have been established and should be used routinely:

9 From a terminal: **xterm -n (host) &**

10 From the xterm invoked: **telnet (host)**

11 login **ID, PW**:

setenv DISPLAY clientname:0.0 and then press the **Enter** key.

12 **cd /usr/ecs/{MODE}/CUSTOM/utilities /.buildrc**

13 L isted are some of the Gui tools, typical servers and their Host that need to be considered for activation when conducting SSI&T:

ECS Assistant, ADSRV/DM/IOS, p0ins02

ECS Assistant, SDSRV/DSS, p0acs03

ECS Assistant, DPS, p0sps06

ECS Assistant, SBSRV/CSS/IOS, p0ins01

SSIT Manager tools, AITTL/DPS p0ais01

Production Request, PLS, p0pls01

Planning Workbench, PLS, p0pls01 Note: NETSCAPE should be closed to allow for a full screen GUI to be activated.

Monitor PGE , DPS, p0sps06 using AUTOSYS

15 A s econd xterm should be activated with the same login procedures so as to monitor the (log files) when entering SSI&T files from GUI's.

16 Servers can be brought down in any order. To bring them backup requires that they be brought up in **a sequential order to ensure connectivity**, the order is listed as follows:

STMGT, MSS, DDIST, IOS, SDSRV, PDPS

17 The above servers have unique hosts assigned . Each host needs to be logged into the **generic login: ID, pw: ID press Enter Key** before activating ECS Assistant to carryout the downing and bringing up of servers assigned to their respective hosts.

Science Software Integration & Test Flow:

Software is developed at the SCFs using the SDP Toolkit. It is shipped to the DAAC for testing in a package that includes all source code, make files, a version of the Process Control File (for the Toolkit), a Metadata control file (for the Toolkit), some files describing the PGE Profile (this data will be turned into the ODL files that are ingested by AIT to form the PGE registration), documentation, test plans, & test files. It may be shipped via ftp, hard media, or some combination there of. Note that some of the data (particularly large test data files) may be shipped separately from the rest of the stuff. Since all the DAPs are retrieved by the SSIT staff, there shouldn't be any problem integrating a partial DAP with the software, test files, etc.… that go along with it. NOTE that it is possible that the SCFs will want to create the ODL files and ship them with the DAP. While this is possible, the PGE Metadata ODL file template is created via an SSIT tool by reading the PCF. Then this template and the PGE Metadata ODL file would have to be merged some how. There may have to be some kind of turnover form or PGE worksheet for the Instrument Teams to specify the information needed in the ODL files.

Software is ingested into the system (by INGEST) and placed on the Data Server (*in TS1 mode or OPS mode?*). The AIT workstation has a subscription on all new DAPs that is created by the Subscription Editor.

A ClearCase work area is created on the SSIT machine(s) for testing and correcting the software. The source code will be maintained in ClearCase to support versioning. The actual compilation and execution of the source code may be done on the SSIT workstation (currently slated to be a Sun), but will eventually have to be done on the target software (one of the science processors—currently slated to be an SGI). To test software on a science processor, the machine must be "signed out" by the Resource Planner.

The DAP (Delivery Archive Package) is acquired from the Data Server. This is done through an SSIT script tool. It is unpacked (via "tar" commands) at the SSIT machine. This is currently a manual process, though we could automate it in the same script that is used to retrieve the DAP.

The files of the DAP are placed in ClearCase for configuration management. Since the DAP is just a big tar file, or series of tar files, it has (included within it) the directory structure of the code. When the code is extracted, it will be placed (via untaring) into the correct directory structure.

The PGE is complied using SDP Toolkit tools, and verified using a combination of Toolkit and SSIT custom functions (Prohibited Function checker, Fortran77 checker). SSIT tools are brought up from the SSIT GUI menu. First the SCF version of the Toolkit is used, then the DAAC version of the Toolkit is used to verify the compilation and tests. The compilation of the PGE will take place on the target hardware (science processors), not the SSIT workstation(s).

Tests are performed as they were at the SCF. Test input is put into local directory on the machine where the test is being performed and then the PGE is run from the command line using the delivered PCF (with minor modifications).

Any problems found can be fixed on the local copy of the source code. Tests will be rerun to verify completeness of the fix. Errors found are entered into DDTS for tracking. Once the command line tests of the PGE have completed successfully, then the executables must be stored at the Data Server (running in SSIT mode) so that they can be retrieved by PDPS.

The PGE executable and any associated binary files (Toolkit files or any binaries called by the main program of the PGE) are stored on the Data Server as tar files. DPS will stage and extract the tar file when the PGE is scheduled to run. The tar file is untared in one directory, with the profile (any environment variables that must be set), top level shell (main program of the PGE) and the Toolkit SMFs (message files) in the main directory. This is used both when testing the PGE in a non-operational PDPS system (with Data Server in "SSIT" mode) and in the production system (in "OPS" mode). There is a script to insert the exe tar file into the Data Server, as well as any static files needed by the PGE.

When standalone tests have completed successfully, information about the PGE is entered into the PDPS Database (a copy resides on the AIT hardware) so that it can be tested in the Production Environment. Input and Output Data Types, Production Rules (new for Release B), etc... are supplied by the ODL files. The PGE ODL file is generated from the PCF (see next bullet) but must be completed by the SSIT personnel. ODL files must also be created for the ESDTs used by the PGE and any production rules. A SSIT tool parses the ODL files and places the data in the PDPS database. The PGE Registration GUI is used to finish the

information about the PGE, allowing the input of Performance Statistics and Resource Requirements (collected via EcDpPrRuseage) and any text description for User Parameters.

The original PCF (that was part of the DAP) is NOT used by DPS when the PGE is run in PDPS. All information from the original PCF is captured in the PDPS database (as part of the PGE profile). DPS uses the entries in the PDPS database to create a PCF each time the PGE is run. The original PCF will be saved in the SSAP. There is an SSIT Tool that takes as input the PCF of the PGE and then generates a template ODL file.

The original PCF must be changed before the ODL files are created. Any entries of Attitude/Ephemeris/DEM data in the PCF are removed if the source code is found to NOT have the corresponding Toolkit calls to access the data. Also, any PCF entries for binary metadata output files (called ".met" files) will be removed and replaced with a Runtime Parameter (at the same logical id) that tells the Toolkit the logical id of the binary output for which the ascii ".met" file is for. This allows the Toolkit to name the ".met" file after the binary output and helps DPS later.

PGE is tested in the production system. The PLS and DPS software will coexist on the SSIT hardware and will be brought up there in "SSIT" mode. SSIT hardware will also include AutoSys for processing. A science processor will have to be signed out to allow the tested PGE to be executed.

DAAC and SCF personnel agree that the PGE performs properly. It is accepted as "product ready" software.

The Science Software Archive Package for the PGE is defined. All source code, documentation, test plans, scripts, test data (maybe only a pointer to this) must be put into the SSAP. The SSAP GUI allows the SSIT operators to add files to the SSAP and then Insert it all at once. The SSAP is actually two different data types at the Data Server. The Algorithm Package is just metadata, information about the SSAP that must be filled in by the SSAP GUI. Then each SSAP component (source code, test data, etc.….) must be inserted into the Data Server separately. To "copy" the files into the SSAP, the user will have to know their location on the SSIT workstation (and in ClearCase) so that they can be selected on the GUI. Once the SSAP is complete, it is inserted to the Data Server (many different inserts by the GUI itself). Note that the GUI also allows the used to modify an existing SSAP (retrieve it and then re-insert it) and delete an existing SSAP.

NOTE that the SSIT personnel may want to save the SSAP to both the "SSIT" and "OPS" Data Servers. To allow this, the SSAP GUI must be able to save SSAPs to the local disk and recreate them so they can be "copied" from one mode of the Data Server to the other.

The PGE executables are stored separately from the SSAP, as are any static files needed during PGE execution. There is a tar file that contains the exe and any Toolkit files that the executable needs. A script tool performs the Insert of this to the Data Server. The static files are also stored via a script tool.

The PDPS database (on the AIT hardware) that was populated during testing is then copied into the "operational" PDPS database on other hardware. This is done by re-running the tool that populates the PDPS database from the ODL files in the "OPS" mode. This means that the ODL files need to be saved (possibly *checked into ClearCase?*) so that the production database can be populated after SSIT.

Descriptions of the various Ids that SSIT creates

PgeId: This is the identifier of the PGE, used as a key into the PDPS database. SSIT takes the PGE name, and concatenates the PGE Version and Profile Ids. The maximum length is 20characters, with the maximum name 10 characters, maximum version 10 characters and the profile Id 3 characters. '#' is used to pad the end of the name or version if it is not equal to the maximum length.

SswId: This is the identifier of the PGE executable, also used as a key (for different tables) in the PDPS database. SSIT takes the PGE name, and concatenates the SSW Version (version of the source code). The maximum length is 20 characters, with the maximum name 10 characters (as in PgeId), and the maximum SSW version 10 characters. '#' can be used to pad the end of the name if it is not equal to the maximum length.

DataTypeId: This is the identifier of Data Types, and is used as a key in the PDPS database. SSIT forms it in one of two ways: for normal Data Types, it takes the Data Type Name (ESDT Short Name) and concatenates it with the Data Type Version (ESDT Version). The maximum length is 20 characters, but since the maximum name is 10 characters and the maximum version 10 characters, a '#' can be used to pad the end of the name.

The Science Server Archive Package (SSAP):

This is a grouping of science software, documentation, and other related files that is stored at the DAAC. These are accessed and updated through an SSIT SSAP GUI. It is supposed to be a record of the software, complete with source code, documentation, what and how it was tested, etc….. This is both for recording purposes and so that the tests can/could be repeated later. Note that the executables and any static files needed by the PGE are stored separately from the SSAP.

The SSAP is not what is received from the SCF. This is currently being called the DAP (Delivery Algorithm Package) and it is similar to the SSAP in that it contains test data, source code, etc…. Much of what is in the DAP will make it into the SSAP (although it may be modified, i.e. code may have bug fixes).

The SSAP will be made up of 2 different Data Types at the Data Server. The Algorithm Package is metadata about the SSAP, such as the name of the PGE, name of the instrument, date accepted, etc…. Each part of the SSAP (source code, documentation, test data) will be stored as an SSAP component, with its own metadata in addition to the files. SSAP components such as source code will be tared to retain the directory structure

(so they much be untared when retrieved). The executables and static files are stored separately from the SSAP, and thus have their own Data Types (ESDTs).

What follows is a list of items in the SSAP taken from the DID205). See also the Core Metadata model under DAP for a graphical representation of the SSAP.

The following make up an SSAP:

Documentation

–Delivery Memo (*will this be deleted upon delivery?*)
–Summary Information for each PGE.
–System Description Document (SDD).
–Operations Manual
–Processing Files Description Document
–Test Plans (these include the test cases)
–Scientific documents
–Interface Definition Document
–Detailed Performance Testing Results
–Detailed design/implementation documents
–COTS User or Programmer Guides

Software & Control files:

–Science software source code (including make files & scripts)
–Testing software source code (including make files & scripts)
–Test Data Input (this may only be the UR for this)
–Expected Test Output
–Coefficient Files
–Process Control File
–Metadata Configuration File
–ODL files. These define the PGE, and its related Data Types to the PDPS database. They currently don't have official names.

Other files:

–There is a log file that is created by the SSAP GUI to describe any changes made to the SSAP.

The Current list of File List types in the SSAP code:

(This is my best guess as to what the various File List types there will be inside of an SSAP. Right now everything is interpreted as a tar file.)

Change Log

This should be where the access log (created by EcDpAtAccess) goes. Note sure what else would go here.

Context Diagram

This would be the context diagram(s) for the PGE.

Delivery Contents

A copy of the delivery memo?

Manual

Any manuals about the PGE, test code, etc….

Performance Test Results

The results of any performance tests.

Processing File Description

Programmers Guide

A pointer or a copy to the Programmers guide used to write the software.

Test Plan

The plan for testing the software.

Software Standard

The software standard used to write the software.

System Description

Description of the system used to run the software (I think).

Compilation Information

Information on how to compile the software.

Description

A description of the PGE/software. Note that this might not be an actual part of the SSAP. It might only exist in metadata.

PGE Error Log

Log of PGE errors. I'm not sure if this is errors recorded while running the PGE, or how to log errors when the PGE is being run.

PGE Executables

The executable(s) that make up the PGE. These are what are executed to run the PGE. Note that the executables that are used by processing are currently retrieved from somewhere else.

Dependencies

Test Site Configuration

The configuration needed to test the software.

Version

PGE version. Note that this might not be an actual part of the SSAP. It might only exist in metadata.

External Data

Data needed when running the PGE. Note that I think this is only for testing.

Engineering Data

More data needed when running the PGE. Note that I think this is only for testing.

Science Data

Scientific data (temporary products) needed to test the PGE.

Metadata Control File

Needed by the Toolkit to allow for updates to the metadata of a product by the PGE.

Ancillary Data

More data needed to run the PGE. This data may NOT be for test only.

Control parameters and Resource files

More information on running the PGE. This might include runtime parameter values.

Results Report

Report of tests run.

Results Product files

Files created during testing. These also could be comparison files for use after a test is run (to compare current output with expected output).

Results Temporary files

Temporary files created during testing.

Test Scripts

Scripts needed when testing the software.

Test Source

Source code for test drivers, stubs, etc….

Compilation Scripts

Scripts needed to compile the software.

Software Source

The source code for the software.

Software Scripts

Scripts needed to develop or run the software.

Data Server had to write a special interface for SSAPs. AIT will use standard Data Server commands to access the SSAPs, INSERT (add), ACQUIRE (Retrieve), QUERY/INSPECT (give me a list of them). The Algorithm Package will be inserted as just metadata, while the rest of the SSAP will be insert with both metadata and component file(s). There is currently a requirement for creation of HTML pages to allow users to access and retrieve parts of an SSAP. These HTML page requirements may go away, since all of that can be performed by the SSAP GUI. The SSAP Editor is covered in more detail in the 313-CD-510-002 Release 5B ECS Internal Interface Control Document for the ECS Project, section 3.8.6. Additional information on the SSAP Editor is also available in 609-CD-003-004 Operations Tools Manual for the ECS Project , section 4.5.1.2.9.6.

ODL File(s)/PGE Metadata:

These are parameter = value files that describe the PGE, its inputs and outputs, along with other information needed in the PDPS database. The current plan is for these files to be kept in ClearCase with the PGE, and thus to be configuration managed along with the software. They could also be stored as part of the SSAP.
There are currently 5 types of ODL files:

- PGE Metadata (information about the PGE)
- ESDT Metadata (information about each input/output of the PGE),
- ORBIT Metadata (information about the orbit of the spacecraft supplying the data for the PGE) ,
- TILE Metadata (information about the geographic tiles that the PGE will produce) and
- PATHMAP_ODL (information about mapping between Absolute Path Number and a Sequential numbering ranging from 1-233).

There are ".template" files for each type of Metadata/ODL file that describe each field and give examples of how its used. A PGE Metadata ODL file must exist for each PGE Profile in the system, and an ESDT Metadata ODL file must exist for each input/output of those PGEs. ESDT Metadata ODL files can be shared by PGEs (that have the same input/output). The ORBIT Metadata ODL File is only required for PGEs that are scheduled by "Orbit" rather than time, and it will only exist for each type of Spacecraft for which there is data to be processed. The TILE Metadata ODL File will exist for each definition of Tiles, where a PGE wants to be scheduled to produce data for a certain geographical location. The PATHMAP_ODL is required if the PGE _ODL specifies a PATHMAP . The PATHMAP definition ODL files must reside in directory $DPAT_RULE_SCIENCE_MD. Each file must be named PATHMAP_<Pathmap_Name >.odl.

The PGE Metadata ODL file can be created from a PGEs PCF by running the CreateODLTemplate tool or it can be created from scratch by copying the PGE_ODL.template file. In either case the file must be edited to add ESDT information, correct numbers of input and output files, type of scheduling for the PGE, and many other things. The ESDT Metadata ODL File must be created from either an existing ODL File or the ESDT_ODL.template file. The Tile and Orbit Metadata ODL Files are also created from the template files (TILE_ODL.template, ORBIT_ODL.template) are copied from an existing ODL file and edited. Once all ODL files for a PGE are created (PGE Metadata ODL File, any ESDT Metadata ODL Files needed to describe inputs and outputs, an ORBIT Metadata ODL File to describe spacecraft orbits and a TILE Metadata ODL File to describe geographic outputs), then the information about the PGE can be stored in the PDPS database so it can be scheduled and executed by Planning and Processing.

This page intentionally left blank.

Science Software Integration and Test (SSIT) Manager

SSIT Manager Overview

The principal tool used during SSI&T is the SSIT Manager. The SSIT Manager is the top-level graphical user interface (GUI) environment presented to SSI&T personnel. Its purpose is to bring together the tools needed for SSI&T into a single, graphical environment. Please refer to Release 5B, Operational Tools Manual 609-CD-510-002, section 4.5, for the latest update on expanded uses of the SSIT Manager.

The SSIT Manager is the starting point for use by the SSI&T specialist to check in and verify the science software delivered by the instrument teams at the SCFs. The SSIT Manager application provides access to all COTS tools and custom applications that are part of the SSI&T environment. The SSIT Manager GUI is capable of kicking off instrument-specific compilation and execution scripts, configuration management scripts, custom code checking, file display and comparison tools, and COTS tools such as analysis environment programs. The SSIT Manager GUI contains a checklist of SSI&T steps in delivery and testing of science software, and a display of a log file recording SSI&T events. The checklist and log are the only inherent functionality to SSIT Manager; all other programs run from the Manager are also accessible from the Unix command line.

The terms Process Control File (PCF) and Object Description Language (ODL) are used in the following sections. The PCF is a file that tells an executable where to find its various inputs and outputs, as well as the values for any specific runtime parameters. Different various of these files are used both by the SSIT Manager and by PGEs. ODL is a parameter = value format for input files. It is used to define the PGEs to the Planning and Data Processing Database.

Table 2.1-1, Presents a summary of the capabilities provided via the SSIT Manager GUI.

Table 2.1-1. Common ECS Operator Functions Performed through the SSIT Manager GUI (1 of 3)

Operating Function	Command/Script or GUI	Description	When and Why to Use
Prepare SSI&T checklist.	SSIT Manager GUI	• SSIT Manager GUI requires a checklist which it can access • the checklist contains all the operational procedures for science software integration and test	• during normal SSIT operations, used to keep track of activities pending and completed • checklist of operational procedures must first be prepared using a text editor. • SSIT Manager must then be linked to the checklist before invoked
Change SSI&T Checklist.	SSIT Manager GUI	• SSIT Manager GUI requires to change the checklist • the checklist contains all the operational procedures for science software integration and test	• Update tracking of activities pending and completed • checklist of operational procedures must first be prepared using a text editor. • SSIT Manager must then be linked to the checklist before invoked
open xterm session.	• open xterm window via Tools pull-down menu • also can be opened via Unix command *xterm*	Standard Unix command line window.	As needed for ad hoc use.
Code analysis.	select SPARCworks via Tools: Code Analysis pull-down menu	Used for ad hoc analysis of science software.	used to debug problems (e.g., memory leaks)

Table 2.1-1. Common ECS Operator Functions Performed through the SSIT Manager GUI (2 of 3)

Operating Function	Command/Script or GUI	Description	When and Why to Use
check for standards compliance.	select the following via Tools: Standards Checkers pull-down menu: • FORCHECK • Prohibited Function Checker • Process Control File (PCF) Checker • Prolog extractor	• check FORTRAN 77 science software • check if certain functions are used in the science software which conflict with the production environment • check the syntax of the data in the Process Control File • Extract prologs from science software	• to ensure that science code conforms to ECS standards • to ensure that the delivered PCF is of the proper syntax • To extract prologs from science software
Product Examination.	select the following via Tools:Product Examination pull-down menu: • IDL • EOSView	Opened via Tools pull-down menu.	• Ad-hoc graphical analysis • For viewing an arbitrary file (e.g., standard product) in HDF format
File Comparison.	select the following via Tools: Product Examination File Comparison • ASCII files • Binary files • HDF files (GUI) • HDF files (hdiff)	Compares the outputs of the science software between the DAAC and SCF.	Ensures that the output that was generated at the SCF when running the science software is the same output that is generated at the DAAC.
Edit text.	select the following via Tools: Text Editors pull-down menu: • Emacs • Xedit	Text editors.	Edit arbitrary text file.

Table 2.1-1. Common ECS Operator Functions Performed through the SSIT Manager GUI (3 of 3)

Operating Function	Command/Script or GUI	Description	When and Why to Use
PDPS Database	select the following via Tools:PDPS Database pull-down menu: • PCF ODL Template • Check ODL Files • SSIT Science Metadata Update • SSIT Operational Metadata Update GUI	• creates an ODL file template from the science software PCF • Check the ODL file updates PGE and ESDT SCIENCE metadata in the PDPS /SSIT database • updates PGE OPERATIONAL metadata via GUI in the PDPS /SSIT database	To initialize and update the Planning/Production (PDPS) databases: • SSIT version • Production version
Data Server	select the following via Tools: Data Server pull-down menu: 1. Acquire DAP 2. Insert Static 3. Insert Test Dynamic 4. Insert EXE TAR 5. Edit SSAP 6. Get MCF	1. Acquires DAP. 2. Inserts static input file. 3. Inserts test dynamic input file. 4. Inserts tar file with files needed for Processing. 5. Edits Science Software Archive Package (SSAP) components. 6. Acquires Metadata Configuration Files from the Data Server.	1. After DAP notification received by email. 2. After ESDT is registered in Data Server, before test PGE run. 3. After ESDT is registered in Data Server, before test PGE run. 4. After PGE compilation, before test PGE run. 5. After PGE testing is complete, at time of promotion to Production, as needed to edit/review SSAP components. 6. As needed to retrieve MCF.

Quick Start Using SSIT Manager

The SSIT Manager provides a common interface to the SSI&T tools. An overview of the SSIT Manager GUI is provided in Section 2.1.2. A more detailed discussion of the tools accessed via this GUI is provided in subsequent sections.

The following assumptions are made with regard to the use of the SSIT Manager application.

- The operator is located at a workstation or server to which the SSIT Manager has been configured
- The operator has proper authorization to access the PDPS/SSIT database and the Data Server
- To access files in ClearCase, the operator has a ClearCase view already set
- The operator's environment has been configured as documented in the pertinent sections of the Help menu, available from the main window of the SSIT Manager The Index submenu of the Help menu provides access, through the Netscape browser, to a number of topics that help the operator in the environment configuration. The Help menu contains a list of topics that can be searched through.

To start SSIT Manager at the Unix command line:

DpAtMgr ConfigFile <config_filename> ecs_mode <mode> &

where:

config_filename is the name of the user's personal Process Framework configuration file for this program, as customized from **$ECS_HOME/CUSTOM/cfg/**.

ecs_mode is the ECS mode of operation, e.g., OPS, TS2

Invoking SSIT Manager From the Command Line Interface

To start SSIT Manager at the Unix command line:

DpAtMgr ConfigFile <config_filename> ecs_mode <MODE> &

where config_filename is the name of the user's personal Process Framework configuration file for this program, as customized from **$ECS_HOME/CUSTOM/cfg/**.

Sun Platform

Table 2.1-2 lists the SSI&T command line interfaces for the Sun workstation.

Table 2.1-2. Command Line Interfaces (Sun) (1 of 2)

Command Line Interface	Description and Format	When and Why Used
EcDpAtMgr	Startup of SSIT Manager.	To do SSI&T, and record items accomplished in the log.
EcDpAtMgrLogDump	Used to dump/print a log file to the screen.	As needed.
xterm	Open a Unix command line window.	As needed.
sparcworks	Ad hoc code analysis	As needed.
ghostview	Postscript file viewer	As needed.
netscape	WWW browser Netscape	As needed.
acroread	PDF file viewer Adobe Acrobat	As needed.
DpAtMgrForcheck	FORTRAN 77 code analysis	Determine whether FORTRAN 77 science software adheres to standards.
EcDpAtBadFuncGui	Prohibited function checker (GUI)	Determine whether science software adheres to standards.
EcDpAtBadFunc	Prohibited function checker (command line)	Determine whether science software adheres to standards.
EcDpAtCheckPCF	Process Control File checker (GUI)	Determine whether PCF is valid.
EcDpAtMgrPrologs	Prolog extractor	Extract science software code prologs.
/data/IDL/idl_4/bin/idl	IDL	As needed.
EOSView	EOSView	HDF file viewer.
EcDpAtMgrXdiff	ASCII file comparison	Compare 2 text files.
EcDpAtBinDiffGui	Binary file difference environment	Compare 2 binary files.
DpAtCheckHdfFile	HDF file comparison (GUI)	Compare 2 HDF files.
DpAtHdiff	HDF file comparison (command line)	Compare 2 HDF files.
xedit	Text editor	As needed.
emacs	Text editor	As needed.
EcDpAt CreateODLtmplate	Create PGE metadata ODL template file.	Before running EcDpAtDefinePGE .
EcDpAtVerifyODL	Verify PGE metadata ODL template file.	Before Running EcDpAtDefinePGE.
EcDpAtDefinePGE	Update PDPS/SSIT database with SCIENCE metadata.	Before executing PGE in SSIT environment.
DpAtOpDbGui	Update PDPS/SSIT database with OPERATIONAL metadata.	Before executing PGE in SSIT environment.
EcDpAtStageDAP	Acquires DAP from the Data Server.	After email subscription notification received.
DpAtInsertStatic	Inserts static input file into the Data Server.	Before executing PGE in SSIT or Production environment.
DpAtInsertTest	Inserts test dynamic input file into the Data Server.	Before executing PGE in SSIT environment.

Table 2.1-2. Command Line Interfaces (Sun) (2 of 2)

Command Line Interface	Description and Format	When and Why Used
DpAtInsertExeTar	Inserts tar file of executables, etc. needed to run PGE file into the Data Server.	Before executing PGE in SSIT or Production environment.
EcDpAtSSAPGui	Edit and inserts a single SSAP component into the Data Server.	After SSI&T is finished, before official promotion to Production.
netscape <html page name>	HTML pages for acquiring SSAP components from the Data Server, including test outputs.	During SSI&T, to get test outputs; After SSI&T is finished.
EcDpAtaCQUIREMCF	Get ESDT from the Data Server and insert MCF.	Before inserting MCF in the Data Server.

SGI Platform

It is intended that the SSI&T tools be most often run from the SSIT Manager. A small number of SSIT tools run only on the SGI platform. Because of security considerations, these tools cannot be run from the SSIT Manager on the Sun. They may only be run from the Unix command line on the SGI platform as indicated in Table 2.1-3.

Table 2.1-3. Command Line Interfaces (SGI)

Command Line Interface	Description and Format	When and Why Used
usr/sbin/cvproj	ProDev Workshop: Used for ad hoc analysis of science software.	Used to determine causes of problems (e.g., memory leaks).
DpAtRusage	Measures PGE performance.	Output of this tool is to be typed into the “Performance Statistics” section of the PROFILE screen of the PDPS/SSIT Database Update GUI.

Table 2.1-4 lists SGI platform tools associated with the SSIT process.

Table 2.1-4. SGI Tools Description

Categories /Tools	Tool Description & Use	Further Information
ProDev Workshop	• ProDev Workshop is a COTS package developed by SGI • This tool is targeted within ECS for applications running on the SGI science processors • ProDev Workshop is a software development support tool which includes several tools that may have applicability to SSIT • Among these tools is the capability to perform static code analysis to aid in the detection of memory leaks	• ProDev Workshop includes online documentation describing its features • Other ProDev Workshop documentation is delivered with ECS. • ProDev Workshop is not available from the SSIT menu. This tool must be started from the Command Line Interface: see Table 2.1-3.
PGE Performance	• DpAtRusage is a custom tool developed by ECS • It measures performance parameters such as CPU time used for a PGE linked to the SDP Toolkit, SCF version.	A help message is printed if the tool is invoked without input parameters.

Across the top of the SSIT Manager GUI are the toolbar items **File**, **Tools**, and **Run**. Clicking on each of these invokes a pull-down menu.

Under the **File** pull-down menu, the only item is **Exit**. Clicking on this causes the SSIT Manager to terminate.

The **Tools** pull-down menu has most of the SSIT Manager's tools. The menu functions are:

The Run pull-down menu initially contains no menu items. Its purpose, however, is to allow a place for SSI&T personnel to place their own custom tools and cripts SSIT Manager GUI

This GUI (Figure 5) is the starting point for SSI&T activities. It provides access to a collection of tools that will be useful for this purpose.

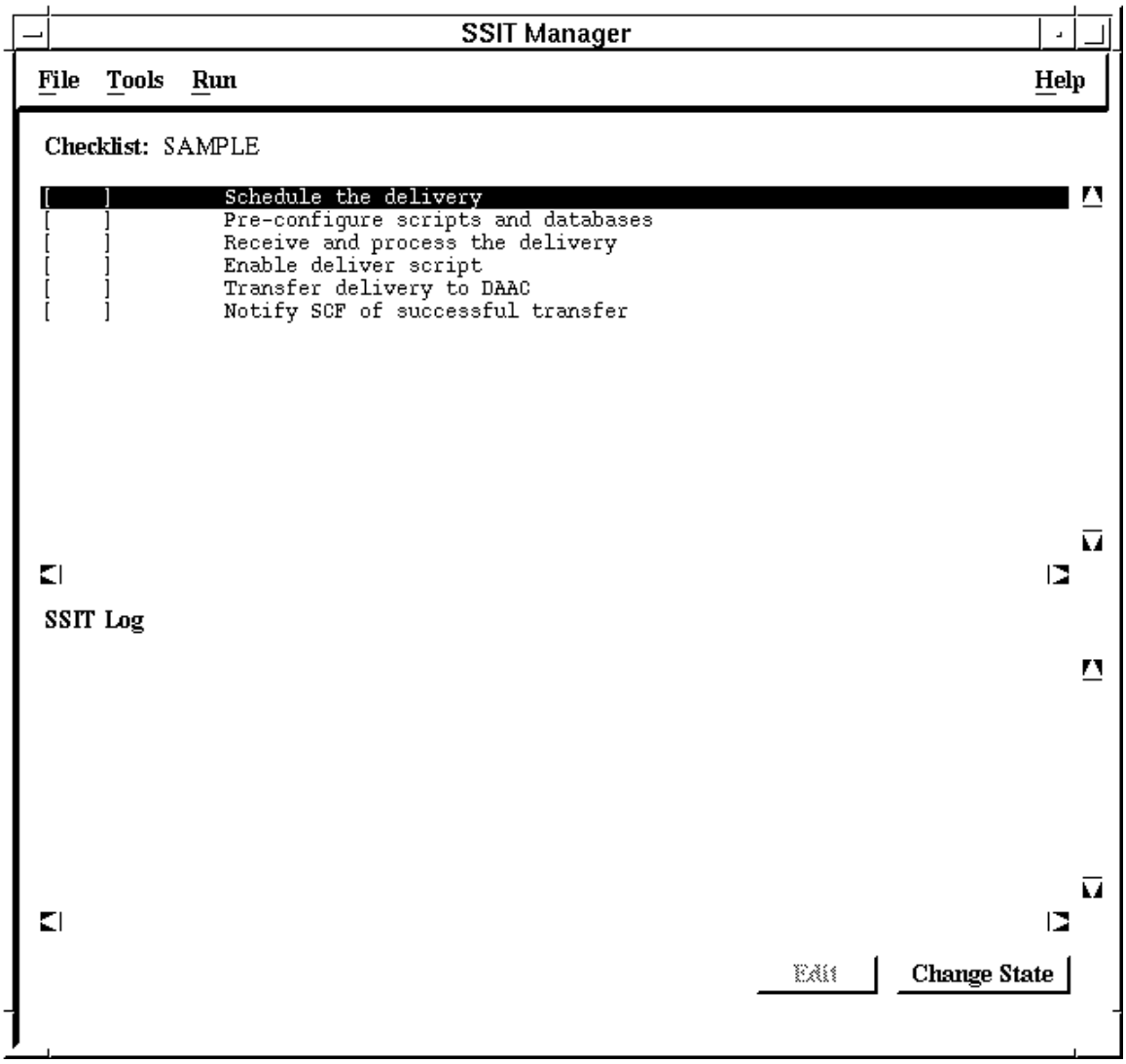


Figure 5. SSIT Manager Window

General Set Up of the SSIT Manager

The SSIT Manager requires a configured environment within which to run; it runs only on the AIT Suns. The set up steps described in this section need only be done the first time a SSI&T operator uses the SSIT Manager

To set up the environment for the SSIT Manager, execute the procedure steps that follow.

On workstation **x0ais##,** at the UNIX prompt in a terminal window, type as in step 1 below your **user id** and **password.**

NOTE: The **x** in the workstation name will be a letter designating your site:

g = GSFC, m = SMC, l = LaRC, e = EDC, n = NSIDC, o = ORNL, a = ASF, j = JPL, p=PVC; the ## will be an identifying two-digit number (e.g.,g0ais01 indicates a Data Processing Subsystem (DPS) workstation at GSFC).

Enter **setenv DISPLAY <local_workstation IP address>:0.0** after the rlogin, before entering the command. The **<ipaddress>** is the ip address of **x0ais##,** and xterm is required when entering this command on a Sun terminal.

1 This procedure was tested on an SSI&T server by a telnet to **p0ais01**, **ID:, PW:,** or. **setenv DISPLAY clientname:0.0** and then press the **Enter** key.

2 The ***mode*** is the ECS mode in which you are operating. This mode should be **TS1** or another mode assigned beforehand to operate in.

For example, cd /usr/ecs/TS1/CUSTOM/utilities and then press the Enter key.

3 This directory should contain scripts pertaining to setting the environment for SSIT Manager. Type in: **EcDpAtMgrStart <mode>.** This invokes the **SSIT Manager GUI** that should be displayed.

SSIT Manager Tools

There are several tools that are accessible through the SSIT Manager GUI. After selecting the TOOLS menu option of the menu bar, a set of options is available. See Figure 6, which indicates the use of the Tool menu item.

Using the SSIT Manager:

The following is a list of tools, and or assumptions:

The SSIT Manager is running.

The source file(s) are available, accessible, and have read permissions.

The below listed formatted text (ASCII) files containing the list of prohibited functions exist in the directory stored in the environment variable DPATMGR_DAT:

prohibitedFunctionsAda

prohibitedFunctions.C++

prohibitedFunctions.C

prohibitedFunctions.F77

prohibitedFunctions.F90

If the source code files to be checked are in a VOB in ClearCase, a view has been set before the SSIT Manager was started.

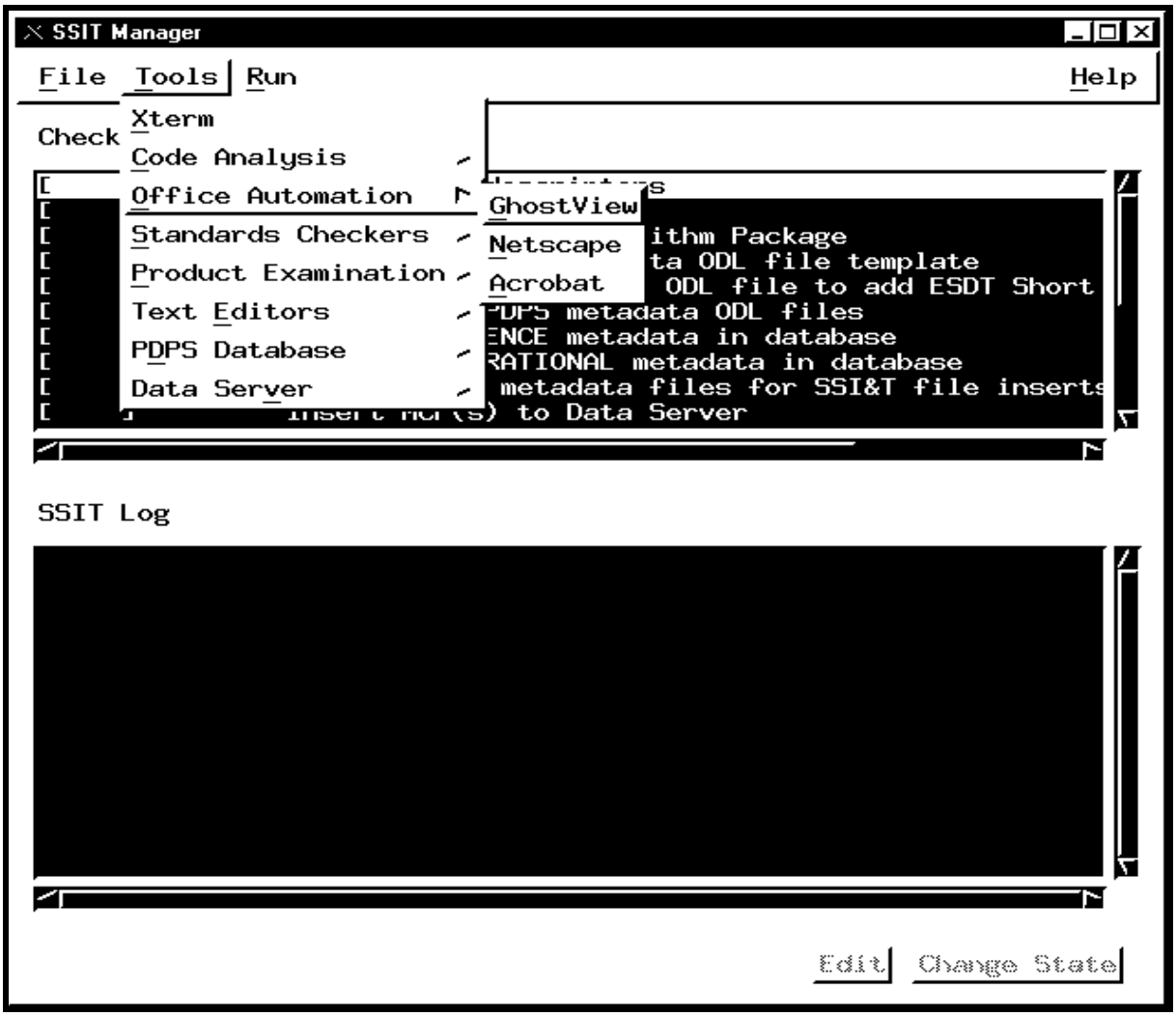


Figure 6. SSIT Manager Window - Tools Menu

This page intentionally left blank.

Acquiring and Unpacking the Delivered Algorithm Package (DAP)

The SSIT team receives the science software from the science community instrument teams. The following procedures will provide the SSIT team with the procedural steps that are used to acquire files.

Acquiring the Algorithm Package via FTP

Acquiring the science software via FTP is another method that the SSIT team will use in order to receive the science software. For this training class, a simple synthetic product generation executive (PGE) has been prepared for use by the students. This example will be used to demonstrate the FTP of the tar file from a remote machine. The students will then copy the tar file from the instructor's home directory.

Acquiring the Algorithm Package via FTP

1 Log into one of the AIT Sun workstations by typing: **username** then press the **Enter** key.

On workstation **x0ais##**, at the UNIX prompt in a terminal window, type as in step 3 below your **user id** and **password**.

NOTE: The **x** in the workstation name will be a letter designating your site:

g = GSFC, m = SMC, l = LaRC, e = EDC, n = NSIDC, o = ORNL, a = ASF, j = JPL, p=PVC; the ## will be an identifying two-digit number (e.g.,g0ais01 indicates a SSIT workstation at GSFC).

Enter the **password** then press the **Enter** key.

4 At a UNIX prompt, type **cd *DeliveryPathname***, then press the **Enter** key.

The ***DeliveryPathname* is** the full path name to the directory that has been set aside for ftp pull of DAPs from the Instrument Team. For example, **cd /home/*user*** where ***user*** is the user's login directory, then press the **Enter** key.

If the DAP is to be copied into a subdirectory, change to this subdirectory.

5 At a UNIX prompt, type **ftp *machineIPaddress***, then press the **Enter** key.

The ***machineIPaddress*** is the IP address or fully qualified domain name of the remote SCF machine. For example, **ftp 192.116.53.2**, then press the **Enter** key.

6 At the ftp prompt on the remote machine, enter user login name, then press the **Enter** key.

The remote machine will typically respond with **331** Password required for ***username***:

7 At the ftp prompt on the remote machine, enter user password, then press the **Enter** key.

The remote machine will typically respond with **230** User ***username*** logged in and display the ftp> prompt for further ftp commands.

8 At the ftp prompt on the remote machine, type **cd *DAPpathname*** then press the **Enter** key.

The ***DAPpathname*** is the full path name to the directory on the remote machine containing the DAP to retrieve. For example, **cd /home/mac** , then press the **Enter** key. The directory location should be known.

9 At the ftp prompt on the remote machine, type **binary**, then press the **Enter** key.

The binary command causes subsequent file transfers to be in binary mode, preserving the integrity of the file to retrieve without interpretation (as would be done in ASCII mode.

The system will typically respond with the message **200** Type set to I indicating that binary mode has been set.

10 At the ftp prompt on the remote machine, type **get *DAPfilename***, then press the **Enter** key.

The ***DAPfilename*** is the file name of the DAP to retrieve.

For example, type get TestPGE.tar, then press the **Enter** key.

The user may need to type **dir** then press **Enter** to display a listing of the files in the current directory. The system will likely display several lines of messages once the transfer has completed. For large files, this may take a long time (minutes to hours depending upon the size of the DAP and the bandwidth of the connection).

11 At the ftp prompt on the remote machine, repeat step **8** or type **bye**, then press the **Enter** key.

Typing bye and pressing Enter closes the ftp connection with the remote machine.

Retrieve other DAP files by repeating step 8. The DAPs retrieved will reside in ***DeliveryPathname*** on the local machine.

Unpacking a DAP

Once a DAP has been acquired via electronic means or physical media, it typically needs to be unpacked before its contents are accessible for SSI&T. Several mechanisms are available under standard UNIX for packing and unpacking files to and from a file archive, the most common being UNIX *tar*.

Unpacking a DAP

On workstation **x0ais##**, at the UNIX prompt in a terminal window, type as in step 1 below your **user id** and **password.**

NOTE: The **x** in the workstation name will be a letter designating your site:

g = GSFC, m = SMC, l = LaRC, e = EDC, n = NSIDC, o = ORNL, a = ASF, j = JPL; p=PVC, the ## will be an identifying two-digit number (e.g.,g0ais01 indicates a Data Processing Subsystem (DPS) workstation at GSFC).

Log into one of the AIT Sun workstations where the retrieved DAP resides by typing: username then press the **Enter** key.

3 Enter the **password** then press the **Enter** key.

4 At a UNIX prompt, type **cd *UnpackPathname***, then press the **Enter** key.

The ***UnpackPathname*** is the path name of the directory that has been set aside for unpacking of DAPs.

This directory now contains the DAP tar file. For example, **cd /home/*user***, where ***user*** is the user's login directory, then press the **Enter** key.

5 If the tar file is compressed, at a UNIX prompt, type **uncompress *PackedDAP.Z***, then press the **Enter** key.

The ***PackedDAP.Z*** is the file name of the compressed DAP file.

The file name extension of **.Z** is a convention indicating UNIX compressed files. The uncompress utility expects this file name extension by default. A resulting error may indicate that the DAP file was not compressed or that another compression utility was used. If the file name extension was **.Z**, the uncompressed version will have the same file name but without the **.Z,** for example ***PackedDAP***.

The tar file for the SSI&T Training will not be compressed.

6 At the UNIX prompt, type **tar xvf *PackedDAP***, then press the **Enter** key.

The ***PackedDAP*** is the file name of the uncompressed DAP file.

The tar archive will be unpacked in the current directory. If the archive contained directories and subdirectories, these will be created by the tar utility and populated by the files that belong.

SSIT Software Operating Instructions

Starting the SSIT Manager GUI:

1 Log into an Algorithm and Test Tools (AITTL) environment using a machine so configured. At the mini-daac this machine is **p0ais01.** A special host has been established using the id: and password:. Type: **setenv DISPLAY clientname:0.0** and then press the **Enter** key**.**

2 This directory should contain scripts pertaining to setting the environment for SSIT Manager. Type in: **EcDpAtMgrStart <mode> &**

This invokes the SSIT Manager GUI, which should be displayed.

What must be done via SSIT tools:

Since SSIT is just a calibration of various tools, there is no specific order for which they must be run. Most tools can be brought up from the SSIT Manager GUI as well as started on their own.

The File menu provides the capability to exit the manager. The Tools menu provides access to the various tools that make up SSIT. The Run menu is customizable (allowing you to add your own scripts and tools) by editing the file *ssit_run_menu* in the *data/DPS* directory.

The checklist (first window on the GUI) allows you to check off various activities by double clicking on them. You may enter a commentary on the activity in the second window when checking off a particular item. The file checklist.sample in the data/DPS directory can be edited to change the items in the checklist or its' location.

Subscribing to the DAP:

The following Servers/Services must be up and operational:

Data Server, Advertising Service, & Subscription Server.

The following must have occurred between those Servers/Services:

Data Server must have inserted the DAP ESDT. The DAP ESDT services must have been posted to the Advertising Service by Data Server. Advertising must have approved subscriptions on the DAP ESDT.

What the user must do before trying SSIT functionality:

1 telnet to (CSS) **p0ins01** (Subscription Server)

2 login: **ID, password**:

3 **cd /usr/ecs/TS1/CUSTOM/utilities**

4 **runOpGui.csh TS1** (ECS Subscription Service GUI should appear)

5 Select Add Subscription…

Add/Edit Subscription GUI will now appear

6 Use Browse Events to choose Event ID.

7 Place Event ID into User ID slot.

8 Place your Email Address in slot provided. (the subscription notification will be returned via email).

9 Place Email Text Identification in slot provided. (Limited to one Word)

10 **Start Date Group** must be the current Date.

11 **Stop Date** is a date in the future or if ran this day, use current Date.

12 Select Action

13 Enter the Data Type Id — ***DAP* + # + <*ESDT Version*> --** of the ESDT that you wish to search for (probably *DAP#1*).

Do NOT override the provider, submit the subscription and use the default service (which is Notification of the Insert of a granule). A good status message should be displayed.

Email will be returned when a DAP is Ingested (by Ingest)

DAP Insert

The **DAP** (Delivered Algorithm Package) is the vehicle by which the PGE, source code, supporting files, documentation, etc. are delivered to a DAAC for SSI&T. Typically, the DAP is a compressed TAR file with a file name of form ***string*.tar.Z** . After initial processing, the DAP is broken apart into its components and those components will be subsequently processed and used based on their intended function.

The **insert** service is used to put the DAP into the Data Server. Once the DAP is in the Data Server, the **acquire** service is used to retrieve it.

Performing a DAP Insert

The method described here, for performing a DAP insert, is by command-line promptsand responses.

Assumptions:

1. The DAP ESDT has been installed on the Data Server.
2. The Target MCF for the DAP has been created for the Insert.
3. The SSIT Manager is running. This procedure was tested using **p0acs03**.

To Insert a DAP to the Science Data Server, execute the following steps:

1 From the SSIT Manager, click on the **Tools** menu, then choose Data Server and then **Insert Test Dynamic.**

An xterm with title "SSIT: PGE Test Dynamic Input File Insertion" will be displayed.

2 At the program prompt **Configuration filename? (enter for default: ../..cfg/EcDpAtInsertTestFile.CFG)** and then press the **Enter** key.

3 At the program prompt **ECS Mode of operations?**

Type in the **<mode>** you are working in. For example, **TS1** or **OPS.** Press **Enter.**

4 At the program prompt **ESDT short name for the file(s) to insert?** type ***ESDTShortName***, press **Enter**

The ***ESDTShortName*** is the ShortName for the DAP file, which is DAP.

5 At the program prompt **ESDT Version for the file(s) to insert?** Type in the ESDT version and press **Enter.**

6 At the program prompt **Is there more than one data file to this Dynamic Granule (Y = Yes, N = No)? (enter for default: N)?**

7 At the program prompt **Single Filename to Insert? (including FULL path)** type ***pathname/GranuleFilename***, press **Enter**.

The ***pathname/GranuleFileName*** is the full path name and DAP file name.

8 At the program prompt **Associated ASCII Metadata Filename to Insert (including FULL path)** , Type ***pathname/DAP.tar.met*** and press **Enter**.

pathname is full name of the path and ***DAP.tar.met*** is the name of the associated **.met file.**

9 At the program prompt **Hit enter to run again, 'q <enter>' to quit:** type **q** and press **Enter** to quit or just press **Enter** to insert additional dynamic granules.

If continuing, repeat steps 2 through 8.

An Example of a DAP Metadata File

The following is an example of a DAP target metadata file. Three attribute values needs to be modified based on each DAP. The attributes are DAPPGEName, DAPPGEVersion, and DAPSWVersion.

The following is an example of a faked.hdf metadata file. Three attribute values needs to be modified based on each DAP. The attributes are **DAPPGEName, DAPPGEVersion**, and **DAPSWVersion.**

```
GROUP = INVENTORYMETADATA
..... .
                GROUP                = INVENTORYMETADATA
                  GROUPTYPE            = MASTERGROUP

                  GROUP                = ECSDATAGRANULE

                    OBJECT               = LOCALGRANULEID
                      NUM_VAL              = 1
                      VALUE              =
                "MOD04L_O.A1997226.0704.002.1999204222146.hdf"
   END_OBJECT           = LOCALGRANULEID

                    OBJECT               = PRODUCTIONDATETIME
                      NUM_VAL              = 1
    VALUE                = "1999-07-23T22:21:46.000Z"
                    END_OBJECT             = PRODUCTIONDATETIME

                    OBJECT               = DAYNIGHTFLAG
                      NUM_VAL              = 1
                      VALUE              = "Day"
   END_OBJECT           = DAYNIGHTFLAG

                    OBJECT               = REPROCESSINGACTUAL
                      NUM_VAL              = 1
                      VALUE              = "processed once"
   END_OBJECT           = REPROCESSINGACTUAL

                    OBJECT               = LOCALVERSIONID
                      NUM_VAL              = 1
    VALUE                = "002"
                    END_OBJECT             = LOCALVERSIONID

                    OBJECT               = REPROCESSINGPLANNED
                      NUM_VAL              = 1
    VALUE                = "further update is anticipated"
                    END_OBJECT             = REPROCESSINGPLANNED

                  END_GROUP              = ECSDATAGRANULE

                  GROUP                = COLLECTIONDESCRIPTIONCLASS

                    OBJECT               = VERSIONID
                      NUM_VAL              = 1
                      VALUE              = 1
   END_OBJECT           = VERSIONID

                    OBJECT               = SHORTNAME
                      NUM_VAL              = 1
    VALUE                = "MOD04L_O"
                    END_OBJECT             = SHORTNAME
```

```
  END_GROUP             = COLLECTIONDESCRIPTIONCLASS

  GROUP                 = INPUTGRANULE

    OBJECT              = INPUTPOINTER
      NUM_VAL             = 20
      VALUE               =
("MOD04_L2.A1997226.0725.002.1999204060629.hdf",
"MOD04_L2.A1997226.0730.002.1999204060716.hdf",
"MOD04_L2.A1997226.0735.002.1999204060620.hdf",
"MOD04_L2.A1997226.0740.002.1999204060637.hdf",
"MOD04_L2.A1997226.0745.002.1999204060647.hdf",
"MOD04_L2.A1997226.0750.002.1999204060644.hdf",
"MOD04_L2.A1997226.0800.002.1999204195105.hdf",
"MOD04_L2.A1997226.0805.002.1999204194908.hdf",
"EOSAM1_1997-08-14.eph", "EOSAM1_1997-08-14.att")
    END_OBJECT            = INPUTPOINTER

  END_GROUP             = INPUTGRANULE

  GROUP                 = PGEVERSIONCLASS

    OBJECT              = PGEVERSION
      NUM_VAL             = 1
      VALUE               = "2.3.0"
    END_OBJECT            = PGEVERSION

  END_GROUP             = PGEVERSIONCLASS

  GROUP                 = RANGEDATETIME

    OBJECT              = RANGEENDINGDATE
      NUM_VAL             = 1
      VALUE               = "1997-08-14"
    END_OBJECT            = RANGEENDINGDATE

    OBJECT              = RANGEENDINGTIME
      NUM_VAL             = 1
      VALUE               = "08:45:02"
    END_OBJECT            = RANGEENDINGTIME

    OBJECT              = RANGEBEGINNINGDATE
      NUM_VAL             = 1
      VALUE               = "1997-08-14"
    END_OBJECT            = RANGEBEGINNINGDATE

    OBJECT              = RANGEBEGINNINGTIME
      NUM_VAL             = 1
      VALUE               = "07:04:58"
    END_OBJECT            = RANGEBEGINNINGTIME
```

```
  END_GROUP             = RANGEDATETIME

  GROUP                 =
ORBITCALCULATEDSPATIALDOMAIN

    OBJECT                =
ORBITCALCULATEDSPATIALDOMAINCONTAINER
      CLASS               = "1"

      OBJECT              = EQUATORCROSSINGDATE
        CLASS             = "1"
        NUM_VAL           = 1
        VALUE             = "1997-08-14"
      END_OBJECT          = EQUATORCROSSINGDATE

      OBJECT              = EQUATORCROSSINGTIME
        CLASS             = "1"
        NUM_VAL           = 1
        VALUE             = "07:52:01.841571Z"
      END_OBJECT          = EQUATORCROSSINGTIME

      OBJECT              = ORBITNUMBER
        CLASS             = "1"
        NUM_VAL           = 1
        VALUE             = 24565
      END_OBJECT          = ORBITNUMBER

      OBJECT              = EQUATORCROSSINGLONGITUDE
        CLASS             = "1"
        NUM_VAL           = 1
        VALUE             = 0.700127
      END_OBJECT          =
EQUATORCROSSINGLONGITUDE

    END_OBJECT            =
ORBITCALCULATEDSPATIALDOMAINCONTAINER

  END_GROUP               =
ORBITCALCULATEDSPATIALDOMAIN

  GROUP                   = SPATIALDOMAINCONTAINER

    GROUP                 =
HORIZONTALSPATIALDOMAINCONTAINER

      GROUP               = GPOLYGON

        OBJECT            = GPOLYGONCONTAINER
          CLASS           = "1"

          GROUP           = GRINGPOINT
```

```
        CLASS              = "1"

        OBJECT              = GRINGPOINTLONGITUDE
          NUM_VAL             = 6
          CLASS              = "1"
          VALUE               = (160.828918, 57.056137, 44.058830,
-
1.859100, 34.022232, -114.758156)
        END_OBJECT            = GRINGPOINTLONGITUDE

        OBJECT              = GRINGPOINTLATITUDE
          NUM_VAL             = 6
          CLASS              = "1"
          VALUE              = (69.526398, 23.441734, -65.489883, -
58.831890, 26.757355, 81.381241)
        END_OBJECT            = GRINGPOINTLATITUDE

        OBJECT              = GRINGPOINTSEQUENCENO
          NUM_VAL             = 6
          CLASS              = "1"
          VALUE              = (1, 2, 3, 4, 5, 6)
        END_OBJECT            = GRINGPOINTSEQUENCENO

       END_GROUP            = GRINGPOINT


       GROUP               = GRING
        CLASS              = "1"

        OBJECT              = EXCLUSIONGRINGFLAG
          NUM_VAL             = 1
          CLASS              = "1"
          VALUE              = "N"
        END_OBJECT            = EXCLUSIONGRINGFLAG

       END_GROUP            = GRING

      END_OBJECT            = GPOLYGONCONTAINER

    END_GROUP            = GPOLYGON

   END_GROUP            =
HORIZONTALSPATIALDOMAINCONTAINER

  END_GROUP            = SPATIALDOMAINCONTAINER

  GROUP               =
ASSOCIATEDPLATFORMINSTRUMENTSENSOR

   OBJECT              =
ASSOCIATEDPLATFORMINSTRUMENTSENSORCONTAINE
R
```

```
    CLASS            = "1"

    OBJECT           = ASSOCIATEDSENSORSHORTNAME
      CLASS          = "1"
      NUM_VAL        = 1
      VALUE          = "CCD"
    END_OBJECT       =
ASSOCIATEDSENSORSHORTNAME

    OBJECT           =
ASSOCIATEDPLATFORMSHORTNAME
      CLASS          = "1"
      NUM_VAL        = 1
      VALUE          = "AM-1"
    END_OBJECT       =
ASSOCIATEDPLATFORMSHORTNAME

    OBJECT           =
ASSOCIATEDINSTRUMENTSHORTNAME
      CLASS          = "1"
      NUM_VAL        = 1
      VALUE          = "MODIS"
    END_OBJECT       =
ASSOCIATEDINSTRUMENTSHORTNAME

   END_OBJECT        =
ASSOCIATEDPLATFORMINSTRUMENTSENSORCONTAINE
R

  END_GROUP          =
ASSOCIATEDPLATFORMINSTRUMENTSENSOR

END_GROUP            = INVENTORYMETADATA

END
```

Mail Template

```
From ts2cm@t1ins01.vatc.ecs.nasa.gov Thu Jan  8 16:32 EST 1998
Received:      from      t1ins02      (t1ins02.vatc.ecs.nasa.gov
[198.118.232.41])
          by t1ins01.vatc.ecs.nasa.gov (8.8.6/8.8.4) with SMTP
        id QAA10950 for <dps@t1ins01.vatc.ecs.nasa.gov>; Thu, 8
Jan 1998 16:32:01 -0500 (EST)
Received: by t1ins02 (SMI-8.6) id QAA19469; Thu, 8 Jan 1998
16:32:01 -0500
From: Code2 Install Team <ts2cm@t1ins01.vatc.ecs.nasa.gov>
Date: Thu, 8 Jan 1998 16:32:01 -0500
Message-Id: <199801082132.QAA19469@t1ins02>
```

```
Subject: ECS Notification
To: dps@t1ins01.vatc.ecs.nasa.gov
Content-Type: text
Content-Length: 79
Status: RO

UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[VTC:DSSDSRV]:20:SC:MO
DPTQKM.001:5043
```

DAP MODIS PGE07-Training Example

Figure 7. MODIS I/O PGE07

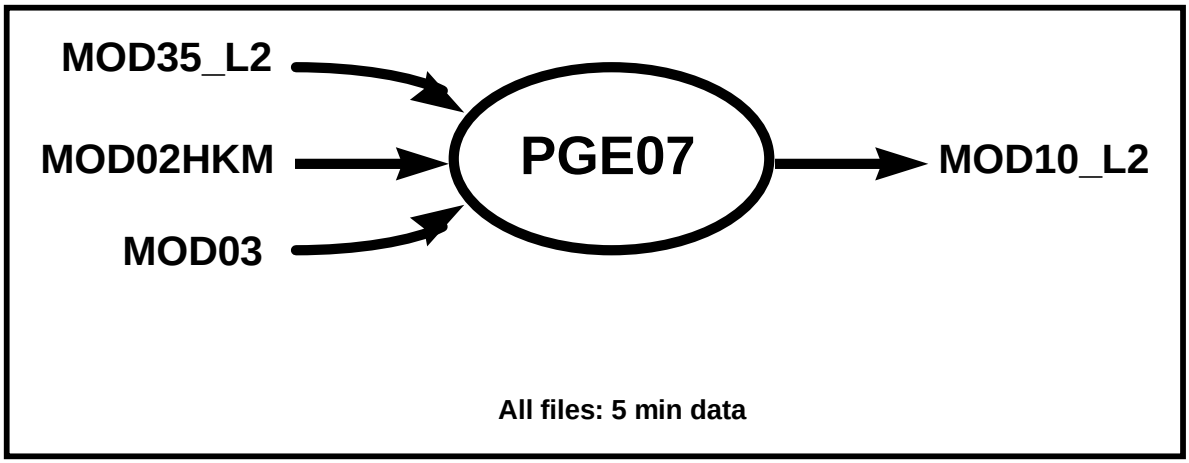


Figure 8. MODIS PGE07 SDSRV Files

Filename	Description
SDSRV:	
DsESDTMoMOD03.001.desc	Input ESDT descriptor file
DsESDTMoMOD02HKM.001.desc	Input ESDT descriptor file
DsESDTMoMOD35_L2.001.desc	Input ESDT descriptor file
DSESDTMoMOD10_L2.001.desc	Output ESDT descriptor file
libDsESDTMoMOD03.001Sh.so	Shared library for input ESDT
libDsESDTMoMOD02HKM.001Sh.so	Shared library for input ESDT
libDsESDTMoMOD35_L2.001Sh.so	Shared library for input ESDT
libDsESDTMoMOD10_L2.001Sh.so	Shared library for output ESDT

Filename	Description
PDPS:	
PGE07.tar	PGE executable
PGE07.tar.met	Target MCF for PGE executable
PGE_PGE07#1.0#01.odl	ODL for PGE07
ESDT_ MOD03#2.0.odl	ODL file for binary input granule
ESDT_ MOD02HKM#2.0.odl	ODL file for binary input granule
ESDT_ MOD35#2.0.odl	ODL file for binary input granule
ESDT_ MOD10_L2#2.0.odl	ODL file for binary output granule
MOD02HKM.A1996218.1555.002.hdf	Binary input data granule
MOD03.A1996218.1555.002.hdf	Binary input data granule
MOD35_L2.A1996218.1555.002.hdf	Binary input data granule

Figure 9. MODIS PGE07 PDPS Files

Example of a PGE Executables Tar File Insertion Script

This example was produced in Drop 4 and is provided for review only. The section below shows the placing of the Science Software Executable (SSEP) on the Data Server which includes the Insertion of a PGE Tar file..

```
Configuration filename?  (enter for default:
../../cfg/EcDpAtInsertExeTarFile.CFG)
ECS Mode of operations?  (enter for default: OPS)
Name of PGE?  (enter for default: PGE07)
Science software version of PGE?  (enter for default: 2)
Staged filename to insert (including FULL path)?  (enter for default:
/home/emcleod/SSEP/PGE07.tar)
Associated ASCII metadata filename to insert (including FULL path)?  (enter for
defaul /home/emcleod/SSEP/PGE07.tar.met)
Top level shell filename within tar file?  (enter for default: PGE07.csh)
PGE07.csh
Warning: Could not open message catalog "oodce.cat"
/usr/ecs//OPS/CUSTOM/bin/DPS/EcDpAtInsertExeTarFile: Process Framework:
ConfigFile ..//cfg/EcDpAtInsertExeTarFile.CFG  ecs_mode OPS
Performing INSERT......
Retrieved from IOS for ESDT = PGEEXE the DSS UR =
UR:15:DsShSciServerUR:13:[MDC:DSSDSR
Trying to make a request to [MDC:DSSDSRV]
Trying to make a request to [MDC:DSSDSRV]
Insert to Data Server and PDPS database update successful for:
   PGE name = 'PGE07'
   Ssw version = '2'
   ESDT = 'PGEEXE'
   ESDT Version = ''
   staged file = '/home/emcleod/SSEP/PGE07.tar'
   metadata file = '/home/emcleod/SSEP/PGE07.tar.met'
   Top level shell name = 'PGE07.csh'
Inserted at UR:
   R:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[MDC:DSSDSRV]:14:LM:PGEEXE:94'
Hit Enter to run again, 'q <Enter>' to quit:
```

DAP Acquire

The **DAP** (Delivered Algorithm Package) is the vehicle by which the PGE, source code, supporting files, documentation, etc. are delivered to a DAAC for SSI&T. Typically, the DAP is a compressed TAR file with a file name of form ***string*.tar.Z** . After initial processing, the DAP is broken apart into its components and those components will be subsequently processed and used based on their intended function.

The insert service is used to put the DAP into the Data Server. Once the DAP is in the Data Server, the acquire service is used to retrieve it.

DAP is acquired from Data Server and placed in the specified directory. Note there will be 2 files, the DAP itself (a big tar file) and the metadata associated with the DAP. The metadata may be helpful in the creating the SSAP.

Performing a DAP Acquire Using SSIT Manager

Generally, the preferred approach to accomplishing a DAP **acquire** will be through the use of the SSIT Manager GUI., see Figure 10.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The following servers/services are up and operational:
Data Server, Subscription Server, Storage management
2. The following must have occurred between those Servers/Services:
   - Ingest must have ingested DAP and Inserted it into the Data Server.
   - Subscription Server must have gotten notification from the Data Server of the Insert.
   - Subscription Server must send email to the SSIT operator notifying him/her of DAP Insertion and giving him (in the email) the UR of the DAP.
   - The SSIT Manager is available
   - The X Window **DISPLAY** environment variable is pointing to your screen

To perform a DAP acquire, execute the procedure steps that follow:

1 If not already on an AIT Sun, log onto one from your current machine.

2 Bring up the SSIT Manager GUI. At the UNIX prompt by typing **mgr.**

3 After a short while, the SSIT Manager GUI will appear. From the SSIT Manager top menu bar, select **Tools -> Data Server -> Acquire DAP** See figure 10. If the SSIT Manager GUI is used to initiate the DAP processing, Step 4 can be skipped.

4 Alternately, one can initiate the DPA processing sequence from the command line. To do this

5 Type **source /usr/ecs/TS1/CUSTOM/bin/DPS/.buildrc <ENTER>**
Note: This step only needs to be done once per login

6 Type **/usr/ecs/TS1/CUSTOM/bin/DPS/DpAtStageAlgorithmPackage.sh** <ENTER>

7 The user will be prompted with:
** DAP Staging Tool **
Configuration filename? (enter for default: DpAtAA.CFG)
To respond, type **<ENTER>**

8 The user will be prompted with:

ECS Mode of operations? (enter for default: OPS)

To respond, type **TS1 <ENTER>**

9 The user will be prompted with:
Name of email message file (including path)?
To respond, type the required file name plus the path, e.g.,
/home/diascone/emessage01.asc <Enter>

10 The user will be prompted with:
Directory to receive staged file?

11 To respond, type the required directory, e.g.,
/home/diascone/staged <Enter>

Acquire DAP using Ingest

This program is used to retrieve the Delivered Algorithm Package (DAP) from the Data Server.

Before its use, the following events must have occurred:

1. A subscription for the DAP must have been registered with the Data Server. The subscription delivery option will be sent to email, to a specified DAAC operator or maildrop.
2. The DAP must have been Ingested. There are two ways for this to occur. First, the DAP may be processed by Ingest. When this occurs, Ingest inserts the DAP into the Data Server, triggering subscription notification. Second, the DAP may be inserted with the Insert Test File.
3. After the DAAC operator received the email subscription notification, he or she must have saved it to a file.

To run the program, select Acquire DAP from the Data Server submenu. The program prompts for input parameters Process Framework configuration filename, email message filename and directory to receive staged file.

The program reads the DAP UR from the email file, acquires the DAP from the Data Server, and stages the DAP on the local disk. The name of the staged file is written to standard output. The name of the staged file is written to standard output.

Note, this function may be used to acquire any type of granule with a known UR.

After the file is staged, the operator may unpack it and install its components in ClearCase or in other places as appropriate.

NOTE: Use of this program is optional, in the sense that the DAP may arrive at the DAAC through other means than Ingest, e.g., simple ftp.

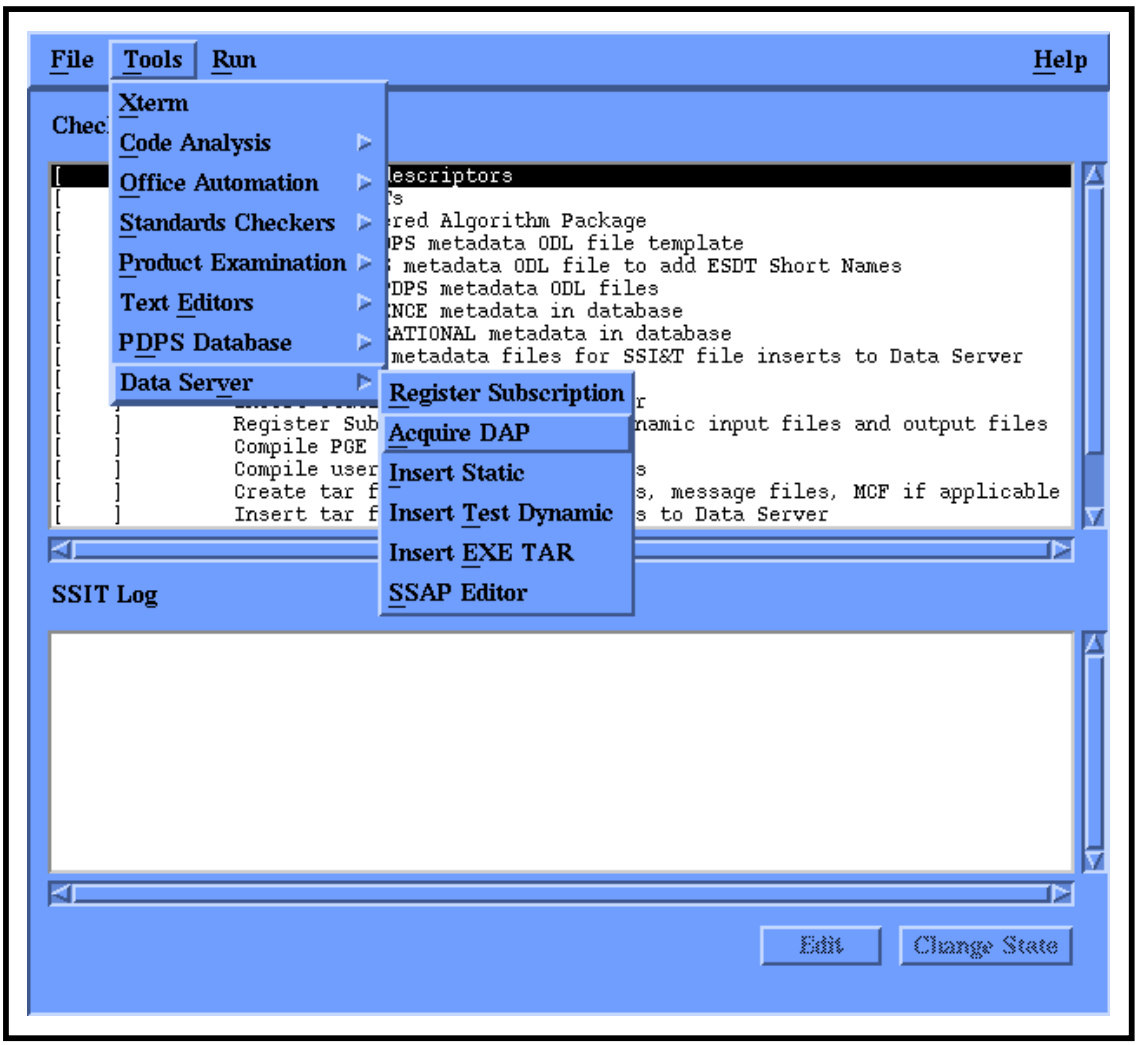


Figure 10. Selecting DAP Acquire from SSIT Manager

Notes: If the message indicates failure to acquire, see Appendix A.3 to diagnose the problem.

An Alternative Way to Acquire a DAP

The method described here, for performing a DAP acquire, is by command-line prompts and responses.

Assumptions:

1. All required Servers are up running.
2. The proper Sun Platform, Science Data Server host, is logged onto

To perform a DAP acquire, execute the procedure steps that follow:

Start acquire sequence by going to proper location and executing dsses

1 Type **cd /usr/ecs/<mode>/CUSTOM/bin/DSS** <**ENTER**>

2 Type **dsses** <**ENTER**>

3 The prompts and a user responses are shown below:

4 Enter dataserver UR (default is local dataserver [:DSSDSRV])=> **[MDC:DSSDSRV] <ENTER>**

5 Search for data

6 Insert data

7 Acquire data

8 Delete data

9 Exit

10 Please make selection=> **1**

11 Entering search routine.....

12 Enter data type=> **DAP**

13 Type UR

14 DAP UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[MDC:DSSDSRV]:17:DP:DAP.001:1364

15 Search for data

16 Insert data

17 Acquire data

18 Delete data

19 Exit

20 Please make selection=> **3**

21 There is(are)1 granules in the collection

22 Index = 0 DAP.001 DP:DAP.001:1364

23 Please enter the indices of the desired URs: **0**

24 Select Media Type:

FtpPull

FtpPush

8MM

4MM

CDROM

9TRK

D3

25 Enter media type: **2**

26 Enter mediaformat=> **FORMAT**

27 Enter userProfID=> **cmops**

28 Enter username=> **cmops**

29 Enter password=>

30 Enter host=> **p0acs03**

31 Enter destination=> **/home/cmops**

32 Acquire successful. Please <CR> to continue.

33 Search for data

34 Insert data

35 Acquire data

36 Delete data

37 Exit

38 Please make selection=> **5**

39 Exiting program....

Notes: The message above **Acquire successful** only indicates Science Data Server side, a user may still not get the DAP into final destination. Then see Appendix A.3 to diagnose the problems.

Inserting a Science Software Archive Package (SSAP)

The Science Software Archive Package (SSAP) is a grouping of science software, documentation, and other related files that is stored at the DAAC. They are accessed and updated through an SSIT SSAP GUI. The SSAP is supposed to be a record of the software, complete with source code, documentation, what and how it was tested, etc., and is created both for recording purposes and so that the tests can/could be repeated later. Note that the executables and any static files needed by the PGE are stored separately from the SSAP.

The SSAP is similar, but not identical to the DAP (Delivered Algorithm Package), in that it contains test data, source code, etc. Much of what is in the DAP will make it into the SSAP (although it may be modified, i.e., code may have bug fixes). Basically, the DAP is what arrives at the SSIT doorstep, while the SSAP is a similarly packaged finished product of the SSIT process, and so contains some things that were not in the DAP (and vice versa).

The SSAP will be made up of 2 different Data Types at the Data Server. The Algorithm Package is metadata about the SSAP, such as the name of the PGE, name of the instrument, date accepted, etc. Each part of the SSAP (source code, documentation, test data) will be stored as an SSAP component, with its own metadata in addition to the files. SSAP components such as source code will be tarred to retain the directory structure (so they must be untarred when retrieved). The executables and static files are stored separately from the SSAP, and thus have their own Data Types (ESDTs).

What follows is a list of items in the SSAP (taken from the DID205). See also the Core Metadata model under DAP for a graphical representation of the SSAP.

The following make up an SSAP:

Documentation

–Delivery Memo
–Summary Information for each PGE.
–System Description Document (SDD).
–Operations Manual
–Processing Files Description Document
–Test Plans (these include the test cases)
–Scientific documents
–Interface Definition Document
–Detailed Performance Testing Results
–Detailed design/implementation documents
–COTS User or Programmer Guides

Software & Control files:

–Science software source code (including make files & scripts)
–Testing software source code (including make files & scripts)

–Test Data Input (this may only be the UR for this)
–Expected Test Output
–Coefficient Files
–Process Control File
–Metadata Configuration File

–ODL files. These define the PGE and its related Data Types to the PDPS database. They don't currently have official names.

Other files:

–A change log created by the SSAP GUI to track changes to the SSAP.

Inserting a SSAP into PDPS

This procedure describes how to insert an SSAP into PDPS.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. All required Servers are up running.
2. SSI&T is up running. (see section 6 how to bring up SSIT Manager)
3. The C shell (or a derivative) is the current command shell.

FORCHECK is available only on the AIT Suns.

To create the SSAP, execute the procedure steps that follow:

1 If not already on an AIT Sun, log into one from your machine.

2 Launch the SSIT Manager. (see section 6).

3 From the SSIT Manager choose ***Tools*** menu and then ***Data Server*** submenu. Choose ***SSAP Editor***.

The GUI starts. Note that it will first query Data Server for a list of SSAPs that have previously been created. This list will appear in the window at the center (if any SSAPs already exist). Current SSAP field will be blank, and only Refresh and Create buttons will be active. All three tabs (Main, Files, and Metadata) will be active.

4 Click on ***Create*** to create a new SSAP.

The Create SSAP window appears. If no OK button is visible, resize the window so that the OK button is visible.

5 Enter **the name** of the SSAP in the first field.

6 Enter **SSAP version** in the second field. Note that version has a limit of 20 characters.

7 Click *OK* and the window disappears.

On the main GUI, the SSAP created (what was entered in the step above) will appear. Current SSAP is now set to that value. All buttons are now active.

8 Click on the ***File List tab*** to set up SSAP components.

The File List Tab displays files in the local directory to the left and files in the selected SSAP component to the right. On the bottom left is a directory listing and a method to move through the directory tree on the local machine. Delete and Reset buttons -- both active – are to the right.

9 Click on the ***File Type*** button to select the SSAP component to manipulate.

10 Choose **one** of the menu items.

11 Select **a file (or files)** from the left window to add to the component.

12 Click the ***Add Arrow*** button to add the files. They will appear in the right window because they are now part of that SSAP Component.

13 Now select the ***Metadata* tab** to set the metadata for the new SSAP. The Metadata Tab displays the metadata for the new SSAP. Only the Name and Version will be filled in automatically. The rest of the fields will have default information. While the SSAP can be submitted with the default information, it is wise to fill in valid values. To change a value, Click the mouse in the field you wish to change and type in a new value. For dates click in the first box or use the up/down arrows to move the date up or down. When finished entering a date, click the *OK* button. For text fields just hit the *Enter* key. The button marked "Edit Assoc Collections" on the bottom of the window must be hit and an Associated Collection entered for the SSAP.

14 Click the ***Edit Assoc Collections*** button.

The Edit Associated Collections window displays a list of associated collections and fields for the entry of new ShortNames and Versions (which make up an Associated Collection).

15 Enter a **shortname** (of an ESDT that has been installed in the Data Server) — must be eight or fewer characters. Note that the Data Server will verify if the Shortname exists.

16 Enter the **version** (of the installed ESDT).

17 Click ***OK*** and the new entry to the collection should appear in the window.

18 Click ***Done*** to close the window.

19 Click on the **Metadata tab.**

20 Click ***Save*** to save the updated metadata.

21 Click ***Main tab*** to get back to the Main tab.

22 Click ***Submit*** to send the new SSAP to Data Server. When finished, a message should pop up that says **"SSAP Successfully inserted to the Data Server"**.

This page intentionally left blank.

Updating a Science Software Archive Package (SSAP)

The Science Software Archive Package (SSAP) is a grouping of science software, documentation, and other related files that is stored at the DAAC. For a discussion of the SSAP and its contents

Updating a SSAP

This procedure describes how to update an existing SSAP in PDPS.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. All required Servers are up running.
2. An SSAP must already have been inserted into the Data Server.
3. The C shell (or a derivative) is the current command shell.

FORCHECK is available only on the AIT Suns.

To update the SSAP, execute the procedure steps that follow:

1 If not already on an AIT Sun, log into one from your machine.

2 Launch the SSIT Manager.

3 From the SSIT Manager choose Tools menu and then Data Server submenu. Choose SSAP Editor.

The GUI starts. Note that it will first query Data Server for a list of SSAPs that have previously been created. This list will appear in the window at the center (if any SSAPs already exist--if not, one will, of course, have to be created before the remainder of this procedure can be performed). Current SSAP field will be blank, and only Refresh and Create buttons will be active. All three tabs (Main, Files, and Metadata) will be active.

There is currently missing functionality in the SSAP Editor, so before updating the new SSAP you must hit the Refresh button to refresh the data about the new SSAP.

4 Click on the **Metadata tab** to update the SSAP.

The Metadata Tab displays the metadata for the SSAP. All fields will be set to the values entered when the SSAP was created, and the Algorithm Name field will be grayed out (because it may not be updated). If you want to create a new SSAP from the an existing one, go back to the Main tab and hit the Create With button.

5 Click on the Algorithm Version field **(**currently called **Algorithm Description**) and enter a new version (different from what is in the field when the tab is clicked).

6 Update any other fields that you wish to change. You can even add a new Associated Collection by clicking on the Assoc Collection button and following the steps described in Creating an SSAP.

7 Before you leave the Metadata tab, click **Save** to save the updated metadata.

8 Click on the **File List** tab to set up new SSAP components.

The File List Tab displays files in the local directory to the left and files in the selected SSAP component to the right. On the bottom left is a directory listing and a method to move through the directory tree on the local machine. Delete and Reset buttons -- both active – are to the right.

9 Click on the **File Type** button to select the SSAP component to manipulate.

10 Choose one of the menu items.

11 Select a **file** (or files) from the left window to add to the component.

12 Click the **Add Arrow** button to add the files. They will appear in the right window because they are now part of that SSAP Component.

13 Click **Main** to get back to the Main tab.

14 On the **Main tab**, click **Submit** to send the new SSAP to Data Server. When finished, a message should pop up that says “SSAP Successfully inserted to the Data Server”. The SSAP has been updated at the Data Server.

Standards Checking of Science Software

Standards Checking Overview

The purpose of standards checking is to verify that the source files of the science software are compliant with the ESDIS Data Production Software Computing Facility (SCF) Standards and Guidelines document.

Checking FORTRAN 77 ESDIS Standards Compliance

The ESDIS Data Production Software Computing Facility (SCF) Standards and Guidelines document requires all FORTRAN 77 code to be compliant with the ANSI FORTRAN 77. The COTS used for this task is FORCHECK.

The following is a list of tools, and or assumptions:

Assumptions:

1. The FORTRAN 77 science software source code is available, accessible, and has read permissions for the user.
2. SSIT Manager is available for use.
3. FORCHECK is available only on the AIT Suns.

To check for ESDIS standards compliance in FORTRAN 77 code, execute the procedure steps that follow:

1 If not already on an AIT Sun, log into one from your machine.

Once logged onto proper Sun, remember to set the **DISPLAY** environmental variable to point to your X Window screen.

2 If required, at the UNIX prompt on the AIT Sun, type **cleartool setview *ViewName*** and then press the **Enter** key.

The ***ViewName*** is the name of a view allowing the FORTRAN 77 source files to be accessible.

This step is only necessary if any of the FORTRAN 77 source files are in ClearCase (in the VOB under configuration management).

3 If your general environment setup doesn't include transparent access to the SSIT Manager GUI, then you need to set that up. One way to do it is as follows:

Set up an alias, manually or from shell script, to set up preliminary environment. At UNIX prompt, type **alias do_buildrc "source /usr/ecs/TS1/CUSTOM/bin/DPS/.buildrc**"

Set up an alias, manually or through shell script, to invoke SSIT Manager. At UNIX prompt, type alias do_ssit_man "/usr/ecs/TS1/CUSTOM/bin/DPS/EcDpAtMgr ConfigFile /usr/ecs/TS1/CUSTOM/cfg/EcDpAtMG.CFG ecs_mode TS1& "

4 Set up the preliminary environment (do_buildrc). This only needs to be done once per session. Then, run SSIT Manager (do_ssit_man).

Type do_buildrc

Type do_ssit_man

5 Once the SSIT Manager comes up, the following steps need to be taken to invoke **FORCHECK**

From the top menu bar, select Tools.

From the Tools menu, select Standards Checkers.

From the Standards Checkers menu, select **FORCHECK.**

See Figure 11. for a screen snapshot of this step.

6 A separate FORCHECK window will now open.

The user will be prompted for input. The first prompt will be*global option(s) and list file?*

The second prompt will belocal option(s) and file(s)?

The second prompt will be repeated until there is a blank line and carriage return.

In order to understand what the proper responses should be, the user is encouraged to find hardcopy documentation for FORCHECK or to use the UNIX man facility and type ***man forchk***
.

7 At the UNIX prompt on the AIT Sun, type **vi *FORCHECKoutput*** and then press the **Enter** key.

The ***FORCHECKoutput*** is the file name for the output file produced in step 6.

The ***FORCHECKoutput*** file will contain any warnings, errors, and other messages from FORCHECK. A summary will be at the bottom of the file.

Any text editor may be used for this procedure step.

8 At the UNIX prompt on the AIT Sun, type **vi *ListFile*** and then press the **Enter** key.

The ***ListFile*** is the file name for the list file specified at the FORCHECK prompt.

The ***ListFile*** file will contain FORCHECK messages similar to the ***FORCHECKoutput*** file embedded in the source code listing.

Any text editor may be used for this procedure step.

Checking for ESDIS Standards Compliance in Fortran 90

This procedure describes how to use the Fortran 90 compiler flags on the SPR SGI machines to check science software written in Fortran 90 for ESDIS standards compliance.

Unlike with FORTRAN 77, no COTS tool is used to check Fortran 90 science software. Instead, this procedure describes how to use the compiler to perform the checking (ESDIS standards for Fortran 90 are ANSI). Since the Fortran 90 compiler is used, the checking for standards compliance can be naturally tied in with building the science software (since this procedure will produce object files suitable for linking). However, in this procedure, the building of the software (compiling *and* linking) is deferred to a later procedure.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The Fortran 90 science software source code is available, accessible, and has read permissions for the user.
2. Required Status Message Facility (SMF) files have been compiled.
3. The C shell (or a derivative) is the current command shell.
4. The Fortran 90 compiler is available on the SPR SGI.

To check for ESDIS standards compliance in Fortran 90 code, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then **telnet** to the **SPR SGI.**

Alternatively, in any currently available xterm window, spawn a new session: type xterm & and then press the **Enter** key. Then **telnet** to the **SPR SGI.**

2 At the UNIX prompt on the SPR SGI, set up the proper environment for the compiler to be used by typing **source *ToolkitPathname* /bin/*sgiXX*/pgs-dev-env.csh .**

ToolkitPathname is the home directory of the desired SDP Toolkit version .

The directory ***sgiXX*** should be replaced with **sgi32** or **sgi64** as appropriate for the specific compiler desired.

3 On workstation **x0spg##,** at the UNIX prompt in a terminal window, type **source /data3/ecs/TS1/CUSTOM/toolkit/bin/sgi64/pgs-dev-env.csh** . This will set up the various environment parameters, such as PGSHOME, to enable the 64 bit version of the FORTRAN 90 compiler to be run.

NOTE: The x in the workstation name will be a letter designating your site:

g = GSFC, m = SMC, l = LaRC, e = EDC, n = NSIDC, o = ORNL, a = ASF, j = JPL; p=PVC, the ## will be an identifying two-digit number (e.g.,g0spg03 indicates a science processor subsystem workstation at GSFC).

4 Prior to the rlogin, enter **setenv DISPLAY <local_workstation IP address>:0.0**. The <ipaddress> is the ip address of x0spg01. For example, on the PVC platform "p0spg01", type **source /data3/ecs/TS1/CUSTOM/toolkit/bin/sgi64/pgs-dev-env.csh** . This will set up the various environment parameters, such as PGSHOME, to enable the 64 bit version of the FORTRAN 90 compiler to be run.

5 If required, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName*** and then press the **Enter** key.

The ***ViewName*** is the name of a view allowing the Fortran 90 source files to be accessible.

This step is only necessary if any of the Fortran 90 source files are in ClearCase (in the VOB under configuration management).

6 At the UNIX prompt on the SPR SGI, type **cd *SrcPathname*** and then press the **Enter** key.

The ***SrcPathname*** is the full path name to the location of the Fortran 90 source files to be checked.

The ***SrcPathname*** will be in the ClearCase VOB is the Fortran 90 source files are checked into ClearCase.

7 At the UNIX prompt on the SPR SGI, type **f90 -c -ansi *[*-I$PGSINC*] [*-I$HDFINC*] [[*-IOtherIncFiles*…] SourceFiles* >& *ReportFile*** and then press the **Enter** key.

The terms in square brackets **(*[ ]*)** are used to optionally specify locations of include and module (.mod) files. The **$PGSINC** already contains the SDP Toolkit include directory and **$HDFINC** already contains the HDF include directory. The OtherIncFiles represents one or more additional include or module directories.

The ***SourceFiles*** is a list (space delimited) of Fortran 90 source files or a wildcard template (*e.g.* *.f90).

The **>&** is a C shell construct that causes standard error (where the output from the Fortran 90 compiler normally emerges) to be redirected to a file.

The ***ReportFile*** is the file name under which to save the results of the compile process.

The -c flag causes only compilation (no linking).

The **-ansi flag** enables ANSI checking.

Apply the terms in square brackets only as necessary. Do not include the brackets in the actual command. See example below.

Do not use the **-I option** for include or module files that are in the standard directories or in the current directory.

The makefile for the science software may contain the names of additional include files needed by the software.

For example, type f90 -c –ansi-I$PGSINC -I$HDFINC -I/ecs/modis/pge5/include/ *.f90 >& pge10.report and then press the Enter key.

8 At the UNIX prompt on the **SPR SGI**, type **vi *ReportFile*** and then press the **Enter** key.

The ***ReportFile*** is the file name for the compilation results as produced in step 5.

Any text editor may be used for this procedure step.

Checking for ESDIS Standards Compliance in C

This procedure describes how to use the C compiler flags on the SPR SGI machines to check science software written in C for ESDIS standards compliance.

Unlike with FORTRAN 77, no COTS tool is used to check C science software. Instead, this procedure describes how to use the compiler to perform the checking (ESDIS standards for C are essentially ANSI). Since the C compiler is used, the checking for standards compliance can be naturally tied in with building the science software (since this procedure will produce object files suitable for linking). However, in this procedure, the building of the software (compiling *and* linking) is deferred to a later procedure.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C science software source code is available, accessible, and has read permissions for the user.
2. Required Status Message Facility (SMF) files have been compiled.
3. The C shell (or a derivative) is the current command shell.
    3. The C compiler is available on the SPR SGI.

To check for ESDIS standards compliance in C code, execute the procedure steps that follow:

---

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then **telnet** to the **SPR SGI.**

Alternatively, in any currently available xterm window, spawn a new session: type xterm & and then press the **Enter** key. Then **telnet** to the **SPR SGI.**

2 At the UNIX prompt on the SPR SGI, type **setenv PGSHOME *ToolkitPathname*** and then press the Enter key. Then type, source **$PGSHOME/bin/*sgiX*/pgs-dev-env.csh** and then press the **Enter** key.

The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version.

The ***sgiX*** refers to the appropriate processor. For example, type **source $PGSHOME/bin/sgi/pgs-dev-env.csh** and then press the **Enter** key.

3 If required, at the UNIX prompt on the **SPR SGI,** type **cleartool setview *ViewName*** and then press the **Enter** key.

The ***ViewName*** is the name of a view allowing the C source files to be accessible.

This step is only necessary if any of the C source files are in ClearCase (in the VOB under configuration management).

4 At the UNIX prompt on the **SPR SGI**, type **cd *SrcPathname*** and then press the **Enter** key.

The ***SrcPathname*** is the full path name to the location of the C source files to be checked.

The ***SrcPathname*** will be in the ClearCase VOB is the C source files are checked into ClearCase.

5 At the UNIX prompt on the SPR SGI, type **cc -c -ansiposix *[*-I$PGSINC*] [*-I$HDFINC*] [[*-IOtherIncFiles*…] SourceFiles* >& *ReportFile*** and then press the **Enter** key.

The terms in square brackets **(*[ ]*)** are used to optionally specify locations of include and module (.mod) files. The **$PGSINC** already contains the SDP Toolkit include directory and **$HDFINC** already contains the HDF include directory. The OtherIncFiles represents one or more additional include directories.

The ***SourceFiles*** is a list (space delimited) of C source files or a wildcard template **(*e.g.* *.c).**

The **>&** is a C shell construct that causes standard error (where the output from the C compiler normally emerges) to be redirected to a file.

The ***ReportFile*** is the file name under which to save the results of the compile process.

The **-c** flag causes only compilation (no linking).

The **-ansiposix flag** enables ANSI and POSIX checking.

Apply the terms in square brackets only as necessary. Do not include the brackets in the actual command. See example below.

Do not use the **-I option** for include files that are in the standard directories (*e.g.* /usr/include) or in the current directory.

The makefile for the science software may contain the names of additional include files needed by the software.

For example, type cc -c -I$PGSINC -I$HDFINC -I/ecs/modis/pge5/include/ *.c >& pge10.report and then press the Enter key.

6 At the UNIX prompt on the SPR SGI, type **vi *ReportFile*** and then press the **Enter** key.

The ***ReportFile*** is the file name for the compilation results as produced in step 5.

Any text editor may be used for this procedure step.

Checking for ESDIS Standards Compliance in Ada

This procedures describes how to use Ada compilers on the SPR SGI machines to check science software written in Ada for ESDIS standards compliance.

Unlike with FORTRAN 77, Fortran 90, or C, Ada compilers are subjected to a validation process by the DoD Ada Committee. Thus, any code that compiles successfully by a validated compiler is, by definition, fully ANSI compliant. Since the Ada compiler is used, the checking for standards compliance can be naturally tied in with building the science software (since this procedure will produce object files suitable for linking). However, in this procedure, the building of the software (compiling *and* linking) is deferred to a later procedure.

Checking for ESDIS Standards Compliance in Ada: Verdix COTS

This procedure describes compiling Ada software using the COTS Verdix Ada Development System (VADS) which provides a complete environment for building (and developing) Ada software. See the *gcc* compiler in compiling Ada code.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The Ada science software source code is available, accessible, and has read permissions for the user.
2. The C shell (or a derivative) is the current command shell.

The Ada compiler is available on the **SPR SGI**.

To check for ESDIS standards compliance in Ada code, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then **telnet** to the **SPR SGI.**

Alternatively, in any currently available xterm window, spawn a new session: type **xterm &** and then press the **Enter** key. Then **telnet** to the **SPR SGI.**

2 If required, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName*** and then press the **Enter** key.

The ***ViewName*** is the name of a view allowing the Ada source files to be accessible.

This step is only necessary if any of the Ada source files are in ClearCase (in the VOB under configuration management).

3 At the UNIX prompt on the SPR SGI, type **setenv SGI_ABI -32** and then press the **Enter** key.

This command sets the environment variable SGI_ABI for 32-bit mode compilation.

4 At the UNIX prompt on the SPR SGI, type **cd *SrcPathname*** and then press the **Enter** key.

The ***SrcPathname*** is the full path name to the location of the Ada source files to be checked.

The ***SrcPathname*** will be in the ClearCase VOB if the Ada source files are checked into ClearCase.

5 At the UNIX prompt on the SPR SGI, type **a.mklib** and then press the **Enter** key.

This command creates a VADS library directory. All Ada compilation must occur in a VADS Ada library.

6 At the UNIX prompt on the SPR SGI, type **a.make -v -f *SourceFiles* >& *ReportFile*** and then press the **Enter** key.

The ***SourceFiles*** is a list (space delimited) of Ada source files or a wildcard template **(*e.g.* *.ada).**

The **>&** is a C shell construct that causes standard error (where the output from the Ada compiler normally emerges) to be redirected to a file.

The ***ReportFile*** is the file name under which to save the results of the compile process.

The **-v flag** enables verbose output.

The **-f flag** indicates that what immediately follows are the source files. The order of the flags is therefore important.

7 At the UNIX prompt on the AIT Sun, type **vi *ReportFile*** and then press the **Enter** key.

The ***ReportFile*** is the file name for the compilation results as produced in step 6.

Any text editor may be used for this procedure step.

Checking for ESDIS Standards Compliance in Ada: GNU *gcc* Compiler

This procedure describes compiling Ada software using the **GNU C compiler, *gcc*.**

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The Ada science software source code is available, accessible, and has read permissions for the user.
2. The C shell (or a derivative) is the current command shell.

The GNU *gcc* compiler is available on the SPR SGI.

To check for ESDIS standards compliance in Ada code, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then **telnet** to the **SPR SGI.**

Alternatively, in any currently available xterm window, spawn a new session: type xterm & and then press the **Enter** key. **Then telnet** to the **SPR SGI.**

It is recommended that this procedure begin within a new command shell on the SPR SGI.

2 If required, at the UNIX prompt on the SPR SGI, type **cleartool setview *ViewName*** and then press the **Enter** key.

The ***ViewName*** is the name of a view allowing the Ada source files to be accessible.

This step is only necessary if any of the Ada source files are in ClearCase (in the VOB under configuration management).

3 At the UNIX prompt on the SPR SGI, type **setenv SGI_ABI -32** and then press the **Enter** key.

This command sets the environment variable SGI_ABI for 32-bit mode compilation.

4 At the UNIX prompt on the SPR SGI, type **cd *SrcPathname*** and then press the **Enter** key.

The ***SrcPathname*** is the full path name to the location of the Ada source files to be checked.

The ***SrcPathname*** will be in the ClearCase VOB is the Ada source files are checked into ClearCase.

5 At the UNIX prompt on the SPR SGI, type **gcc -c -gnat83 *SourceFiles* >& *ReportFile*** and then press the **Enter** key.

The ***SourceFiles*** is a list (space delimited) of Ada source files or a wildcard template **(*e.g.* *.ada).**

The **>&** is a C shell construct that causes standard error (where the output from the *gcc* compiler normally emerges) to be redirected to a file.

The ***ReportFile*** is the file name under which to save the results of the compile process.

The **-c flag** causes only compilation (no linking).

The **-gnat83** enables compilation of Ada using the 1983 Ada Standard. Note that without this flag, the compiler would assume the 1995 Ada proposed Standard.

6 At the UNIX prompt on the SPR SGI, type **vi *ReportFile*** and then press the **Enter** key.

The ***ReportFile*** is the file name for the compilation results as produced in step 5.

Any text editor may be used for this procedure step

Prohibited Function Checker

The use of certain functions in the PGE is prohibited. The Prohibited Function Checker (Figure 11) is used to check C, FORTRAN 77, FORTRAN 90, and Ada language source files for the occurrence of functions that are prohibited in the ECS DAAC production environment.

Checking for Prohibited Functions: Command-Line Version

This procedure describes using the command-line version of the Prohibited Function Checker to check science software for the prohibited functions.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The source files to be checked are available, accessible, and have read permissions for the operator.
2. Source files to be checked are Ada, C, FORTRAN 77, Fortran 90, C shell, Korn shell, Bourne shell, or Perl and have recognizable file name extensions.

To check for prohibited functions in delivered source files, execute the procedure steps that follow:

1 If required, at the UNIX prompt on an AIT Sun, type **cleartool setview *ViewName*** and then press the **Enter** key.

The ***ViewName*** is the name of a view allowing the source files to be accessible.

This step is only necessary if any of the source files are in ClearCase (in the VOB under configuration management).

2 At the UNIX prompt on the AIT Sun, type **cd *SrcPathname*** and then press the **Enter** key.

The ***SrcPathname*** is the full path name to the location of the source files to be checked.

The ***SrcPathname*** will be in the ClearCase VOB if the source files are checked into ClearCase.

The ***SrcPathname*** can contain other directories that contain source files and/or more directories. The Prohibited Function Checker will search out all source files in subdirectories recursively.

3 At the UNIX prompt on the AIT Sun, type
/data3/ecs/TS1/CUSTOM/bin/DPS/EcDpAtMgrBadFunc ConfigFile

/data3/ecs/TS1/CUSTOM/cfg/EcDpAtBA.CFG ***FilesOrDirectories*** **>** ***ResultsFile*** and then press the **Enter** key.

The ***FilesOrDirectories*** is a list of source file names or directory names of directories containing source files.

The ***ResultsFile*** is the file name for the results that are output.

For example, type **/data3/ecs/TS1/CUSTOM/bin/DPS/EcDpAtMgrBadFunc ConfigFile /data3/ecs/TS1/CUSTOM/cfg/EcDpAtBA.CFG main.c utils/** > myOutput and then press the **Enter** key. Here, main.c is a source file and utils/ is a directory that contains other source files.

4 At the UNIX prompt on the AIT Sun, type **vi** ***ResultsFile*** and then press the **Enter** key.

The ***ResultsFile*** is the file name for the output results as produced in step 3.

Any text editor may be used for this procedure step.

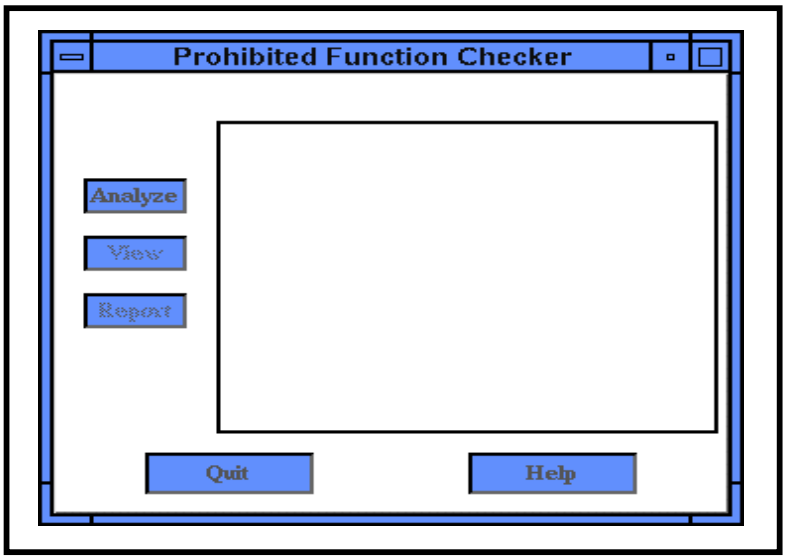


Figure 11. Prohibited Function Checker

Prohibited Function Checker GUI

1 From the SSIT Manager, select **Tools → Standards Checkers → Prohibited Function Checker** from the menu.

The Prohibited Function Checker GUI will be displayed.

2 In the Prohibited Function Checker GUI, click on the **Analyze** button.

The File Selector GUI will be displayed.

3 Within the **Directories** subwindow, double click on the desired directory.

Repeat this step until the directory with the source files to be checked are displayed in the Files subwindow.

4 Within the **Files** subwindow, click on the source files to be checked. Each file clicked on will be highlighted.

To choose groups of contiguous files, hold down the left mouse button and drag the mouse.

To choose non-contiguous files, hold down the Control key while clicking on file names.

5 In the File Selector GUI, click on the **OK** button.

The File Selector GUI will disappear.

The files selected in step 5 will be displayed in the Prohibited Function Checker GUI window as they are being checked.

6 In the Prohibited Function Checker GUI, click on the **Report** button.

The Report GUI will be displayed.

For each file, a list of prohibited functions found will be displayed.

7 Optionally, click on the **Print** button or the **Save** button.

Choose Save to save the results to a file; choose Print to have the results printed on the default printer.

Choosing Save will bring up a GUI labeled Save To File. Specify the directory and file name in which to save the results file.

8 Optionally, in the Prohibited Function Checker GUI, highlight one of the source files listed. Then click on **View**.

The Source Code GUI will be displayed.

Occurrences of prohibited functions found in that source file will be highlighted.

Click on the Next button to bring into the window successive occurrences of prohibited functions (the Next button does not bring in the next source file).

Click on the Done button to close the Source Code GUI. Other source files may be examined similarly, one at a time.

9 In the Prohibited Function Checker GUI, click on the **Quit** button.

The Prohibited Function Checker GUI will disappear.

This ends the session.

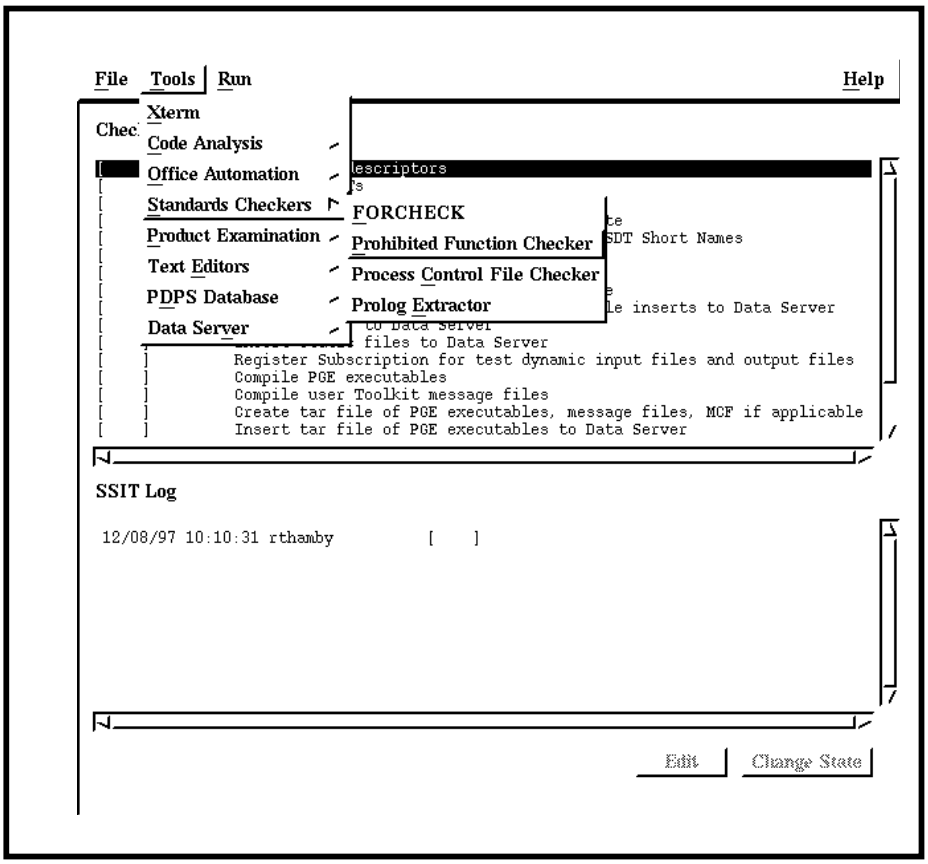


Figure 12. Invoking the Prohibited Function Checker

Checking Process Control Files

The next task to accomplish is to check that the PCFs are syntactically correct and contains all the necessary information for PGE's to run within the ECS DAAC production environment. Only one PCF can be associated with a PGE. The following procedure describes how to check PCFs for valid syntax and format.

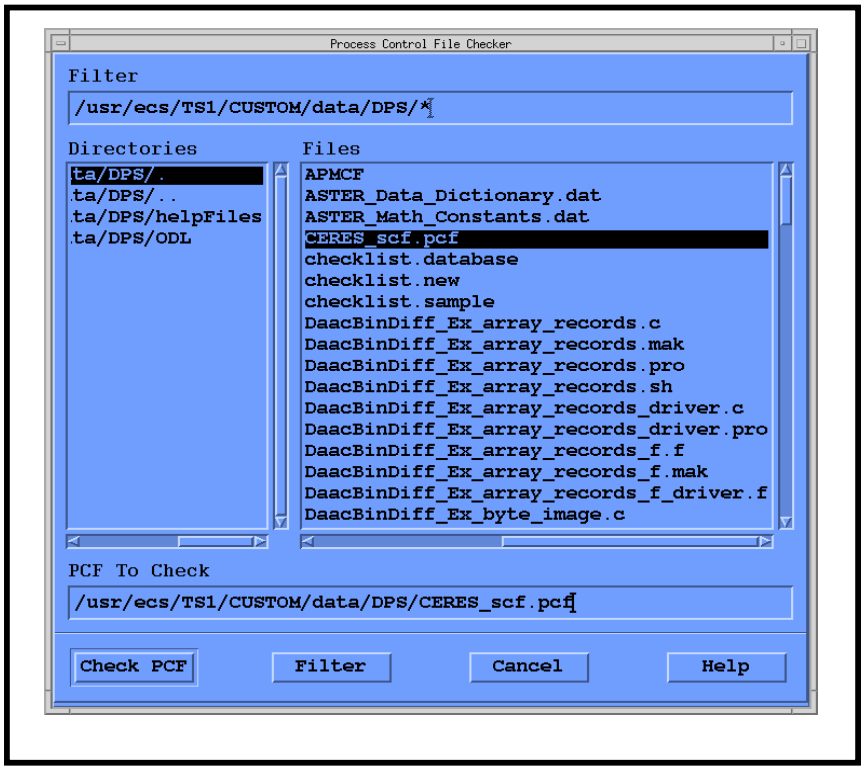


Figure 13. Process Control File Checker GUI

The following is a list of tools, and or assumptions:

1. The SSIT Manager is running.
2. The Process Control File(s) are available, accessible, and have read permissions.
3. If the source code files to be checked are in a VOB in ClearCase, a view has been set before the SSIT Manager was started.

Checking Process Control Files GUI

1 From the SSIT Manager, select **Tools → Standards Checkers → Process Control File Checker** from the menu, see figure 13.

The Process Control File Checker GUI will be displayed.

2 In the **Directories** subwindow, double click on the desired directory.

Repeat this step until the directory with the PCF(s) to be checked are displayed in the Files window.

Use the Filter subwindow to limit which files are displayed.

3 Within the **Files** subwindow, click on the PCF to be checked**.**

The file clicked on will be highlighted.

Only one PCF can be checked at a time.

4 Click on the **Check PCF** button.

A GUI labeled PCF Checker Results will be displayed.

Results will be displayed in this window.

5 Optionally, click on the **Save** button or on the **Print** button.

Choose Save to save the results to a file; choose Print to have the results printed on the default printer.

Choosing Save will bring up a GUI labeled Save To File. Specify the directory and file name in which to save the results file.

Choosing Print and then clicking on the **OK button** will send the results to the default printer.

6 Click on the **Check Another** button or on the **Quit** button.

Choosing Check Another allows another PCF to be checked. Repeat steps 2 through 5.

Choosing Quit causes the Process Control File Checker GUI to disappear and ends the session.

Checking Process Control Files: Command-Line Version

This procedure describes using the command-line version of the Process Control File Checker to check process control files delivered with the science software.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The PCF files to be checked are available, accessible, and have read permissions for the operator.
2. You will need the command **pccheck.sh**. One way to see if this is available is to type **which pccheck.sh** and then press the **Enter** key. If a path is displayed, then the directory is in your path. On the PVC Sun platform **p0ais01,** the pathname for the command is **/ecs/formal/TOOLKIT/bin/sun5/pccheck.sh.** In this case, you will have to set a ClearCase view to access that area.

To check Process Control Files, execute the procedure steps that follow:

1 If required, at the UNIX prompt on an AIT Sun, type **cleartool setview *ViewName*** and then press the Enter key.

The ***ViewName*** is the name of a view allowing the Process Control File(s) to be accessible.

This step is only necessary if any of the Process Control Files are in ClearCase (in the VOB under configuration management).

2 At the UNIX prompt on AIT Sun, type **cd *PCFpathname*** and then press the **Enter** key.

The ***PCFpathname*** is the full path name to the location of the Process Control File(s) to be checked.

The ***PCFpathname*** will be in the ClearCase VOB if the Process Control Files are checked into ClearCase.

3 At the UNIX prompt on an AIT Sun, type **/ecs/formal/TOOLKIT/bin/sun5/pccheck.sh -i *PCFfilename* > *ResultsFile*** and then press the **Enter** key.

The ***PCFfilename*** is the full path name (directory and file name) to the Process Control File to check.

The ***ResultsFile*** is the file name for the results that are output.

The PCF Checker is also available on the SPR SGI machines. The easiest way to access it is to set a SDP Toolkit environment (any will do for purposes here) and type **$PGSBIN/pccheck.sh -i *PCFfilename* > *ResultsFile*** and then press the **Enter** key.

4 At the UNIX prompt on the SPR SGI, **p0spg01**, type **vi *ResultsFile*** and then press the **Enter** key.

The ***ResultsFile*** is the file name for the output results as produced in step 4.

Any text editor may be used for this procedure step.

Extracting Prologs

The Project standards and guidelines are contained in the document *Data Production Software and Science Computing Facility (SCF) Standards and Guidelines, Revision A, October 1996* (423-16-01). This ESDIS document mandates that science software delivered to the DAACs to be integrated into the ECS contain prologs in the source files. Prologs are internal documentation containing information about the software. The details are specified in the ESDIS document. Prologs must be at the top of every function, subroutine, procedure, or program module.

This procedure describes using the Prolog Extractor to extract prologs into a file. Note that the prolog extractor only extract the prologs it finds. It does not check the contents of prologs.

The following is a list of tools, and or assumptions:

The SSIT Manager is running.

Prologs are assumed to be delimited by particular delimiters depending on the language type. Delimiters are listed in the table 3 below:

Table 3. Prolog Delimiters

Language	Type	Delimiter
FORTRAN 77	source	!F77
Fortran 90	source	!F90
C	source	!C
Ada	source	!Ada
FORTRAN 77	include	!F77-INC
Fortran 90	include	!F90-INC
C	include	!C-INC
Any Language	any	!PROLOG
All Languages	The end delimiter is always !END	

The Prolog Extractor recognizes the language type of the file by its file name extension. The table below lists assumed file name extensions:

Table 4. File Name Extensions

File Type	File Name Extensions
FORTRAN 77	f, f77, ftn, for, F, F77, FTN, FOR
Fortran 90	f90, F90, f, F
FORTRAN 77/Fortran 90 include	inc, INC
C	c
C/C++ header	h
Ada	a, ada

Extracting Prologs

The Prolog Extractor can be started from the UNIX prompt. To do this, at the UNIX prompt on the AIT Sun, type **/data3/ecs/TS1/CUSTOM/bin/DPS/EcDpAtMgrPrologs**, press **Enter**

or

1 From the SSIT Manager, select the **Tools → Standards Checkers → Prolog Extractor** from the menu.

An xterm will be displayed on the AIT Sun.

Select the default ConfigFile. The output goes to a file called Prologs in the directory from which the SSIT Manager was started.

The Prologs file can be viewed by changing directories to the SSIT Manager directory and invoking a text editor. The file may also be sent to a printer.

2 At the **Files(S)? (-h help)** prompt, type in the file names and/or directory names containing the files.

Separate items with spaces.

The contents of the directory will be search recursively for files with valid file name extensions.

Use **./** to indicate current directory.

The time needed for the Prolog Extractor could be very long for large numbers of files and directories.

When extraction is complete, the message Output written to file: **./prologs** will be displayed.

3 At the program prompt **Hit Enter for another, ‘q <Enter>‘ to quit:** , press **Enter** to repeat process with another set of source files or type **q** and press **Enter** to quit.

The xterm will disappear.

4 At a UNIX prompt on the AIT Sun, type **vi prologs**, then press the **Enter** key.

The extracted prologs file, named **prologs**, will be brought into the editor.

The default location of the prologs file is the directory from which the SSIT Manager was invoked.

5 Once the extracted prologs file has been examined, exit the editor.

```
SOURCE CODE PROLOG EXTRACTOR

Configuration filename? (enter for default: ../../cfg/EcDpAtPrologs.CFG)

ECS mode? (enter for default: OPS)
TS1
File(s)?  (enter -h for help)
/home/dps/ssit/*.c
Warning: Could not open message catalog "oodce.cat"
[Warning:
Invalid Resource Catalog directory path or no catalog installed
Applications can run with or without Resource Catalog
FYI : Values of ECS_HOME env variable and RC Directory path:/usr/ecs/ecsmode/CU
STOM/data/DPS/ResourceCatalogs
]

EcDpAtPrologs: Process Framework: ConfigFile ../../cfg/EcDpAtPrologs.CFG  ecs_m
ode ecsmode

Output written to file: /usr/ecs//TS1/CUSTOM/logs/prologs.txt

Hit return for another, 'q <return>' to quit:
```

Figure 14. Prolog Extractor Sample Run.

This page intentionally left blank.

Compiling and Linking Science Software

Science software is developed at independent Science Computing Facilities (SCFs) using the SDP Toolkit. The SDP Toolkit allows science software to be developed for ECS at independent SCFs. Once delivered to the DAACs for SSI&T, science software needs to be compiled and linked to one of the SDP Toolkit versions resident at the DAAC. The (PCFs) Process Control Files provide the interface between the science software and the production system in the ECS. Since the process control files delivered to the DAACs for SSI&T were created and used at the SCFs, the path names in the PCF will need to be checked and revised to work at the DAACs.

To save time for the SSI&T Training Lesson, the compile and link with the SCF Version of the Toolkit will be omitted. The procedures are included in the student guide for future reference.

The next step is to set up a DAAC version SDP Toolkit environment, compile the PGE, and link to the DAAC Toolkit. This procedure will be performed at the SSI&T Training. The procedure steps for the two processes are the same except for the set up for the Toolkit environment and link with the corresponding Toolkit library.

Updating the Process Control Files (PCFs)

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. A PCF for the PGE has been delivered and is available, accessible, and has read permissions.

To update the PCF, execute the procedure steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then **telnet** to the **SGI.**

Alternatively, in any currently available xterm window, spawn a new session: type **xterm &** and then press the **Enter** key. Then **telnet** to the **SGI**.

2 If required, at the UNIX prompt on the SGI, type **cleartool setview *ViewName*** and then press the **Enter** key.

The ***ViewName*** is the name of a view allowing the PCF to be accessible.

This step is only necessary if the PCF is in ClearCase (in the VOB under configuration management).

3 At the UNIX prompt on the Sun or on the SGI, type **cd *PCFpathname*** and then press the **Enter** key.

The ***PCFpathname*** is the full path name to the location of the PCF. This location will be in the ClearCase VOB if the PCF is under configuration management.

4 At the UNIX prompt on the Sun or on the SGI, type **cleartool checkout -nc *PCFfilename*** and then press the **Enter** key.

The ***PCFfilename*** is the file name of the **PCF** that is to be checked out (and later modified). The **-nc flag** means **"no comment"**; it suppresses ClearCase from prompting for a comment to be associated with the check out step.

5 Run the Process Control File Checker on the delivered **PCF**.

This will verify that the delivered **PCF** is correct before editing.

6 At a UNIX prompt on the Sun, type **vi *PCFfilename*** and then press the **Enter** key.

The ***PCFfilename*** is the file name of the PCF to update.

Any text editor may be used such as ***emacs*.** For example, **emacs AST02.pcf** and then press the **Enter** key.

7 In the file, make changes to the default directories specified in each section of the **PCF**. All path names specified in the **PCF** must exist on the **SGI.**

Each section begins with a line consisting of a **?** in the first column followed by a label:

```
? PRODUCT INPUT FILES
? PRODUCT OUTPUT FILES
? SUPPORT INPUT FILES
? SUPPORT OUTPUT FILES
? INTERMEDIATE INPUT
? INTERMEDIATE OUTPUT
? TEMPORARY I/O
```

Each of the above section heading lines will then be followed (not necessarily immediately; there may be comment lines) by a line that begins with a **!** in the first column. These lines specify the default path names for each section.

If the line reads:

```
! ~/runtime
```

leave it unchanged. The **tilde (~)** is a symbol that represents **$PGSHOME**.

If another path name is listed instead, it will probably need to be changed to a path name that exists at the DAAC on the SGI. When specifying a path name, use an absolute path name, not a relative path name.

8 In the file, look for science software specific entries in each section and make changes to the path names (field 3) as necessary. All path names specified in the PCF must exist on the SGI.

The science software specific entries will have logical IDs (first field) ***outside*** of the range 10,000 to 10,999.

Where necessary, replace the path names in the third field of each entry with the path names appropriate to the DAAC environment.

Do not alter file entries that are used by the SDP Toolkit itself. These have logical IDs ***in*** the range 10,000 to 10,999.

For example, if the following entry was found in the **PCF**:

100|A.granule|/MODIS/run/input||||1

change /MODIS/run/input to the appropriate path name in the DAAC where the file A.granule is stored.

When specifying a path name, use **an absolute path name**, not a relative path name.

Do not include the file name with the path name. The file name belongs in field 2 by itself.

9 In the file, verify that the SUPPORT OUTPUT FILES section contains an entry to the shared memory pointer file.

Look for the entry: 10111|ShmMem|~/runtime||||1

The third field may be blank; this will work too.

If this entry is not within this section, add it.

Once changes have been made to the **PCF**, save the changes and exit the editor.

The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is :**wq** and then press the **Enter** key.

For other editors, refer to that editor's documentation.

11 Again, run the Process Control File Checker on the PCF.

12 If the PCF had been checked out of ClearCase, at the UNIX prompt on the SGI, type **cleartool checkin -nc *PCFfilename*** and then press the **Enter** key.

The ***PCFfilename*** is the file name of the modified **PCF**. The **-nc flag** means **"no comment"**; it suppresses ClearCase from prompting for a comment to be associated with the check in step.

Setting up a SDP Toolkit Environment

The purpose of the SDP Toolkit is to allow science software to be developed for ECS at independent SCFs and to provide:

An interface to the ECS system, including PDPS and CSMS and information management.

A method for Science software to be portable to different platforms at the DAAC.

A method to reduces redundant coding at the SCF.

Value added functionality for science software development.

The SDP Toolkit is divided into two groups of tools:

Mandatory Tools

Error and Status Message Facility (SMF) - provides general error handling, status log messaging, and interface to CSMS services.

Process Control Tools - provides the primary interface to the PDPS. Allows access to physical filenames and file attributes and retrieval of user defined parameters.

Generic Input/Output - provides the means to open and close support, temporary and intermediate duration files.

Memory Allocation Tools - simple wrappers on native C functions which track memory usage in the SDPS, and shared memory tools which enable the sharing of memory among executables within a PGE.

Optional Tools

Ancillary Data Access - provides access to NMC data and Digital Elevation (DEM) data.

Celestial Body Position - locates the sun, moon and the planets.

Coordinate System Conversion - coordinate conversions between celestial reference.

Constant and Unit Conversion - physical constants and unit conversions.

IMSL - mathematical and statistical support.

In the description of the Toolkit routines, descriptive information is presented in the following format:

TOOL TITLE

NAME: Procedure or routine name
SYNOPSIS: C: C language call

FORTRAN: FORTRAN77 or Fortran90 language call
DESCRIPTION: Cursory description of routine usage
INPUTS: List and description of data files and parameters input to the routine
OUTPUTS: List and description of data files and parameters output from the routine
ENTERS: List of returned parameters indicating success, failure, etc.
EXAMPLES: Example usage of routine
NOTES: Detailed information about usage and assumptions
REQUIREMENTS: Requirements from PGS Toolkit Specification, Oct. 93 which the routine satisfies

The science software delivered to the DAACs is expected to work with either, the SCF SDP Toolkit or the DAAC SDP Toolkit which are both installed each DAAC. During the pre-SSI&T initial testing, the SCF Toolkit should be used.

There are several versions of the SCF/DAAC SDP Toolkit installed on the SGI Power Challenges at the DAACs for the Release 6A system. The toolkit versions at the DAACs differ according to:

Object Type - The operating system on the SGI Power Challenges on Release 6A is IRIX 6.5, a 64-bit operating system. To be backward compatible, the SGI operating system will allow new 64-bit and 32-bit objects to be built as well as the older 32-bit machines. Each of these object types are designated by placing a cc flag on the command line to enable a particular mode with the SGI C compiler.

New 64-bit: cc flag = -64

New 32-bit: cc flag = -n32

Old 32-bit: cc flag = -32 (SCF's only)

Library Type - The SDP Toolkit uses different libraries depending upon whether FORTRAN 77 or FORTRAN 90 source code is being linked. If C source code is to be linked, then either language version of the library will work.

Note - Each Toolkit library comes with a debug version, for example:
sgi32_daac_cpp/
sgi32_daac_cpp_debug/

The following Table 5, summarizes the available SDP Toolkits used by the SGI science processors.

Table 5. SDP Toolkits used by the SGI science processors (1 of 2)

SDP Version	Language Type	Library Object Type	$PGSBIN
SCF	C++ or C	Old 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi_scf_cpp
SCF	FORTRAN 77 or C	Old 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi_scf_f77
SCF	Fortran 90 or C	Old 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi_scf_f90
SCF	Thread	Old 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi_scf_r
SCF	C++ or C	New 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi32_scf_cpp
SCF	FORTRAN 77 or C	New 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi32_scf_f77
SCF	Fortran 90 or C	New 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi32_scf_f90
SCF	Thread	New 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi32__scf_r
SCF	C++ or C	64-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi64_scf_cpp
SCF	FORTRAN 77 or C	64-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi64_scf_f77
SCF	Fortran 90 or C	64-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi64_scf_f90
SCF	Thread	64-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi64_scf_r
DAAC	C++ or C	Old 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi_daac_cpp
DAAC	FORTRAN 77 or C	Old 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi_daac_f77
DAAC	Fortran 90 or C	Old 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi_daac_f90
DAAC	Thread	Old 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi_daac_r
DAAC	C++ or C	New 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi32_daac_cpp

Table 5. SDP Toolkits used by the SGI science processors (2 of 2)

SDP Version	Language Type	Library Object Type	$PGSBIN
DAAC	FORTRAN 77 or C	New 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi32_daac_f77
DAAC	Fortran 90 or C	New 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi32_daac_f90
DAAC	Thread	New 32-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi32_daac_r
DAAC	C++ or C	64-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi64_daac_cpp
DAAC	FORTRAN 77 or C	64-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi64_daac_f77
DAAC	Fortran 90 or C	64-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi64_daac_f90
DAAC	Thread	64-bit mode	$CUSTOM_HOME/TOOLKIT/toolkit/bin/sgi64_daac_r

$CUSTOM_HOME is an environment variable set to: **/usr/ecs/*MODE*/CUSTOM**
$PGSHOME is an environment variable set to: **usr/ecs/*MODE*/CUSTOM/TOOLKIT/toolkit**

The choice of which version of the SDP Toolkit to use depends upon two factors: The test being performed and the version required by the science software. For running a PGE in a simulated SCF environment (*i.e.* as if at the SCF), a SCF version of the Toolkit should be used (see Section 11). For running a PGE in the fully functional DAAC environment, the DAAC version should be used.

Among the DAAC versions, there are six choices. Most science software will likely require one of the 32-bit versions. If FORTRAN 77 code is being used (with or without C), then the FORTRAN 77 language version of the DAAC Toolkit must be used. Conversely, if Fortran 90 code is being used (again, with or without C), the Fortran 90 language version of the DAAC Toolkit must be used.

If both FORTRAN 77 and Fortran 90 are being used, the procedure becomes more complex. Under such circumstances, refer to document *333-CD-510-002, Release 5B SDP Toolkit Users Guide for the ECS Project, April 2000* .

In addition to the SDP Toolkit interface to the DAAC environment, there are some other interfaces (e.g. MAPI for MODIS codes) and libraries (e.g. OCEAN library for MODIS OCEAN

codes) designed to simplify the processes of building and running PGEs at DAACs. Follow the delivered documentation to build specific interfaces and libraries if necessary.

This procedure describes how to set up the appropriate SDP Toolkit environment. It involves two basic steps. First, set the SDP Toolkit home directory in the environment variable PGSHOME. The second step is to source (run) the set up script in the appropriate bin directory. This step results in a number of other environment variables getting set that will be needed.

Setting Up the SDP Toolkit Environment

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C shell (or a derivative) is the current command shell.

To set up a SDP Toolkit environment, execute the procedure steps that follow:

1 At the UNIX prompt on the **SGI,** type **setenv PGSHOME *ToolkitPathname*** and then press the **Enter** key.

The ***ToolkitPathname*** is the home directory of the particular SDP Toolkit version being used. Refer to Table 5. Note that the setting of **PGSHOME** shown in this table may differ in your local DAAC.

Korn shell users, type **PGSHOME=*ToolkitPathname***; export PGSHOME and then press the **Enter** key.

2 At the UNIX prompt on the SGI, type **source $PGSHOME/bin/*sgiX*/pgs-dev-env.csh** and then press the **Enter** key.

The ***sgi*** uses **sgi64** for 64-bit version of the Toolkit. Refer to the last column of Table 5, for path names to the file to source.

Korn shell users, type **$PGSHOME/bin/*sgiX*/pgs-dev-env.ksh**, press **Enter** (note the **"dot"** and then space at the beginning of this command).

3 This step is optional. Edit the file **$HOME/.cshrc** and add the line **alias *aliasname* 'setenv PGSHOME *ToolkitPathname*; source $PGSHOME/bin/*sgiX*/pgs-dev-env.csh; echo "*textmessage*" '**.

The ***aliasname*** is the name of the alias. For example, to set up an environment for the DAAC version of the Toolkit for FORTRAN 77 (or C), you might use DAACf77 as an ***aliasname*.**

The ***ToolkitPathname*** is the home directory of the particular **SDP Toolkit** version being used. Refer to Table 5. Note that the setting of **PGSHOME** shown in this table may differ in your local DAAC.

The ***sgi*** Toolkit uses **sgi64** for 64-bit version of the Toolkit.

The ***textmessage*** is a message that will be echoed to the screen signifying that a new Toolkit environment has been set up. It must be enclosed within double quotes **(“).** An example may be, “DAAC F77 Toolkit environment is now set.”

A complete example (it should be all on one line in the **.cshrc file**):

alias DAACf77 ‘setenv PGSHOME/$CUSTOM/TOOLKIT/toolkit/bin/sgi64_daac_f77/; source $PGSHOME/toolkit/bin/sgi64/pgs-dev-env.csh; echo “DAAC F77 Toolkit environment is now set” ‘

Other aliases for other versions of the Toolkit can be set up similarly.

An example of Compile procedures used to produce a PGE.exe:

Setup for PGE07:
/home/emcleod/MODIS/STORE/PGE07/MOD_PR10/source
rm MOD_PR10.exe
rm *.o
setenv PGSHOME /usr/ecs/OPS/CUSTOM/TOOLKIT/toolkit/bin/sgi64_daac_f77/
source $PGSHOME/toolkit/bin/sgi32/pgs-dev-env.csh
source
/home/emcleod/MODIS/STORE/PGE07/MOD_PR10/source/MODIS_setup.csh.pge07
alias
n32_f77
env
make -f MOD_PR10.mk &
ls -l *exe
setenv PGS_PC_INFO_FILE
/home/emcleod/MODIS/STORE/PGE07/MOD_PR10/source/MOD_PR10.pcf
ls
MOD_PR10.exe &
confirm execution when done by looking at file **: vi**
MOD_PR10_ClopyL1BmetaToSnow.c
see if job is running: ps -u emcleod "time updating for MOD_PR10"

```
p0spg01{emcleod}88: ps -u emcleod
   PID TTY     TIME CMD
   267 ?       3:13 biod
 25825 pts/11  0:01 csh
 21994 pts/10  0:01 csh
 23215 pts/16  0:00 csh
 26242 pts/10  0:07 MOD_PR10.
 26089 pts/11  0:01 xedit
 26318 pts/10  0:00 ps
p0spg01{emcleod}105: pwd
/tmp_mnt/home/emcleod/MODIS/STORE/PGE07/MOD_PR10/source
p0spg01{emcleod}106: ls
MODIS_setup.csh.pge07        MOD_PR10_CopyL1BmetaToSnow.c
```

```
MODIS_setup_OPS            MOD_PR10_CopyL1BmetaToSnow.o
MOD_PR10.exe               MOD_PR10_MakeMeta.c
MOD_PR10.h                 MOD_PR10_MakeMeta.o
MOD_PR10.mcf               MOD_PR10_Process_Cloud.c
MOD_PR10.mk                MOD_PR10_Process_Cloud.o
MOD_PR10.pcf               MOD_PR10_Process_GEO.c
MOD_PR10_AAmain.c          MOD_PR10_Process_GEO.o
MOD_PR10_AAmain.o          MOD_PR10_Process_L1B.c
MOD_PR10_Compute_Snow.c    MOD_PR10_Process_L1B.o
MOD_PR10_Compute_Snow.o    MOD_PR10_Process_SnowFile.c
MOD_PR10_CopyGEOmetaToSnow.c  MOD_PR10_Process_SnowFile.o
MOD_PR10_CopyGEOmetaToSnow.o  compile_smf.csh
p0spg01{emcleod}107:
```

Compiling Status Message Facility (SMF) Files

Status Message Facility (SMF) files are used by the SDP Toolkit to facilitate a status and error message handling mechanism for use in the science software and to provide a means to send log files, informational messages, and output data files to DAAC personnel or to remote users.

Science software making use of the SMF need particular header (include) files when being built and also need particular runtime message files when being run. Both the header and message files are produced by running a SMF "compiler" on a message text file. These message text files should be part of the science software delivery to the DAAC. They typically have a .t file name extension.

This procedure describes how to compile the SMF message text files to produce both the necessary include files and the necessary runtime message files.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C shell (or a derivative) is the current command shell.

To check compile status message facility (SMF) files, execute the procedure steps that follow:

1 From the SSIT Manager, click on the Tools menu, then choose **Xterm**. Then **telnet** to the **SGI**.

Alternatively, in any currently available xterm window, spawn a new session: **type xterm &** and then press the **Enter** key. Then **telnet** to the **SGI**.

It is recommended that this procedure begin within a new command shell on the **SGI.**

2 If required, at the UNIX prompt on the SGI, type **cleartool setview ViewName** and then press the **Enter** key.

The **ViewName** is the name of a view allowing the SMF files to be accessible.

This step is only necessary if any of the SMF files are in ClearCase (in the VOB under configuration management).

3 At the UNIX prompt on the SGI, type **setenv PGSHOME ToolkitPathname** and then press the **Ente**r key. Then type**, source $PGSHOME/bin/sgiX/pgs-dev-env.csh** and then press the **Enter** key.

The **ToolkitPathname** is the home directory of the desired SDP Toolkit version.

The sgiX refers to the appropriate processor. For example, type **source $PGSHOME/bin/sgi/pgs-dev-env.csh** and then press the **Enter** key.

4 At the UNIX prompt on the SGI, type **cd pathname** and then press the **Enter** key.

The pathname is the full path name to the directory containing the SMF text files.

The SMF text files will typically have **.t file** name extensions.

5 At the UNIX prompt on the SGI, type **smfcompile -f textfile.t -lang** and then press the **Enter** key.

The **-lang** is a flag that indicates for what language to compile. This flag can be one of **-c** to produce C header files, **-f77** to produce FORTRAN 77 include files, and **-ada** to produce Ada include files. The default is for **C** include files. For example, type **smfcompile -f77 PGS_MODIS_39123.t** and then press the **Enter** key.

The textfile is the file name of the SMF text file delivered with the science software.

The SMF text files will typically have .t file name extensions.

File names for SMF text files usually have the **"seed"** value used by the file as part of its file name (e.g. **PGS_MODIS_39123.t** where 39123 is the seed number).

Only one such SMF text file can be compiled at a time; wildcards cannot be used.

The SMF compiler may be run with the additional **flags -r and -i** as in, **smfcompile -f textfile.t -r -i.** The **-r** automatically places the runtime message file in the directory given by the environment variable **PGSMSG**. The **-i** automatically places the include file in the directory given by the environment variable **PGSINC.** For example, type **smfcompile -ada -r -i -f PGS_MODIS_39123.t** and then press **the Enter** key. Note that the **-f flag** must always be immediately followed by the name of the text file.

6 If necessary, at the UNIX prompt on the SGI, type **mv IncludeFilename $PGSINC** and then press the **Enter** key. Then, type **mv RuntimeFilename $PGSMSG** and then press the **Enter** key.

This step is only required if either the **-r or the -i flag** were not used in step 5.

The **IncludeFilename** is the name of the include file created in step 5.

The **RuntimeFilename** is the name of the runtime message file created in step 5.

For example, type **mv PGS_MODIS_39123.h $PGSINC** and then press the **Enter** key. And then type, **mv PGS_MODIS_39123 $PGSMSG** and then press the **Enter** key.

Building Science Software with the SCF Version of the SDP Toolkit

In order to be tested at the DAAC, science software must be compiled and linked to produce binary executables. These binary executables are then packaged into one or more shell scripts as defined by the science software developer (Instrument Team). These science software packages are the Product Generation Executives (PGEs) delivered to the DAACs during SSI&T. PGEs are the smallest schedulable unit of science software in the ECS.

Building science software into PGEs should be done in accordance with supplied documentation. Such documentation should describe the process in detail. In general, science software deliveries will come with make files or other build scripts to automate the build process.

In general, science software will be built, run, and tested with the SCF version of the SDP Toolkit to ensure that the software has been successfully ported to the DAAC. Once this test has been completed successfully, the science software will be re-built, rerun, and re-tested with the DAAC version of the SDP Toolkit. Only with the DAAC Toolkit can the PGE be run within the ECS.

This procedure describes some general principals that may or may not be applicable to a particular science software delivery for building a PGE with the SCF version of the SDP Toolkit. See Section for Building a PGE with the DAAC version of the SDP Toolkit.

Building Science Software with the SCF Version of the SDP Toolkit - Activity Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C shell (or a derivative) is the current command shell.

To build science software with the SCF version of the SDP Toolkit, be aware of the "typical" procedure steps that follow:

1 Read all instructional material supplied with the science software delivery. Such material should be the primary source of information on how to build the science software.

Read the ***Systems Description*** document and the ***Operations Manual***. Both of these or their equivalent should be in the delivery.

Typically, there will be **"readme"** files accompanying each PGE in the directory structure, perhaps in a doc directory.

Text files (ASCII) may be viewed with the UNIX command, ***more*** or with the ***vi*** editor.

PostScript documents may be viewed with ***ghostview***, which is accessible via the SSIT Manager.

PDF formatted documents may be viewed with ***acroread***, the Acrobat Reader, also accessible via the SSIT Manager.

Documents in Microsoft Word and related formats may be viewed through the Microsoft Windows™ 3.1 emulator. The MS Windows emulator may be accessed from the SSIT Manager.

2 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then **telnet** to the **SGI.**

Alternatively, in any currently available xterm window, spawn a new session: type **xterm &** and then press the **Enter** key. Then **telnet** to the **SGI.**

It is recommended that this procedure begin within a new command shell on the SGI.

3 At the UNIX prompt on the SGI, type **setenv PGSHOME *ToolkitPathname*** and then press the **Enter** key. Then type, source **$PGSHOME/bin/*sgiX*/pgs-dev-env.csh** and then press the **Enter** key.

The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version, in this case, an SCF version.

The ***sgiX*** refers to the appropriate processor. For example, type source **$PGSHOME/bin/sgi/pgs-dev-env.csh** and then press the **Enter** key.

4 If make files are in ClearCase, at the UNIX prompt on the SGI, type **cleartool setview *ViewName*** and then press the **Enter** key. Then, **cd *pathname*** and then press the **Ente**r key. And **cleartool checkout -nc *makefile*** and then press the **Enter** key.

The ***ViewName*** is the name of a view allowing the make files to be accessible.

The ***pathname*** is the full path name of the directory (in the VOB) where the make file has been checked in.

The ***makefile*** is the name of the make file to examine and possibly modify.

This step is only necessary if any of the make files (or build scripts) are in **ClearCase** (in the **VOB** under configuration management).

5 Examine and alter (if necessary) any **make** files using any text editor **(*vi, emacs*)**.

There may be several make files for a particular PGE.

Verify that compiler, compiler flag settings, and other environment variable settings are appropriate.

The Toolkit set up (from step 3) will set many environment variables which can be used in the make files. To see the current environment variable settings, at the UNIX prompt on the SGI, type **env** and then press the **Enter** key.

6 Compile any required status message facility (SMF) files and place the header file(s) in the proper directory for building. Example: **p0spg01{cmshared}>/usr/ecs/TS1/CUSTOM/ssit/PGE32/message**.

7 Verify that the directory structure for the PGE source files matches the directory structure expected by the make files or build scripts.

Deliveries may come with install scripts that place files into various directories according to some predefined structure.

8 If necessary, at the UNIX prompt on the SGI, type **cleartool checkout -nc *filename*** and then press the **Enter** key.

The ***filename*** is the file name of the executable, object file, or make file to be checked out of ClearCase. The **-nc flag** means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check out step.

Note that checking in an executable or object files is ***not*** recommended in the first place.

9 Build the software in accordance with instructions delivered.

Science software deliveries may come with a single, top-level script to do the entire build or the build process could involve a series of steps, each of which should be described fully in the delivered documentation.

Choose the most appropriate optimization/debugger flag. During testing, the **"-g"** is often used. This results in larger and slower executables, but assists in debugging. For production, the **"-O" flag** may be used to optimize execution time. Variants of the **"-g"** and **"-O" flags** may be incompatible.

10 If necessary, at the UNIX prompt on the SGI, type **cleartool checkin *filename* -nc** and then press the **Enter** key.

The ***filename*** is the file name of the executable, object file, or make file to be checked into ClearCase. The **-nc flag** means “no comment”; it suppresses ClearCase from prompting for a comment to be associated with the check in step.

Note that checking in executable or object files is ***not*** recommended.

Building Science Software with the DAAC Version of the SDP Toolkit

In general, science software will be built, run, and tested with the SCF version of the SDP Toolkit to ensure that the software has been successfully ported to the DAAC. Once this test has been completed successfully, the science software will be re-built, rerun, and re-tested with the DAAC version of the SDP Toolkit. Only with the DAAC Toolkit can the PGE be run within the ECS.

This procedure describes some general principals that may or may not be applicable to a particular science software delivery for building a PGE with the DAAC version of the SDP Toolkit.

Building Science Software with the DAAC Version of the SDP Toolkit - Activity Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The C shell (or a derivative) is the current command shell.

To build science software with the DAAC version of the SDP Toolkit, be aware of the "typical" procedure steps that follow:

1 Read all instructional material supplied with the science software delivery. Such material should be the primary source of information on how to build the science software.

Read the ***Systems Description*** document and the ***Operations Manual***. Both of these or their equivalent should be in the delivery.

Typically, there will be **"readme"** files accompanying each PGE in the directory structure, perhaps in a doc directory.

Text files (ASCII) may be viewed with the UNIX command, ***more*** or with the ***vi*** editor.

PostScript documents may be viewed with ***ghostview***, which is accessible via the SSIT Manager.

PDF formatted documents may be viewed with ***acroread***, the Acrobat Reader, also accessible via the SSIT Manager.

Documents in Microsoft Word and related formats may be viewed through the Microsoft Windows™ 3.1 emulator. The MS Windows emulator may be accessed from the SSIT Manager.

2 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then **telnet** to the **SGI.**

Alternatively, in any currently available xterm window, spawn a new session: type **xterm &** and then press the **Enter** key. Then **telnet** to the **SGI**.

It is recommended that this procedure begin within a new command shell on the SGI.

3 At the UNIX prompt on the SGI, type **setenv PGSHOME *ToolkitPathname*** and then press the **Enter** key. Then type, source **$PGSHOME/bin/*sgiX*/pgs-dev-env.csh** and then press the **Enter** key.

The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version, in this case, a DAAC version.

The ***sgiX*** refers to the appropriate processor. For example, type source **$PGSHOME/bin/sgi/pgs-dev-env.csh** and then press the **Enter** key.

4 If make files are in ClearCase, at the UNIX prompt on the SGI, type **cleartool setview *ViewName*** and then press the **Enter** key. Then, **cd *pathname*** and then press the **Ente**r key. And **cleartool checkout -nc *makefile*** and then press the **Enter** key.

The ***ViewName*** is the name of a view allowing the make files to be accessible.

The ***pathname*** is the full path name of the directory (in the VOB) where the make file has been checked in.

The ***makefile*** is the name of the make file to examine and possibly modify.

This step is only necessary if any of the make files (or build scripts) are in ClearCase (in the VOB under configuration management).

5 Examine and alter (if necessary) any make files using any text editor **(*vi, emacs*).** If the software had already been built and tested with the SCF version of the SDP Toolkit, this step may be unnecessary.

There may be several make files for a particular PGE.

Verify that compiler, compiler flag settings, and other environment variable settings are appropriate.

The Toolkit set up (from step 3) will set many environment variables that can be used in the make files. To see the current environment variable settings, at the UNIX prompt on the SGI, type **env** and then press the **Enter** key.

6 Compile any required status message facility (SMF) files and place the header file(s) in the proper directory for building.

7 Verify that the directory structure for the PGE source files matches the directory structure expected by the make files or build scripts.

Deliveries may come with install scripts that place files into various directories according to some predefined structure.

8 If necessary, at the UNIX prompt on the SGI, type **cleartool checkout -nc *filename*** and then press the **Enter** key.

The ***filename*** is the file name of the **executable, object file, or make file** to be checked out of ClearCase. The **-nc flag** means **"no comment"**; it suppresses ClearCase from prompting for a comment to be associated with the check out step.

Note that checking in executable or object files is ***not*** recommended in the first place.

9 Build the software in accordance with instructions delivered.

Science software deliveries may come with a single, top-level script to do the entire build or the build process could involve a series of steps, each of which should be described fully in the delivered documentation.

- Choose the most appropriate optimization/debugger flag. During testing, the **"-g"** is often used. This results in larger and slower executables, but assists in debugging. For production, the **"-O" flag** may be used to optimize execution time. Variants of the **"-g"** and **"-O" flags** may be incompatible.

10 If necessary, at the UNIX prompt on the SGI, type **cleartool checkin *filename* -nc** and then press the **Enter** key.

The ***filename*** is the file name of the **executable, object file, or make file** to be checked into ClearCase. The **-nc flag** means **"no comment";** it suppresses ClearCase from prompting for a comment to be associated with the check in step.

Note that checking in executable or object files is ***not*** recommended.

Running a PGE in a Simulated SCF Environment

Science software delivered to the DAACs for SSI&T was developed and tested at individual SCFs using the SCF version of the SDP Toolkit. Before linking the software with the DAAC version of the Toolkit and integrating it with the ECS, it is prudent to first link the software to the SCF version of the Toolkit and run it as it was run at the SCF. This type of testing can reveal problems associated with the process of porting the software to another platform whose architecture may be quite different from the one on which the software was developed.

A simulated SCF environment means that the software is built using the SCF version of the Toolkit and is run from the UNIX command line. The Planning and Data Processing System (PDPS) and the Data Server are not involved.

The procedures which follow describe how to run the science software in a simulated SCF environment.

Setting Up the Environment for Running the PGE

Running a PGE that has been built with the SCF version of the SDP Toolkit requires some environment set up as it does at the SCF. This procedure describes how to set up a simulated SCF environment.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The Process Control File (PCF) exists and has been tailored for the DAAC environment.
2. The C shell or a derivative (*e.g.* T shell) is the current user shell.

To set up an environment for running the PGE, execute the procedure steps that follow:

---

1 From the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then **telnet** to the **SPR SGI.**

Alternatively, in any currently available xterm window, spawn a new session: type **xterm &** and then press the **Enter key**. Then **telnet** to the **SPR SGI**.

It is recommended that this procedure begin within a new command shell on the SPR SGI.

2 At the UNIX prompt on the SPR SGI, type **setenv PGSHOME *ToolkitPathname*** and then press the **Enter** key. Then type, **source $PGSHOME/bin/*sgiX*/pgs-dev-env.csh** and then press the **Enter** key.

The ***ToolkitPathname*** is the home directory of the desired SDP Toolkit version, in this case, an SCF version. For example, **p0spg01{cmshared}>setenv PGSHOME /usr/ecs/TS1/CUSTOM/scf_toolkit_f77.**

The ***sgiX*** refers to the appropriate processor. For example, type source **$PGSHOME/bin/sgi/pgs-dev-env.csh** and then press the **Enter** key.

3 At the UNIX prompt on the SPR SGI, type **setenv PGS_PC_INFO_FILE *PCFpathname*/*PCFfilename*** and then press the **Enter** key.

The ***PCFpathname*** is the full path name to the location of the Process Control File (PCF) to be associated with this PGE.

The ***PCFfilename*** is the file name of the PCF.

For example, setenv PGS_PC_INFO_FILE /disk2/PGE32/PCF/PGE32.pcf and then press the Enter key.

4 Optionally, at the UNIX prompt on the SPR SGI, type **rm *LogPathname*/*LogFilename*** and then press the **Enter** key.

The ***LogPathname*** is the full path name to the location of the PGE log files for this PGE.

The ***LogFilename*** is the file name of the **PGE log file** to remove from a previous run of the same PGE. PGE log files can be **Status, User**, or **Report**.

The ***LogFilename*** may use wildcard characters to remove all of the log files at the same time.

This step is optional. If log files from a previous run of the same PGE are not removed, they will be appended with the information from the current run.

The environment will then be set up. Continue on the next Section.

5 If necessary, set any other shell environment variables needed by the PGE by sourcing the appropriate scripts or setting them on the command line.

For example, for a PGE requiring IMSL, at the UNIX prompt on the SPR SGI, type **source /usr/ecs/TS1/COTS/imsl/vni/ipt/bin/iptsetup.cs**, press **Enter**

For some PGEs, the environment variables to be set will be specified in the documentation or the files to source will be supplied in the delivery. Refer to documentation included in the delivery.

Running and Profiling the PGE

Profiling a PGE refers to the process of gathering information about the runtime behavior of a PGE. The information includes the wall clock time, user time and system time devoted to the PGE; the amount of memory used; the number of page faults; and the number of input and output blocks.

The Planning and Data Processing System (PDPS) database must be populated with the above information when the PGE is registered with the PDPS during the integration phase of SSI&T. This information may be delivered with the PGE or it may need to be determined at the DAAC during SSI&T. This procedure addresses the latter need.

Note that profiling, as used here, does not involve altering the binary executable to produce instrumented code.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The PGE has been built successfully with the SCF version of the SDP Toolkit.
2. The required SMF runtime message files have been produced and placed in the correct locations.
3. The Process Control File (PCF) exists and has been tailored for the DAAC environment.
4. The required environment for running the PGE has been set up.
5. The required input files are available and accessible.
6. The C shell or a derivative **(*e.g.* T shell)** is the current user shell.

To run and profile the PGE, execute the procedure steps that follow:

1 At the UNIX prompt on the SPR SGI in the window containing the set up environment, type **cd *PGEbinPathname***, press **Enter**.

- The ***PGEbinPathname*** is the full path name of the directory containing the built PGE binary executable. For example, **p0spg01{cmshared}cd /usr/ecs/TS1/CUSTOM/ssit/PGE32/bin/**, press **Enter**.

2 At the UNIX prompt on the SPR SGI, type: **/usr/ecs/<mode>/CUSTOM/bin/DPS/E*cDpPrR*usage *PGE.exe* >& *ResultsOut***, press **Enter**.

- The ***PGE.exe*** is the name given to the PGE binary executable.
- The ***ResultsOut*** is the file name in which to capture the profiling results as well as any messages from standard output **(stdout)** and standard error **(stderr)** that may be produced by the running PGE. Note that PGEs should ***not*** write to stdout or stderr.
- For example:
  p0spg01{cmshared}>/usr/ecs/TS1/CUSTOM/bin/DPS/EcDpPrRusage run_PGE32.exe >& ../logs/PGE32.profile.out
- The **EcDpPrRusage** is the profiling program that outputs information about the runtime behavior of the PGE.

- Depending upon the PGE, it may take some time before the UNIX prompt returns.

3 At the UNIX prompt on the SPR SGI, type **echo $status**, press **Enter.**

- The **$status** is an environment variable that stores the exit status of the previous program run, in this case, the PGE.
- A status of zero indicates success; a status of non-zero indicates an error of some kind.
- The meaning of a non-zero exit status should be documented and included with the DAPs.
- This command must be run ***immediately*** after the **EcDpPrRusage** command.

4 At the UNIX prompt on the SPR SGI, type **vi *ResultsOut***, press **Enter**.

- The ***ResultsOut*** is the file name under which the profiling output was saved. Other output of the PGE may also be in this file.
- The **EcDpPrRusage** results may then be recorded and used when the PGE is registered in the PDPS.
- Any text editor/viewer may be used.

Sample of an Reusage File produced:

```
p0spg01{emcleod}6: more Profile.out
#     source .cshrc
#     cd TEST/MOD*
#     ls
#     /usr/ecs/OPS/CUSTOM/bin/DPS/EcDpPrRusage MOD_PR10.exe > Profile.out
p0spg01{emcleod}6:
p0spg01{emcleod}9: more profile.dat
# Resource Usage Information
COMMAND=MOD_PR10.exe
EXIT_STATUS=0
ELAPSED_TIME=233.583145
USER_TIME=10.046158
SYSTEM_TIME=7.555547
MAXIMUM_RESIDENT_SET_SIZE=4080
AVERAGE_SHARED_TEXT_SIZE=0
```

AVERAGE_UNSHARED_DATA_SIZE=0

AVERAGE_UNSHARED_STACK_SIZE=0

PAGE_RECLAIMS=151

PAGE_FAULTS=0

SWAPS=0

BLOCK_INPUT_OPERATIONS=2

BLOCK_OUTPUT_OPERATIONS=2710

MESSAGES_SENT=0

MESSAGES_RECEIVED=0

SIGNALS_RECEIVED=0

VOLUNTARY_CONTEXT_SWITCHES=1095

INVOLUNTARY_CONTEXT_SWITCHES=2

p0spg01{emcleod}10:

Checking the PGE for Memory Leaks

One of the important type of "code inspection" is to check the PGE for memory leak. This session assumes that PGE build and command line run both been successfully run. The task requires to build and run with a memory error checking utility (such as Purify).

Integration and testing are conducted using the Goddard Automated Tools for Enhanced SSI&T (GATES) procedures. The GATES procedures are available on the web at:

http://gsfcsrvr8.gsfcmo.ecs.nasa.gov/GATES/TEST/AIRS/AIRS_FauxSwatter_EnterInfo.html.TEST

This page intentionally left blank.

The ECS Assistant Functionality Replaced in Part by Scripts and Monitor GUI Whazzup

ECS (EOSDIS Core System) is a complex system comprising of many subsystems and components running on multiple heterogeneous host machines. The coordination of all the subsystems for SSI&T is an arduous, time consuming, error prone task. In order to improve our effectiveness and efficiency, an easy-to-use GUI tool, “ECS Assistant,” has been developed to facilitate ECS SSI&T activities. Its origin as a development tool provides "fallback" functionality for other tools, such as HP OpenView. Because of high overhead use of system resources, ECS Assistant has been scaled back in functionality.

Currently, the ECS Assistant has lost its capability to monitor ECS. This capability now exist with GUI Whazzup. ECS Assistant continues to have the ability to Monitor Subsystem functions and is mainly used for doing ECS Assistant Subsystem Installs (E.A.S.I.) and staging ESDT/DLL's into the directories, CUSTOM/data/ESS and CUSTOM/lib/ESS respectively. Database Review capability still exists however. The ESDT Manager Installation/Deletion functions are no longer available, the ability to startup and shutdown subsystem servers and the use of the ECS Logfile Viewer have been taken away in Release 6A.

The use of scripts provides users with the means to perform functions such as subsystem server startup and shutdown. During the course of performing these tasks, SSI&T operators can use the following Scripts to perform the following functions:

To start up or shut down all servers at the same time, a Script is used that accesses a list of subsystem servers.

To start up and shut down servers individually using a Script established within each subsystem.

To graphically monitor the server up/down status with the Whazzup GUI

To view ESDTs for SSI&T

To **review various databases** used in the ECS system by using an **ISQL Browsers** established in each subsystem.

In the following sections, we will address aspects of how to use **Scripts, the Whazzup GUI** and portions of ECS Assistant in our SSI&T activities.

Section one explains how to use **Scripts** to facilitate and manage the subsystems and their servers, including **server start up and shut down**.

Section two contains an **ESDT monitor function** which includes reviewing the Science Data Server database through the **DB Viewer GUI** provided by ECS Assistant

Using Scripts to Start Up / Shut Down Servers

The DAACs may have established their own scripts to Start Up/Shut Down Subsystem servers. This procedure describes Scripts that are used at Landover on the VATC and PVC systems to start up and shut down subsystem servers. The procedure described here will apply to all the servers from different subsystems as well as individual servers.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow

---

All Servers Start Up/Shut Down

1 Script name: **rcmd_start_mode TS1, rcmd_kill_mode TS1**

2 Location: **/home/cmshared/bin/**

3 **HOSTLST** contains all the Subsystem Servers that will be operated upon.

Individual Servers Start Up/Shut Down

1 If you want to Start or Kill a server, you must be on the individual machine that supports the server in question.

2 Type **cd /home/cmshared/bin/** <**ENTER**>

3 Type **start_mode TS1** to start up a server or **kill_mode TS1** to shutdown a server**.** <**ENTER**>

4 Repeat steps 1-3 to start up or shut down other servers.

Bringing Up ECS Assistant

Assumptions:

1. The ECS Assistant has been properly installed.
2. The required environment variables have been set properly.

Note: SGI machines are not a part of ECS Assistant functionality.

To Run The ECS Assistant

---

1 At the UNIX Console or Terminal **setenv DISPLAY clientname:0.0** and then press the **Enter** key.

2 Create an xterm by typing**: xterm -n hostname &**

The ***hostname*** is the name of the machine on which the ECS Assistant is to be displayed, *i.e.*, the machine that your are using.

3 Log into one of the host machines used for SSIT, (Tested using **telnet p0acs03** for **SDSRV** and **p0ins02** for **IOS**, **ID:, PW:**

4 At the UNIX Console or Terminal type **setenv DISPLAY clientname:0.0** and then press the **Enter** key**.**

To verify the setting, type **echo $DISPLAY** and then press the **Enter key.**

setenv ECS_HOME/usr/ecs

setenv TK_LIBRARY/tools/lib/tk4.2

(Mount point called /tools must be mounted.)

5 If necessary, at the UNIX prompt on the host from which the ECS Assistant is to be run, type **cleartool setview *ViewName*** and then press the **Enter** key.

The ***ViewName*** is the ClearCase view to be used while the ECS Assistant is running in this session. For example, type: cleartool jdoe and then press the **Enter** key.

A ClearCase view is required only if the ECS Assistant needs to be able to "see" into a ClearCase VOB; a view is not necessary otherwise.

6 At the UNIX prompt, type **EA** and then press the **Enter** key.

- File **/tools/common/ea** must exist in the path. (This can be set in the **.cshrc** or **.kshrc** file)

EA is an alias for: **/tools/common/ea** is the path where ECS Assistant is installed.

This will invoke the ECS Assistant GUI with three push buttons for selecting the proper activities, as indicated in Figure 15.

Figure 15. ECS Assistant Main GUI

7 At the ECS Assistant GUI, click the **Subsystem Manager** pushbutton.

This will invoke the Subsystem Manager GUI, as indicated in Figure 16.

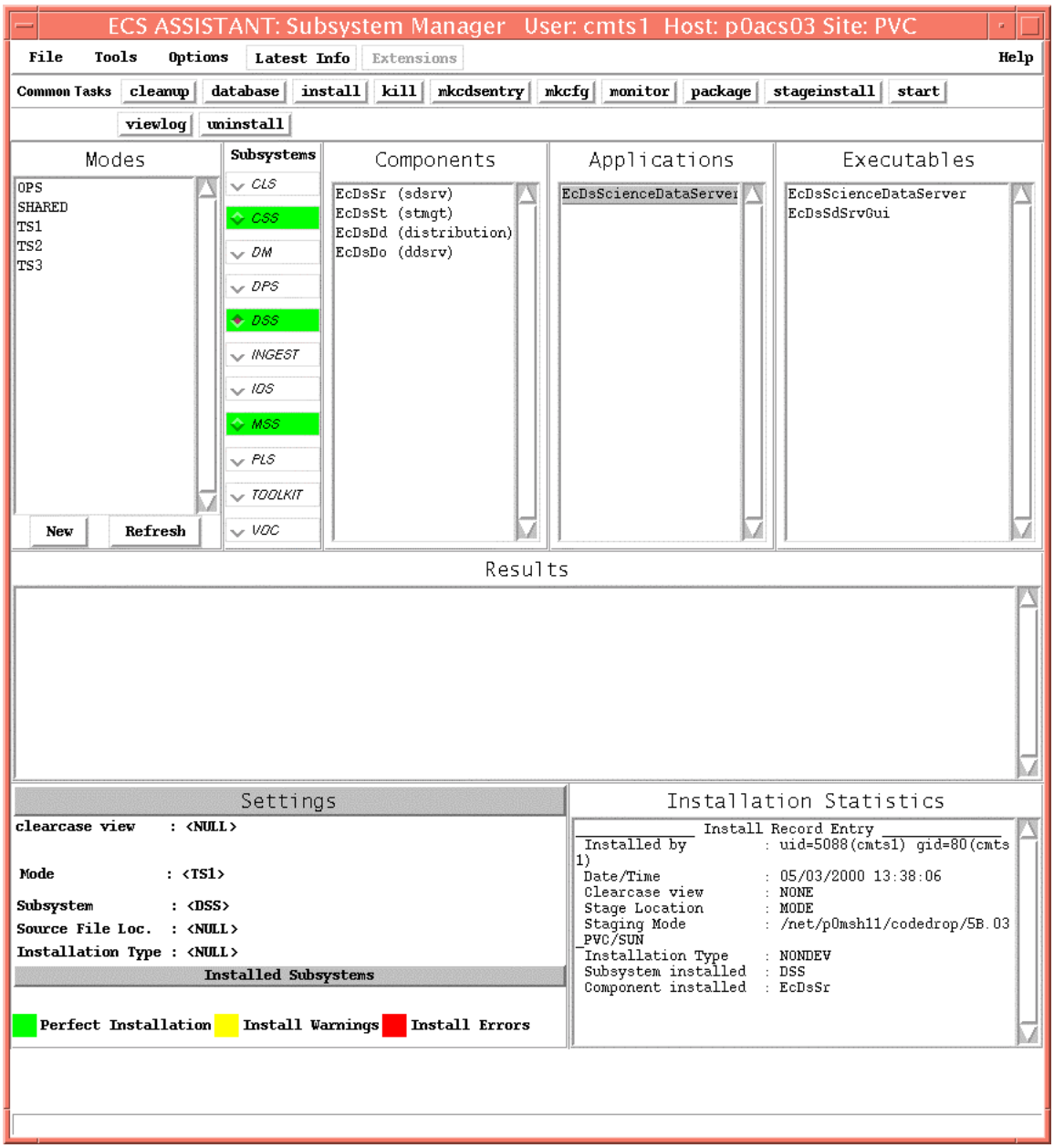


Figure 16. Subsystem Manager GUI

8 Select a mode by clicking a mode in the mode listing. The mode should be the one to be used for SSI&T.

Once the mode is selected, the color of the subsystem name list is changed.

9 Select a subsystem with the **Subsystem** radio button.

The component list for the selected subsystem will appear in the component window.

10 Select a component by clicking the component name under the component window.

The selected component will be highlighted.

The server list corresponding to that component will appear in the server window.

11 Select a server by clicking the server name from the server list under the servers window.

The server selected is highlighted.

Note that the functionality for Viewlog, Kill, Start and Monitor are not supported for Release 6A. Refer to the section titled: "Using Scripts to Start Up / Shut Down Servers."

13 To exit the Subsystem Manager GUI, select **File..Exit** in the menu bar of the Subsystem Manager GUI.

This will terminate the Subsystem Manager GUI.

Monitoring ECS using WHAZZUP Web GUI

To use the WHAZZUP GUI to monitor ECS make the following entries to display the GUI.

1 Telnet into a server that has Web access and links to the WHAZZUP GUI.

2 **Setenv DISPLAY ..:0.0**

3 **/tools/bin/ssh -l cmshared t1code1u.ecs.nasa.gov**

4 Password:

5 **setenv DISPLAY :0.0**

6 **ssh t1pls02**

7 Enter **tools/bin/netscape/** when the Netscape GUI is displayed, then select from the Bookmark: **whazzup.** A GUI such as the one depicted in Figure 17 should appear on the terminal screen.

Notice the other links listed in the bookmark List depicted in Figure 18, that go to sites that support ECS.

Figure 17. WHAZZUP ECS Monitor GUI

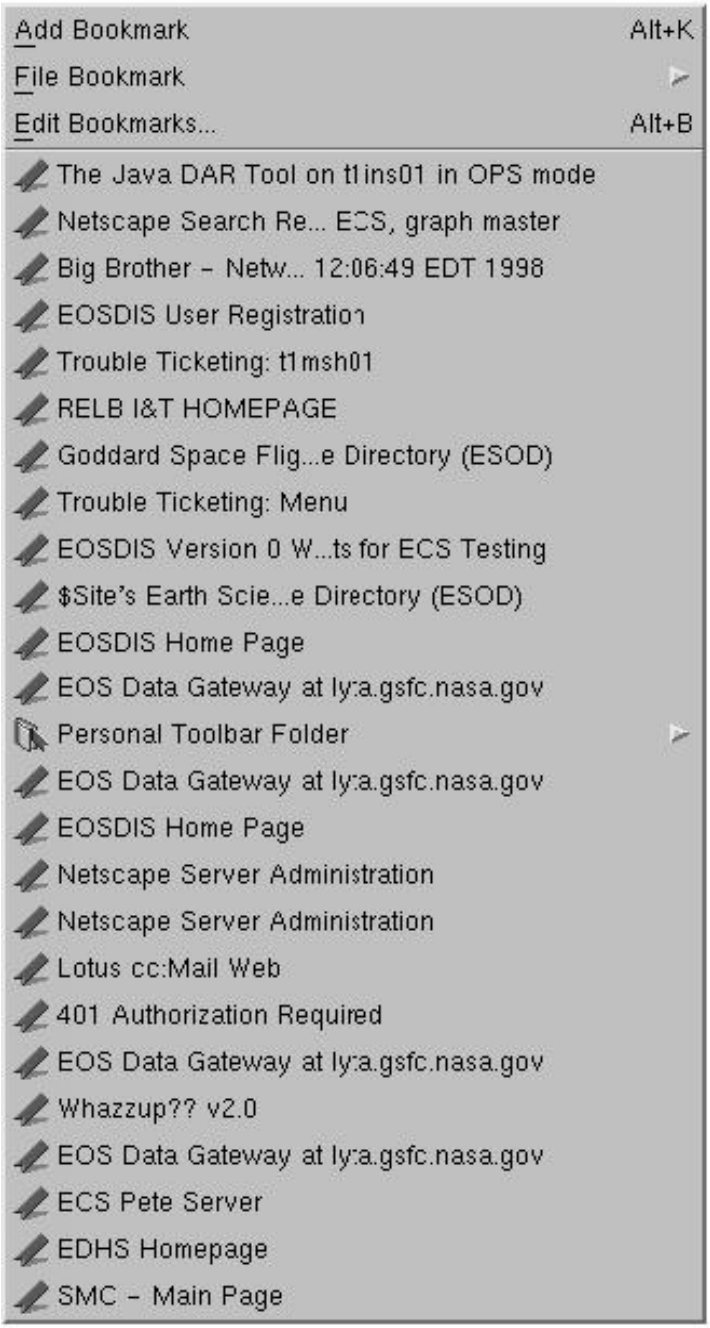


Figure 18. WHAZZUP Bookmark List

Using ECS Assistant to View ECS Science Data Server Database

ESDTs and their granules can be viewed in the ECS Science Data Server database. ECS Assistant provides an easy way to review the records stored in this database by using the ECS Assistant DB Viewer. There are two main windows in the DB Viewer. The first is called Collections and is used to display ESDT information included in the Collection database table. Information listed in this table includes ESDT short names, times last updated, types, etc. If an ESDT is added to the Science Data Server, its record will be shown in this window. The other window is called Granules and is used to display information included in the Granule database table. If a granule is inserted for an ESDT, the granule information will be listed in this window if its ESDT is highlighted in the Collection window. In addition to these two main windows, this DB Viewer GUI can also show ESDT database validation rules, PSA information, and summary information about the database reviewed.

Detailed procedures for the SSI&T operator to perform are provided in the sections that follow.

Assumptions:

1. The ECS Assistant has been properly installed.
2. The ECS Subsystem Manager is running.
3. The environment variables for using the database have been set correctly.

To start up the ECS monitor GUI, execute the procedure steps that follow:

1 Follow Section Bringing Up ECS Assistant to invoke the ECS Assistant GUI.

- The ECS Assistant GUI will be launched.

2 At the ECS Assistant GUI, select ESDT Manager GUI by clicking the ESDT Manager.

- The ESDT manager GUI will appear. See Fig. 19.

3 At the ECS ESDT Manager GUI, select the DB Viewer by clicking the **DB Viewer** button. See Figure 19.

- The Database Login GUI will appear as shown in Figure 20.
- Fill in the fields to point to the specific database for the mode used.
- Click Login to open the DB Viewer.
- The DB Viewer GUI will appear as shown in Figure 21.
- ESDTs are listed in the Collections window.

4 To view the inserted granules for a selected ESDT, first select an ESDT by clicking its short name in the Collections window.

- The selected ESDT is highlighted.
- Granule information for that ESDT, if there is any, will be listed in the Granules window.

5 To exit, click the **EXIT** button. This will end the DB Viewer

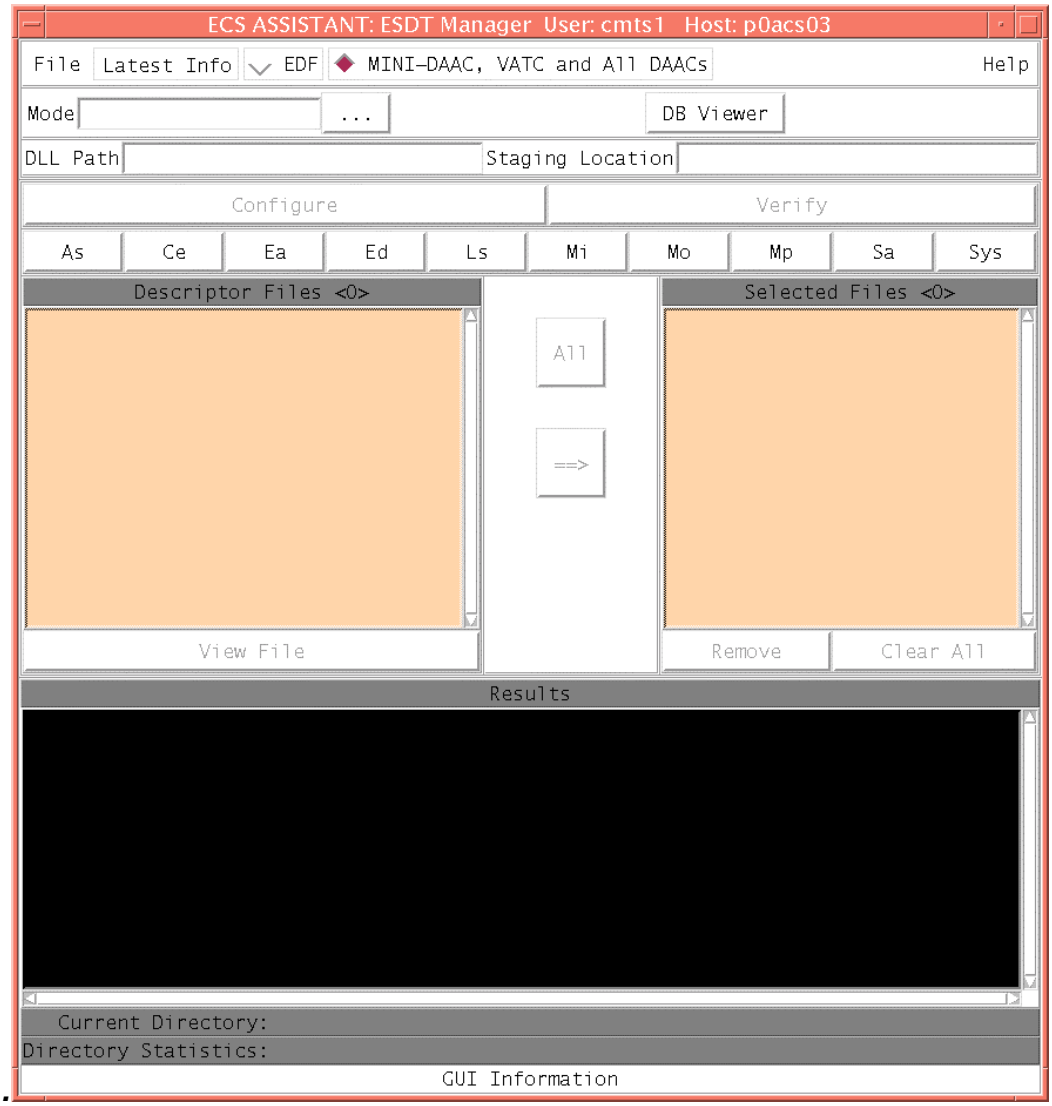


Figure 19. ESDT Manager GUI

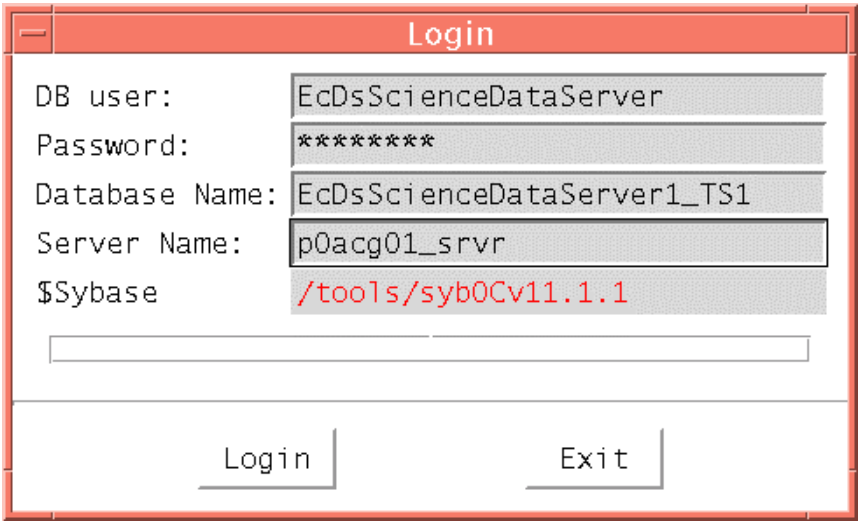


Figure 20. Database Login GUI

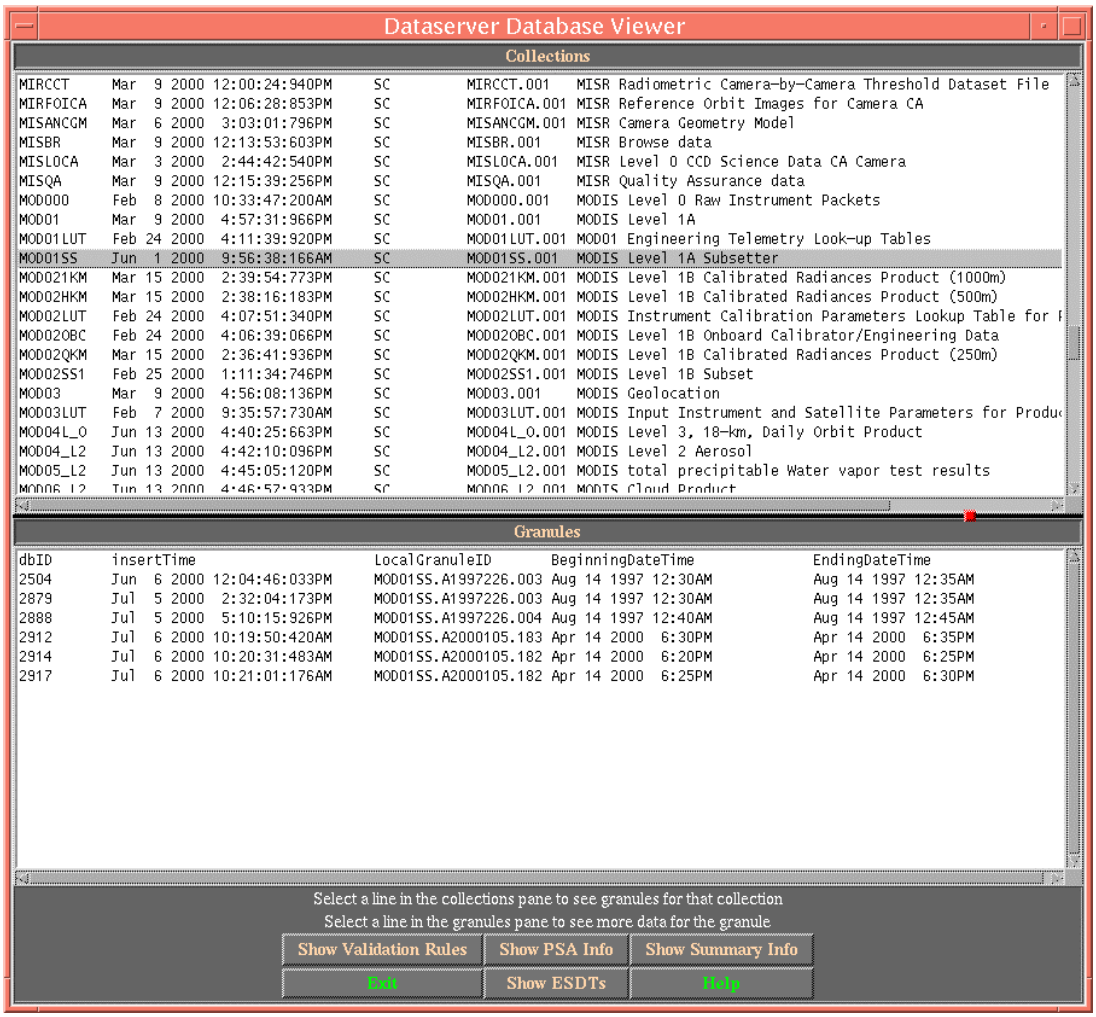


Figure 21. DB Viewer GUI

Using ECS Assistant to View ECS Science Data Server Database

1. Invoke the ECS Assistant GUI.
2. At the ECS Assistant GUI, invoke the **ESDT Manager**.
3. At the ECS ESDT Manager GUI, select the **DB Viewer** to open the login window for the database.
4. Fill in the Login field and open **DB Viewer** for the selected database.
5. View the inserted granules for a selected ESDT.

Preparation of Earth Science Data Types (ESDT's)

Every science data product generated and archived by the ECS must be described to the system by metadata which are put into an inventory and then used to retrieve and distribute the data to users of the system. The ECS Earth Science Data Model organizes the metadata into groups of related attributes and services to be performed on the data products. Granules of the same type of science data are grouped into collections. Every collection is described by an Earth Science Data Type (ESDT) and is made known to the system by an ESDT descriptor file and associated software code which is built into the Data Server's dynamic link library (DLL) to perform the services. The ESDT descriptor is composed of sections containing the following information:

Collection level metadata attributes with values contained in the descriptor.

Granule level metadata attributes whose values are supplied primarily by the Product Generation Executives (PGEs) during runtime.

Valid values and ranges for the attributes.

List of services for the data and events which trigger responses throughout the system.

The ESDTs for all data collections to be input to or output from the PGEs must be built and registered into the ECS before any of the PGEs can be run under the automated processing system.

During the past year, the ECS has collected information from the Instrument Teams on the ESDTs needed for their science software in the Release 6A. This information has been baselined and a set of ESDT descriptor files have been built and tested according to this baseline. The baselined ESDT descriptor files reside in the Release 6A under ECS configuration management in a VOB called the ECS Configuration Area. Since science software has been developed to the baseline, any changes to the baselined ESDT descriptor files should be rare.

Comparing Granule Level Metadata

A PGE accesses granule level metadata attributes and values via a Metadata Configuration File (MCF). There is typically one MCF for each output data set. The ESDTs which have been built and registered in the ECS contain a section for granule level metadata attributes and values for each data set. In terms of content, the MCFs and the granule level metadata section of the corresponding ESDT descriptor files have to be in sync.
Few changes are expected in the Inventory section of the MCF. Changes are more likely to be expected in the Archive section of the MCF and in the Product Specific Metadata. If there are any changes, a new version of the baselined ESDT descriptor file must be generated.

Installing/Removing (ESDT/DLL) using the Science Data Server Operator GUI

Before the ECS can process data, an Earth Science Data Type must be installed into the system via the Science Data Server (SDSRV). The ESDT allows the system to recognize a particular data type and also provides services for accessing the data in the form of a Dynamic Link Library (DLL). The following procedures give step-by-step instructions on configuring the ESDT and installing the ESDT using the Science Data Server GUI., see Figure 22. Science Dataserver Operator GUI.

Installing a single Earth Science Data Type (ESDT with Dynamic Link Library (DLL)

1 Copy the ESDT descriptor file and ESDT/DLL file from the source directory to the directory under the current mode of operations, The ESDT descriptor files are installed in the specified mode.

DLL's located : /usr/ecs/<mode>/CUSTOM/lib/ESS

ESDT Descriptors Located: /usr/ecs/<mode>/CUSTOM/data/ESS

2 Ensure that the following servers are currently executing: **Advertising Service** on the appropriate ADSHW HWCI server machine, **Data Dictionary Service** on the appropriate DMGHW HWCI server machine, **Science Data Server** on the appropriate ACMHW HWCI server machine and the **Subscription Service** that operates on the appropriate CSS server machine.

3 Start the **SDSRV GUI** by entering the following at the UNIX prompt on the SDSRV GUI workstation:

On workstation **x0acs##,** at the UNIX prompt in a terminal window, type as in step a, below, then enter your user id and password.

NOTE: The x in the workstation name will be a letter designating your site:

g = GSFC, m = SMC, l = LaRC, e = EDC, n = NSIDC, o = ORNL, a = ASF, j = JPL, p = PVC; the ## will be an identifying two-digit number (e.g.,g0acs03 indicates a Science Data Server Subsystem (SDSRV) workstation at GSFC).

Then rlogin, enter **setenv DISPLAY <local_workstation IP address>:0.0**. The **<ipaddress>** is the ip address of **x0acs##**, and xterm is required when entering this command on a Sun terminal.

Quick Start Using the Science Data Server

To invoke the ECS Science Data Server GUI, the user types the following command line:

---

1 **telnet** to (SDSRV) **p0acs03** [e.g.]
2 login**: ID, password:**
3 **cd /usr/ecs/<mode>/CUSTOM/utilities/EcDsSdSrvGuiStart <mode>**
4 **>EcDsSdSrvGuiStart** <**mode**>

- <**mode**> is the ECS Mode for the execution, e.g., **OPS, TS1.**
- This command will bring up the GUI with the main screen appearing. The user can then initiate the actions described in the following section.

5 On the main screen select the **Data Types** tab. A list of the ESDTs that have already been installed on the SDSRV will be displayed.

6 Click the **Add** button, this will bring up a first smaller GUI. Go below to bring up the **Add Data Type** window, this will bring up a second small GUI.

7 **Descriptor Filename:** enter path to where the ESDT/DLL is located, including the full ESDT descriptor. Click **OK**. Control goes back to the first GUI.

Note: Within the ESDT descriptor there is a DLL Filename identifying which DLL is to be used with this ESDT.

The descriptor filename and DLL Filename will require the complete directory path name as part of the file name which is the same directory as was specified in step 1 above. (isolate the particular Data Type from the larger List, by using a unique sequence of letters or numbers at the end of the full path to better identify the Data Types ie;**/*___*).**

To specify specific directories, the File.. button to the right of the Descriptor Filename and DLL Filename data entry fields will bring up a standard file selection GUI for this purpose. Also note that the Archive ID field will be constructed using the DSS Storage Management Staging Server UR that is found in the Science Data Server configuration file. The Science Data Server Configuration file is located in:

/usr/ecs/<mode>/CUSTOM/cfg/EcDsScienceDataServer.CFG.

Example: If the **DSSSTMGSTAGEINGSERVERUR** field was set to **DRP1_OPS:VG1** then the Archive ID fields would be set automatically to **DRP1_OPS.**

8 Click the **Ok** button, this will cause the **Add Data Type** window to initiate installation of the ESDT/DLL into the Science Data Server.

9 The Science Data Server GUI will respond in a short time with a window stating that: **MM/DD/YY HH/MM Finished adding ESDT's**. Also, the ESDT will appear alphabetically on the **Science Data Server - Data Types** list under the **Data Types** tab.

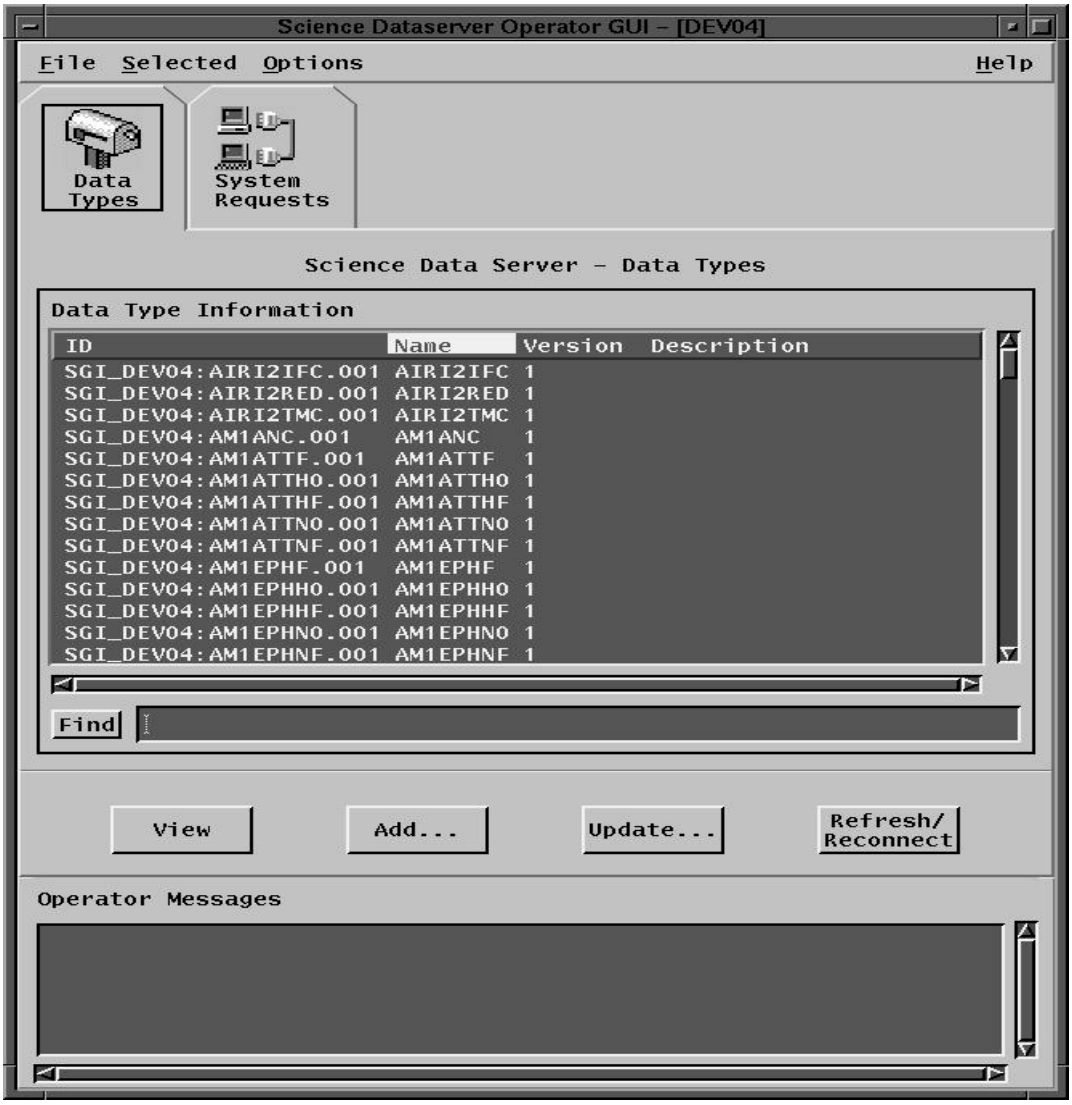


Figure 22. Science Dataserver Operator GUI Shown with Default Data Types Tab.

Validating Successful ESDT Installation

Criteria for success:

The **SDSRV** will display an Event ID to the fact that a new ESDT has been installed successfully.

The following servers will also need to have acknowledged a successful ESDT Event ID before additional work can be done: **ADSRV, DDICT, & SBSRV**.

Science Data Server

The Science Data Server Operator GUI provides the operator two major functions, the management of Earth Science Data Types and the management of all types of requests the Science Data Server Operator is involved with. Further details on these two functions are given in Table 6.

Table 6. Common ECS Operator Functions Performed with the Science Data Server Operator GUI

Operating Function	GUI	Description	When and Why to Use
Manage Science Data Server Earth Science Data Types (ESDTs)	**Data Types Tab**	**Allows operators to manage the ESDTs offered by the Science Data Server**	**As needed, to manage data type descriptor information and add and update ESDTs**
Manage Data Server System Requests	**System Requests Tab**	**Allows operators to manage all the requests within each data server component**	**As required, to manage requests in each data server component**

Science Data Server Main Screen

The ECS Science Data Server operator GUI, shown in Figure 22, has two tabs that provide access to each one of the components' screens.

- The Earth Science Data Type Manager is accessed through the **Data Types** tab
- The System Request Manager is accessed through the **System Request** tab.

The operator can select from the menu bar items at the top of the Science Data Server Operator window for getting help and activating less-frequently used functions. The menu bar capability is available on all Science Data Server Operator GUI screens. The following menus are available:

- **File** - which includes the following item:

  Exit (Ctrl-Q) - Exit application (graceful exit).

- **View** - functionality has not been determined as of this time (TBS).

- **Options** - This menu includes the *System Settings* item that opens the Server Polling Options window. Polling of the data server can be switched On/Off and the SdSrv Polling rate can be adjusted through this window shown in Figure 23.
- **Help** - which provides context sensitive help.

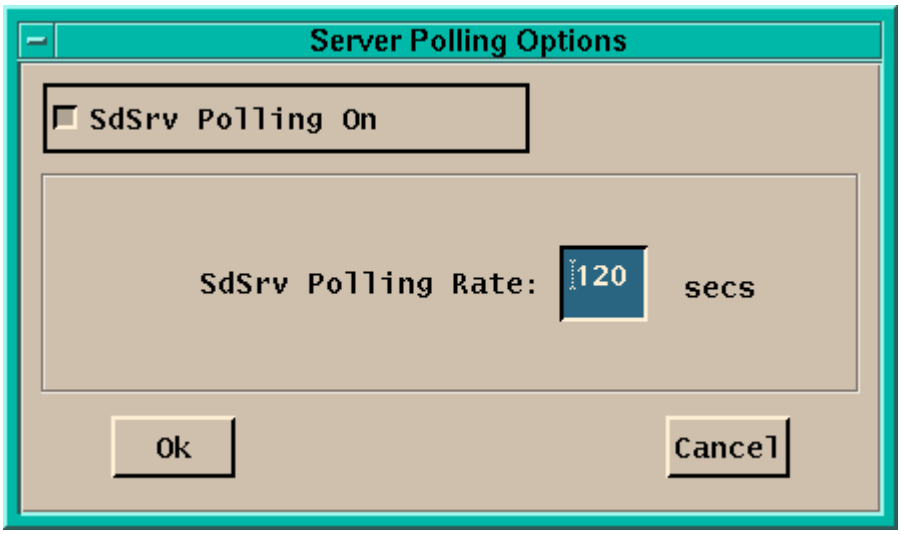


Figure 23. Science Data Server - Server Polling Options

Table 7 describes the information fields on the Server Polling Options screen.

Table 7. Science Data Server - Server Polling Field Description

Field Name	Data Type	Size	Entry	Description
Polling Rate	integer	4 digits	Optional	Specify the rate at which the Science Data Server Operator GUI is updated with data coming from the Data Server. The polling rate default is 120 seconds.

Data Types Tab

The Data Types Tab is the default screen of the Science Data Server Operator GUI shown in Figure 23. This window provides operations personnel at the DAAC the capability to view descriptor information, add new ESDTs and update ESDTs. A list of currently installed ESDTs is shown along with a version number and a brief description of the structure for an ESDT. Additional information that describes the structure, contents, and services that each existing ESDT provides can be viewed by selecting the data type and clicking on the *View* button.

Table 8. describes the Science Data Server Operator - Data Types fields.

Table 8. Science Data Server Operator - Data Types Field Description

Field Name	Data Type	Size	Entry	Description
Data Type ID	character	8	System generated	Uniquely identifies the specific type of ESDT.
Name	Character	25	System generated	Name of ESDT.
Version	Integer	3	System generated	Version number of ESDT, assigned starting at 1.
Description	Character	255	System generated	Includes structure and services available for an ESDT.
Find	Character	255	Optional	This functionality is provided in order to help the user browsing very long ESDT lists.

In addition, the following buttons are provided:

- **View** displays ESDT descriptor information (read-only) and its associated dynamic data link library (DLL) filename. Descriptor information consists of groups, objects, and keywords that define an ESDTs metadata, advertised services, subscribable events, data dictionary information, validation criteria, and science parameters. Descriptor information is necessary for the Science Data Server to properly configure itself to perform services related to an ESDT. A DLL is an executable library that is loaded dynamically when needed to fulfill ESDT services. The Science Data Server - Descriptor Information Dialog (see Figure 24 below) provides the following buttons:
  - **Close** exits the dialog without performing any operations.

– **Help** displays on-line help information.

Figure 24. Science Data Server - Descriptor Information Dialog

- **Add** opens the Data Type Dialog (see Figure 25) which is used to add a new ESDT to the existing installed list of data types based upon input information. The SDSRV GUI has the capability to install multiple ESDTs. Click on the **File…** button to display a list of descriptor filenames to choose from instead of typing them in. Multiple descriptor files can be selected. Click the **OK** button to add the data type. If no error messages appear, then the operation has been successfully completed. Click the **Cancel** button to close the dialog without performing an operation. Click the **Clear** button to start all over again the process of filling in new information. Click the **Help** button to display on-line help information.

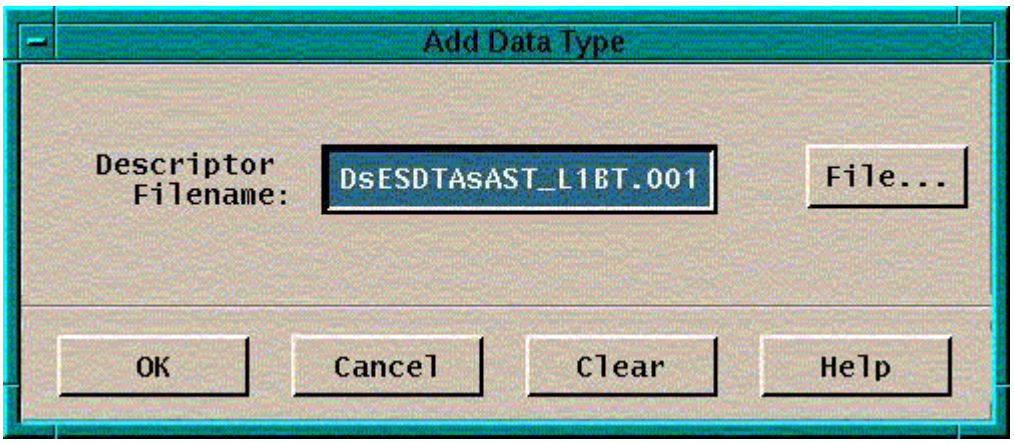


Figure 25. Science Data Server - Add Data Type Dialog

Note: The ESDT ADD GUI is similar in functionality to the Update ESDT GUI.

Table 9, describes the Science Data Server - Add Data Type fields.

Table 9. Science Data Server - Add Data Type Field Description

Field Name	Data Type	Size	Entry	Description
Descriptor Filename	Character string	25	required	Name of an ASCII file containing the ESDT descriptor file.

- **Update** opens the Update ESDT Dialog (see Figure 26) which is used to update a ESDT to the installed list of data types based upon input information. The SDSRV GUI provides the capability to update multiple ESDTs at one time. The Science Data Server needs to be running in **Maintenance mode** to accept this operation. Click on the **File…** button to display a list of descriptor filenames to choose from instead of typing them in. Multiple descriptor files can be selected. Click the **OK** button to update the data type. If no error messages appear, then the operation has been successfully completed. Click the **Cancel** button to close the dialog without performing an operation. Click the **Clear** button to start all over again the process of filling in new information. Click the **Help** button to display on-line help information.

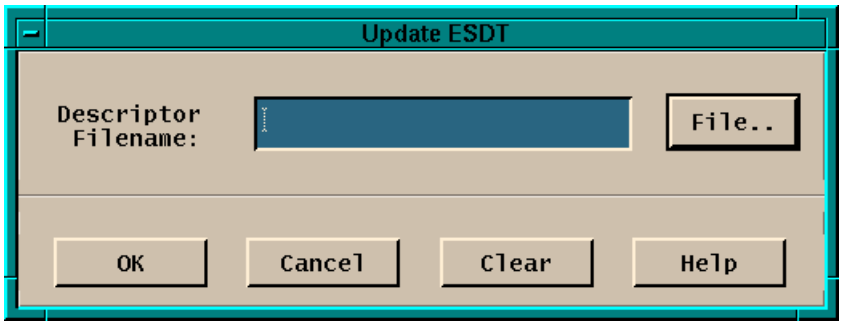


Figure 26. Science Data Server - Update Data Type Dialog

Table 10, describes the Science Data Server - Update Data Type fields.

Table 10. Science Data Server - Update Data Type Field Description

Field Name	Data Type	Size	Entry	Description
Descriptor Filename	character string	255	required	Name of an ASCII file containing the ESDT descriptor file.

- **Refresh/Reconnect** updates the data type information screen with current information.
- **Operator Messages** provides the functionality that displays informational and error messages to the user.

System Requests Tab

Clicking the System Requests tab will bring up the System Management Requests window (see Figure 27). This window provides operations personnel at the DAAC the capability to monitor requests the Science Data Server is working with. All requests within the Science Data Server are displayed. The columns of the list can be sorted by positioning the cursor and clicking on the appropriate column of interest. The requests can be filtered by positioning the cursor and clicking on the **Filter** button and entering the attributes on which to filter.

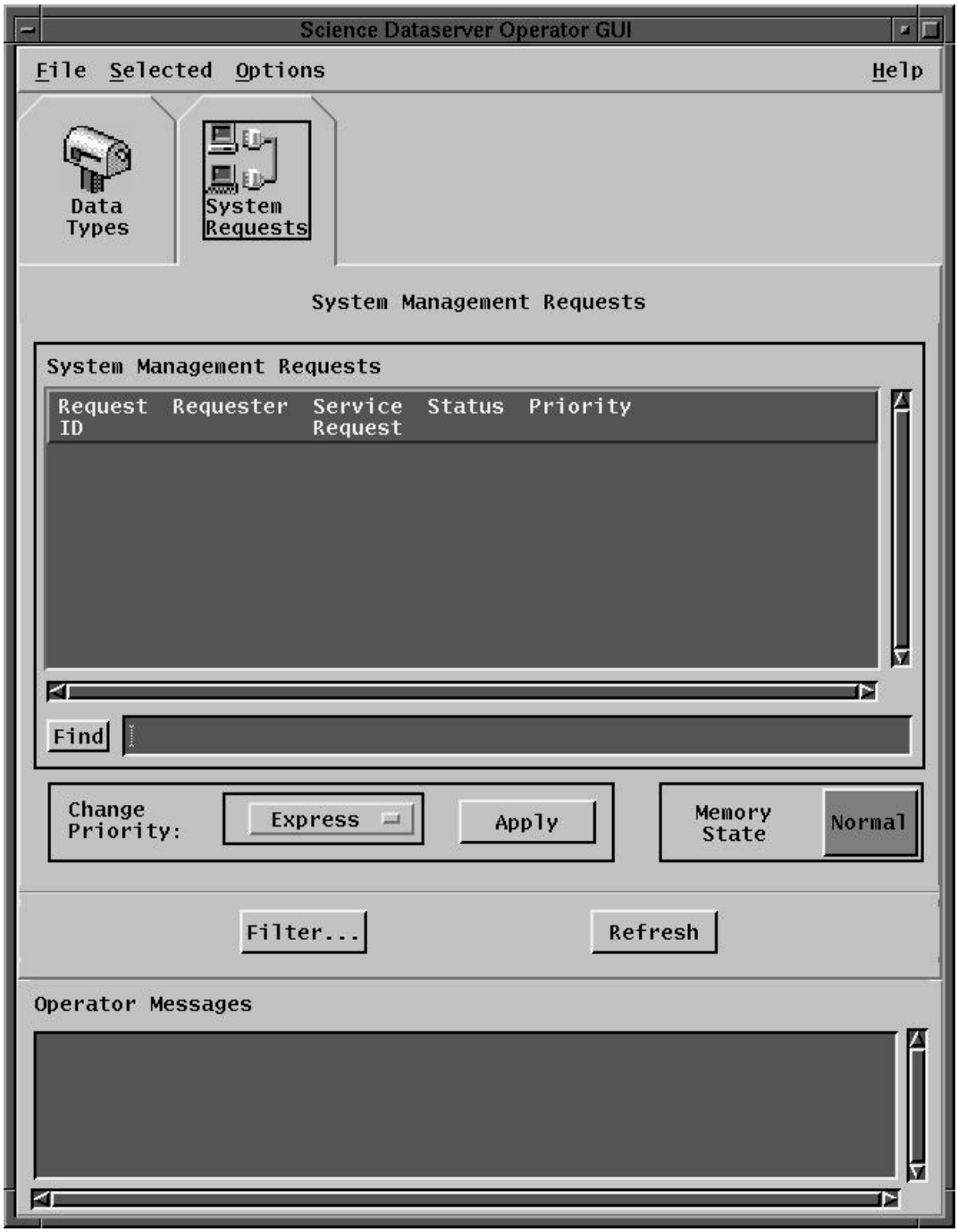


Figure 27. System Management Requests Window

Table 11, describes the System Management Requests Window fields.

Table 11. System Management Requests Field Description

Field Name	Data Type	Size	Entry	Description
Request ID	character	255	system generated	Unique identifier for the request.
Requester	variable character	100	system generated	Identifies the user that submitted the request.
Service Request	character	25	system generated	Types of requests handled are Insert, Acquire, and Delete
Status	character	20	system generated	Possible states are Submitted, Queued, Executing, Failed_Retryable, Failed_Fatal, Failed_Unknown and Done
Priority	variable character	20	system generated	Priority of the data server system requests, i.e., Express, Very High, High, Normal(default), Low.
Find	character	255	optional	If the list is too long, this field can be used to search for an entry

In addition, the following buttons are provided:

- **Change Priority:** changes the priority of each selected request through a pulldown menu. Possible values are: Express, Very High, High, Normal (default) and Low.
- **Apply** allows the operator to commit to the priority change selected through the change priority button.
- **Filter…** (see Figure 28) brings up the System Management Filter Requests dialog which provides a selection of attributes on which to filter for the list of System-wide requests. Filter on system management requests by entering the desired information, then clicking on the Request ID or Requester radio button for the desired attribute. Return to the original list of requests by clicking on the All Requests radio button. Click on other filters associated with State and Priority by clicking on the toggle button. Filter on every attribute associated with a category by clicking the **All** button or clear a category of filters by clicking on the **None** button.
- **Memory State:** it monitors the current memory state of the data server in regards to values that are set on the server side through configuration parameters. Possible values

are: Normal(green color), Low(yellow), Very Low(red). This functionality will only be visible if the server's DSSMEMORYMONITORDISABLEFLAG is off.

Figure 28. System Management Filter Requests Dialog

Table 12, describes the System Management Filter Requests Dialog fields.

Table 12. System Management Filter Requests Field Description

Field Name	Data Type	Size	Entry	Description
Request ID	**character**	**255**	**system generated**	**Unique identifier for the request.**
Requester	**variable character**	**100**	**system generated**	**Identifies the user that submitted the request.**

In addition, the following buttons are provided:

- **OK** implements filter criteria, and the dialog closes.
- **Apply** implements filter criteria, and the dialog remains open for additional filtering.

- **Cancel** closes the dialog without saving
- **Help** displays on-line help information.
- Back to the System Requests tab description (figure 27), **Operator Messages** provides informational and error messages to the user.
- **Refresh** causes the Data Server to be polled for an update on Requests.

Required Operating Environment

For information on the operating environment, tunable parameters, and environment variables refer to the 920-TDx-013 "Custom Code Configuration Parameters" documentation series . The "x" refers to the installed location, e.g. 920-TDG-013 is for GSFC DAAC.

Interfaces and Data Types

Table 13, lists the supporting products that this tool depends upon in order to function properly.

(Change needed to replace DCE)

Table 13. Interface Protocols

Product Dependency	Protocols Used	Comments
SDSRV and all clients	**DCE**	**via client libraries**
SDSRV GUIs	**X-11**	**via client libraries**

Databases

The Science Data Server Operator GUI does not include the direct managing of any database. It has an interface with the Science Data Server Data Base: however this interface is based on a simple parameter passing function. For further information of the Science Data Server Data Base refer to 311-CD-107-005, *Science Data Server Database Design and Schema Specifications for the ECS Project*.

Special Constraints

The Science Data Server Operator GUI runs only if the Science Data Server is running in the background. Note also that at the moment the Science Data Server GUI is started through a command line that specifies the configuration file that is used to initialize the GUI Application.

Outputs

There is no processing associated with the operation of this GUI. The information provided to the operator are retrieved from the Data Server Database described in the previous sections and displayed through the screens discussed therein.

Event and Error Messages

Both event and error messages are listed in Appendix A.

Reports

No reports are produced by this tool.

Browser to View ECS Science SDSRV Database

---

1 Connect to the **SDSRV (P0acs03)** database with login information as follows:

Server Name: **p0acg01_srvr**

User Name: **sdsrvApp**

Password: **welcome**

2 The Browser lets you view all the tables in the **SDSRV** database with the mode you have selected. Example**: <EcDsscienceDataserver1_TS1**> for the PVC.

Using isql commands in the SDSRV

---

1. **ssh p0acs03**
2. **Password:**
3. **Isql -Usdsrv_role -Pwelcome -p0acg01_srvr**
4. **1> use EcDsScienceDataserver1**
5. **2>go**
6. **3>select * from <choose table_name>**
7. **4>go**

---

wisqlite is an isql Browser to View ECS Data Bases

1. login: **ssh p0pls02**
2. **cd /tools/tcl/bin**
3. Execute: **wisqlite &**
4. User ID: **pdps_role**
5. PW: **welcome**
6. Select server_name by clicking on Server button p0pls01_srvr
7. Select: **Sign on**
8. Enter: **pdps, objects, tables**
9. **Select * from <table your choice>**

The following tables are useful to track down the problems in insert *.met:

1 DsDeDictionaryAttribute

2 DsMdAddtionalAttributes

3 DsMdCollections - View with ECS Assistant only

4 DsMdGranules - View with ECS Assistant only

Removing ESDT's from Archive Area using Command Line Scripts

Procedures:

1 **telnet** to (SDSRV) **p0acs03**[e.g.]

2 **login:, password**: <log into cmshared>

2 **cd /usr/ecs/TS1/CUSTOM/utilities**

3 Run script: **cleanesdt.csh <descriptor_file_name>**

- The ESDT will be removed from all four servers, SDSRV, SUBSRV, ADSRVR & DDICT. Check SDSRV Gui to see if removal was successful.

4 Now you have to log into each server and bring each of the four servers mentioned above, down by using the **KILL Script** and the **START Script** that have been specifically taylored for each server at the path identified below in each subsystem:

cd/home/cmshared/bin/

- **SDSRV you are already on, p0acs03, SBSRV is on p0ins01, ADSRV & DDIC are on p0ins02.**

Staging ESDT's for Installation

---

1 Copy into a directory from which the Science Data Server will use to bring ESDT's into the Science Data Server Data Base

2 Once you have the descriptor file copied into <your_directory> in the p0ins01 machine, you have to login into **p0acs03** and copy the file into **/usr/ecs/TS1/CUSTOM/data/ESS** directory. To do so, log into **p0acs03.** From the directory **/usr/ecs/TS1/CUSTOM/data/ESS,** Type command,
cp /home/your_directory/ESDTs/filename .
(Where /home/your_directory/ESDTs is the directory into which you stored the descriptor file.)

Now you have the copy of the latest descriptor file in the **ESS** directory.

Installing ESDT's using the Science Data Server GUI

Procedures:

1 **telnet** to (SDSRV) **p0acs03**[e.g.]

2 **login:, password**: <always log into **cmshared**> to enable use of isql commands and to bring servers up and down.

3 **cd /usr/ecs/TS1/CUSTOM/utilities**

4 If an ESDT has to be removed before replacing it, run script: **cleanesdt.csh <descriptor_file_name>**

- The ESDT will be removed from all four servers, SDSRV, SUBSRV, ADSRVR & DDICT. Check SDSRV Gui to see if removal was successful.

5 Now you have to log into each server and bring each of the four servers mentioned above, **down** by using the **KILL Script** and the **START Script** that have been specifically taylored for each server at the path identified below in each subsystem:

cd/home/cmshared/bin/

- **SDSRV is on, p0acs03, SBSRV is on p0ins01, ADSRV & DDICT are on p0ins02.**

6 Once you know that all the 4 Science Data Servers are down, then log into **p0acs03** machine, go to directory **/usr/ecs/TS1/CUSTOM/utilities** and execute command

EcDsScienceDataServerStart TS1 StartTemperature cold

(This script will bring up the four Data Servers.)

- Then check the status of the servers by using a Server Monitor GUI WHAZZUP. Once all the servers are up, then you can start installing the ESDT.

7 From **p0acs 03,** go to the directory, **/usr/ecs/TS1/CUSTOM/utilities,** type command:

EcDsSdSrvGuiStart TS1

The Science Data Server GUI will be brought up. You will see the list of already installed ESDTs there. To add new one, click on the **Add** button. You will get the **Add ESDT** window. This window is pointing to: **/usr/ecs/TS1/CUSTOM/data/ESS** where ESDT's have been staged. Click on the **File** button to select the descriptor file. Select the correct .desc file and click on **OK**. Enter ArchiveID as **DRP1_TS1.** Click on **OK**. Wait and check the message at the bottom of the window. You will see message – **Successfully added**…Click the **Refresh** button and you will see recently added ESDT into the list.

Re-install or Update ESDT's using the Science Data Server GUI

- Only the Science Data Server (SDSRV) needs to be brought down/up using the Kill and Start scripts before doing step 4 below.

<u>Procedures:</u>

1 **telnet** to (SDSRV) **p0acs03**[e.g.]

2 **login:, password**:

3 **cd /usr/ecs/TS1/CUSTOM/utilities**

(Log into the (SDSRV) using **Maintenance Mode.** This locks out other users while

The data base is being update with ESDT's.)

4 Execute command :

EcDsScienceDataServerStart TS1 StartTemperature maintenance

(This script will bring up the Science Data Server.)

- Then check the status of the SDSRV using a Server Monitor GUI **WHAZZUP**. Once all the servers are up, then you can start installing the ESDT.

5 From **p0acs 03,** go to the directory, **/usr/ecs/TS1/CUSTOM/utilities,** type command:

EcDsSdSrvGuiStart TS1

6 The Science Data Server GUI will be brought up. You will see the list of already installed ESDTs there. To **Update** a ESDT, click on the Update button. You will get the **Update ESDT** window.

- This window is pointing to: **/usr/ecs/TS1/CUSTOM/data/ESS** where ESDT's have been staged.

7 Click on the **File** button to select the descriptor file. Select the correct .desc file and click on **OK**. Enter ArchiveID as **DRP1_TS1.** Click on **OK**. (Wait and check the message at the bottom of the window. You will see message – **Successfully added**…Click the **Refresh** button and you will see recently added ESDT into the list.
Note: The (enter Archive ID as entry will go away with Release 6A).

This page intentionally left blank.

Production Rules

Purpose and Scope

This section describes the production rules supported by the Release B CDR design. It is intended to explain the implementation of these production rules in ECS and to document the information required for each rule. This section does not cover the exact design and implementation of these production rules in the PDPS system. That information may be found in the PDPS design documents, primarily in the Planning Subsystem document 305-CD-026-002, Release B SDPS Planning Subsystem Design Specification for the ECS Project.
While this section does address the syntax and operational procedures for specifying production rules, it does provide detailed information about what data is required for each rule and how that data will be used

Overview of Production Rules

Data Processing in ECS

Before discussing production rules, it is important to have a basic understanding of how the Earth Observing System Data and Information System (EOSDIS) Core System (ECS) conducts its data processing. ECS provides facilities for running product generation executives (PGEs) to produce science data products. The support for this function primarily resides in three subsystems: the Data Processing Subsystem (DPS) which actually executes the PGEs, the Planning Subsystem (PLS) which plans the execution of PGEs to efficiently utilize the local computing resources and the Data Server Subsystem (DSS) which provides PGEs with input data and archives output data. The design and implementation of the DPS and the PLS are carried out by the ECS organization known as the Planning and Data Processing System (PDPS).
Data Processing Subsystem (DPS) - The DPS provides the environment under which PGEs are run. It works with the SDP toolkit to interface with the science software. The DPS coordinates the acquisition of input data (staging) and the archiving of output data (de-staging) with the DSS. The DPS also ensures that there are adequate resources (disk space, RAM, etc.) available before a PGE begins execution. The DPS uses the AutoSys COTS product to implement the PLS production plan.
Planning Subsystem (PLS) - The PLS generally manages the ECS data processing activities and provides the main interface between the DPS and the rest of ECS. The PLS plans the execution of PGEs to provide efficient use of computing resources. The PLS receives. Production Requests (PRs) either from a software tool (Production Request Editor) or, for on-demand production requests (OPRs), from the DSS or DPS (see Error Handling). The PLS breaks up PRs into data processing requests (DPRs), which are individual runs of a PGE. It maintains information about input data for specific DPRs and passes that information to the DPS. It queries the DSS for data which meets required criteria and provides the universal references (URs) of that data to the DPS for acquisition.
Data Server Subsystem (DSS) - The DSS provides input data to the DPS and archives output data from the DPS. It responds to queries from the PLS and provides the PLS notification of the

arrival of input data for all subscriptions the PLS has placed with it. The DSS also submits On-Demand

Production Requests (OPRs) to the PLS.

These three subsystems form the core of the ECS data processing services. Since it has responsibility for managing these activities, most of the information about production rules will be maintained by the PLS.

General Definition of Production Rules Simply stated, production rules are the instructions about how a particular PGE is to be run.

These instructions specify a wide range of information such as the input and output data types, the frequency of execution, activation conditions and error handling instructions. This section focuses on the other types of production rules such as alternate inputs to be used or conditional PGE activation. These and other production rules are defined in the following sections.

Production rules in the ECS system will be tied to PGEs. This means that each PGE will use one or more production rules. The production rules will be initially entered by the Instrument Team with further modifications if necessary when a PGE undergoes Science Software Integration and Test (SSI&T) at the DAAC. Where applicable, default parameter values will be entered at that time. Many of these defaults can be overridden in the production environment when a Production Request is entered.

Production Rule Definitions

Introduction

This section presents a basic definition and examples of each of the production rules currently supported by the Release B CDR design. The production rules are divided into three categories:

Input Data Specification

Conditional Activation

Error Handling

Input Data Specification rules specify how the inputs for a particular run of a PGE are identified. Conditional Activation rules stipulate the conditions of when a particular PGE is to be run. Error Handling rules specify automatic actions to be taken when a PGE fails. Each sub-section is titled with the name of the production rule being discussed.

It is hoped that providing a common nomenclature will facilitate any future discussions between ITs and ECS about production rules. After each rule there is a list of the information required to implement that rule in the production environment (i.e. post-launch, at the DAACs). This information will initially be provided at SSI&T time. In the operational environment many of the values can be set when a Production Request is entered. Most production rule sections also include a simple diagram to help illustrate the concept being presented.

Input Data Specification

These production rules generally stipulate how inputs for a particular run of a PGE (a DPR) are identified/selected/created. Most PGEs have at least one input and one output. The most basic information about these inputs and outputs is the Earth Science Data Type (ESDT). It is assumed

in the examples in this chapter that the ESDT is specified for all data sets which are used or produced by a PGE.

Basic Temporal

The most basic type of production rule specifies, for each input ESDT, the temporal range of that ESDT required by the PGE.
For example, the run of a PGE which produces a particular 2.5 minute MODIS Level 1B data granule requires as input the specific 2.5 minute Level 1A granule which is contemporaneous with the Level 1B granule to be produced.
Another example would be the production of a CERES Level 1A granule from the Level 0 data. CERES L1A granules cover 24 hours and require the PLS to identify and the DPS to stage all of the Level 0 data files that cover a particular 24 hour period. This sort of simple temporal specification is basic to the system and will be used by every PGE.

Inputs required to implement the Basic Temporal rule

For each PGE which uses this rule, the following data must be provided for each ESDT (input or output): Time period - Length of time of ESDT (e.g. 12 hours, 1 week or one year).

Boundary - Date/time to start counting periods

Both of the above parameters are specified at SSI&T time.

Example use of the Basic Temporal Rule

Suppose there is a PGE which creates a daily (24 hour) data set and those data sets should correspond with calendar days. In that case Time Period = 24 hours and Boundary = 1/1/98 00:00 (this could be any date). When a Production Request is entered, the Planning Subsystem will scan backwards from the start time of the PR until it finds the start of a period. It then produces DPRs for each period of time covered by the PR. Table 14, illustrates how this would work. Given the period and boundary defined above, two PRs are entered, the first for the time between 6/3/99 12:00 - 6/7/99 6:00 while the second specifies 6/7/99 18:00 - 6/16/99 12:00. When the first PR (#1) is entered, the Planning Subsystem determines that the start time of the PR is not the start time of a period. It then scans backwards, finding the start time of the period to be 6/3/99 00:00 and creates a DPR (1.1) to process that period. It then continues producing DPRs (1.2, 1.3, 1.4 and 1.5) until it reaches the end of the period which contains the end time of the PR. When the second PR (#2) is entered, the same process takes place. Note that since the end time of PR #1 and the start time of PR #2 fall in the same period, duplicate DPRs (1.5 and 2.1) are created. This sort of duplication is checked for by the Planning Subsystem and the second DPR is not run.

Table 14. Basic Temporal

Boundary (1/1/98 00:00)
= Period (24 Hrs)
PR #1 PR #2
DPR 1.1
DPR 1.2
DPR 1.3
DPR 1.4
DPR 1.5 = DPR 2.1
DPR 2.2
DPR 2.3
DPR 2.10
DPR 2.9
6/1/99 6/23/99

Production_Rules_Syntax

The Production Rules Syntax are presented as part of the Powerpoint Slides that accompany this document.

Production Rules identified thus far are listed as follows:

Basic Temporal, Advanced Temporal, Period Specification, Alternate Inputs, Optional Inputs, Metadata-based Activation, Metadata-Based Query -Static, Metadata-based Query - Dynamic, Orbit-Based Activation, Orbit Path, Runtime Parameter, Multi-File Granules, Multi-Granule ESDT's, Spatial Query, Minimum Number of Granules, Land Tiling, Tiling with Metadata-based Query, Optional DPRs, Ocean Data Day, Most Recent Granule, Alternates based on Minimum number of Granules, Zonal Tiling, Tile Clustering and Smart_Start_of_Year.

Production Requests

Science Software and Production Requests

Science software is one of the keys to production planning and processing:

Performs the actual data processing to create desired products.

Is developed at Science Computing Facilities (SCFs) external to ECS.

Is embodied in Product Generation Executives (PGEs) when the software is integrated into the ECS production processing environment.

- PGEs are science software code (e.g., executable programs or shell scripts) that contain the instructions for processing data to create the desired products.

The production request (PR) is another key to production planning and processing. The Production Planner defines ECS science data processing in terms of PRs.

A PR is an order for data to be produced by the Data Processing Subsystem.

A single PR may specify several jobs (using the same PGE) that are to be run over a period of time or a single job producing a single set of data.

PRs may apply to the processing of new data (standard PRs or standing orders) or the reprocessing of existing data (reprocessing PRs).

Each PR identifies a specific PGE for generating a particular type of product.

- Some PGEs are dependent on others; i.e., some PGEs require input data that are the output of other PGEs.
- The planning software will recognize and reject a PR when the PR specifies a PGE that requires data from another PGE that has not yet been specified in a PR.

The Planning Subsystem performs the following functions:

Uses each PR to generate either one or a series of Data Processing Requests (DPRs).

- Each DPR corresponds to one execution of a single PGE.
- Each DPR contains the information that is needed by the SDP processing function, including PGE-related information.

Checks the availability of the data required for the DPR, either from the data server (if the data have been previously ingested) or from internal predictions (if the data are expected to arrive in the future).

Determines what data will be included in the DPR output so the system can make predictions concerning the availability of data for subsequent PGEs.

Figure 29 shows the relationships among the PGEs, PRs, and DPRs as they are accessed through the Production Request Editor GUI.

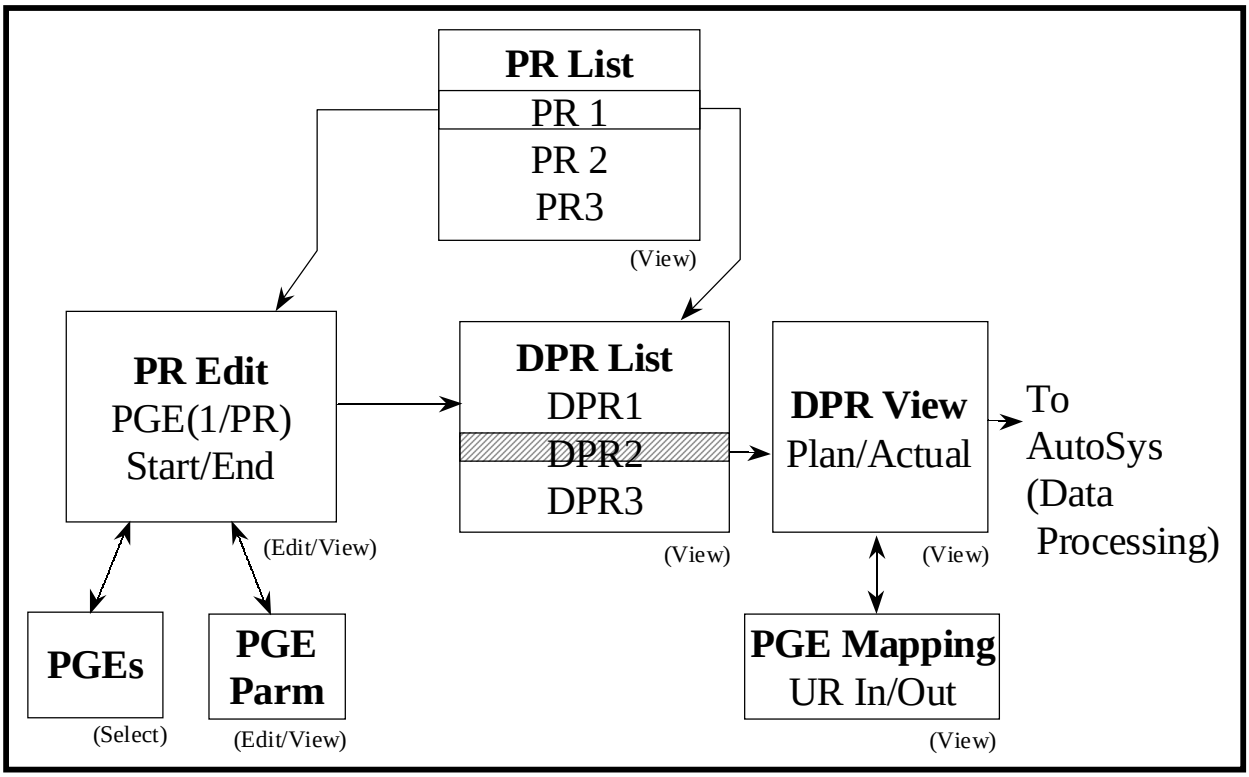


Figure 29. Production Request Editor Flow

Types of Processing

ECS either accommodates or will accommodate the following three general types of data processing:

Routine Processing

Reprocessing

Ad-Hoc Reprocessing

On-Demand Processing

Routine processing is pre-defined software production processing that is periodic and keyed to data arrival. For example, every day a Production Planner includes in the daily schedule a DPR for generating a particular Level 1A product from the most recent Level 0 data from the applicable satellite instrument.

Reprocessing typically involves using a new, improved PGE to process data that had previously been processed with an older version of the PGE. In such cases reprocessing would be a large-

scale operation, especially if several years worth of data were to be reprocessed. Consequently, the Production Planner is likely to schedule reprocessing in manageable quantities so the processing resources can accommodate routine and on-demand processing in addition to the reprocessing.

In addition, **ad-hoc reprocessing** could be necessary at any time. For example, if a product fails a quality assurance (QA) check, the same PGE could be run again on the same data set in the hope of creating an acceptable product. Similarly, if processing of a PGE fails for some reason, it might be possible to rerun the PGE and hopefully achieve a successful outcome.

On-demand processing is ad-hoc processing initiated by either the Planning Subsystem or an end-user (as opposed to the Production Planner). For example, a researcher using data from the Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) instrument on the Terra satellite may need a particular Level 2 product that has not yet been generated. The ASTER researcher would submit an on-demand request to have the product generated from a Level 1B product stored in the archive.

In the future such on-demand processing requests (OPRs) will be entered from a Client Subsystem tool, passed through the Distributed Information Manager (Data Management Subsystem) and the Data Server to the Planning Subsystem. Currently there is a work-around to the automated process which requires the requester to contact DAAC personnel to make the request. So far ASTER researchers are the only identified external users of automated on-demand processing.

Another future feature is automated cross-DAAC planning. It is a process that will be undertaken when products produced at one DAAC require inputs being produced at another DAAC. The predicted production time of remote input products will be used in creating local production plans. The primary mechanism for cross-DAAC planning will be the use of Predicted Data Availability Schedules (PDAS), created when a plan is created. A DAAC's PDAS will be made available to remote DAACs via the Data Server.

Production Rules Describe a PGE to ECS

Production rules are the instructions about how a particular PGE is to be run. The instructions specify a wide range of information such as the input and output data types, the frequency of execution, activation conditions and error handling instructions.

A single PGE may use one or more sets of production rules, known as PGE profiles, since it may be desirable to run the same PGE with different input data sets, or activation conditions. The production rules are entered when a PGE undergoes Science Software Integration and Test (SSI&T) at the DAAC. Where applicable, default parameter values are entered at that time. The initially selected runtime parameters, metadata check parameters, tile IDs, and many of the default parameters can be overridden in the production environment when a Production Request is entered.

Production rules define the PGE to the Planning and Data Processing Subsystems (PDPS). The following types of conditions can be specified for each PGE:

The time period for which the PGE will run.

- A PGE can run every hour, every day, or for every orbit of a satellite. The frequency of how often a PGE runs must be defined to PDPS so that it knows when to plan and execute the PGE. A definition of a satellite's orbit could be included if the PGE were to be executed for some number of satellite passes.

PGE Inputs.

- A PGE can have any number of inputs. The types of the inputs and how frequently they are available helps determine on what basis the PGE is scheduled.
- Most inputs to a PGE are retrieved based on time; the specified inputs are retrieved from the Data Server Subsystem for the time which the PGE is defined to execute. Production Rules allow other conditions to be added to the mix, such as checks or queries against the metadata of the input granules, or the lists of inputs as alternates (for when a primary input is not available) or optionals (for inputs without which the PGE can still run successfully). If inputs are defined as alternate or optional, the number of inputs staged for the execution of the PGE may vary from one run to the next.

PGE Outputs.

- A PGE can have any number of outputs. The characteristics of the outputs can have effects on any downstream PGEs that use them as inputs. For example, it is possible for an output to be defined as optional, in which case it may or may not even be produced. (When an output is not generated, it cannot be used as input for a downstream PGE.)

Runtime Parameter Values.

- A PGE can have any number of runtime parameters, which are values that are placed in the process control file (PCF) under specified logical IDs before the PGE executes. The PGE treats them as constants and normally they are set either during SSI&T or when the Production Request is entered.
- For some production rules (such as Orbital Processing) there is specific information that can be placed in a runtime parameter if so desired by the PGE.

Geographic Tiles.

- A PGE can define a geographic location for which it will process data. Tiles are defined through production rules, and change the staging of inputs from time-based, to a combination of time- and geographically-based. Data are retrieved based on their location on the Earth with respect to the tile that it is currently being processed.

Some (but not all) production rules can work with other production rules.
Production rules are often used for the selection of dynamic inputs.

Dynamic inputs can be either internal or external.

- **Dynamic internal** inputs are produced by other PGEs (they are called dynamic internal inputs because they are produced at an ECS DAAC).
- **Dynamic external** inputs are periodically ingested and stored in the Data Server Subsystem (they are termed dynamic external inputs because they are produced outside of the DAAC).

Static inputs are granules that are inserted during the SSI&T process and are retrieved not on the basis of time but by Earth Science Data Type (ESDT) and science group.

- The Metadata Query Production Rule is the only production rule that works for choosing static inputs.

PGE profiles allow a PGE to be defined to PDPS multiple times, each with a different set of inputs, outputs, or even scheduling information. Each PGE's definition is made up of its name, its version and its profile number. Different PGE name/version pairs define different PGEs to PDPS. The addition of the profile allows for multiple definitions of a PGE name/version pair. There can be up to 999 profiles for each PGE.

Syntax of Production Rules

Production rules are defined in the following two ways:

Through science metadata that is entered in various types of files during the SSI&T process.

By entering parameter values when a Production Request is created to schedule the PGE.

During SSI&T, production rules are defined in files written in Object Description Language (ODL) in a parameter equals value format. There are three general categories of ODL files:

PGE Science Metadata ODL Files.

ESDT Science Metadata ODL Files.

Production Rule-Specific Science Metadata ODL Files

- Orbit Definition ODL Files.
- Path Map Definition ODL Files.
- Tile Definition ODL Files.

When a Production Request is created to schedule a PGE, it is necessary to enter certain information that is essential to implementing the production rules that affect the particular PGE. The information may concern the date and time-range

PGE Science Metadata ODL Files

The PGE science metadata ODL file defines a PGE (or at least the current plan for its operation) to PDPS. It specifies everything from the PGE name and version, to the period for the PGE (how often it runs), all inputs and outputs, any runtime parameters and any exit messages or dependencies. A template version of the PGE science metadata ODL file is created by the **SSIT Create ODL Template** program from a PCF from the PGE.

ESDT Science Metadata ODL Files

The ESDT science metadata ODL file defines a PGE input or output to PDPS. Each input and output of a PGE must have a corresponding ESDT science metadata ODL file defined. It describes everything that PDPS needs to know about the subject input or output file, from its name and version, to its period (how often data is collected), to where it is used and archived. Note that many PGEs can use the same input or output type, and thus can share the same ESDT science metadata ODL file.

Unlike the PGE science metadata ODL file, there is no tool for automatically generating a template ESDT science metadata ODL file. A template version exists under the data directory called ESDT_ODL.template. The template must be copied to a file that follows the naming convention *ESDTShortName#ESDTVersionID*.odl.

Production Rule-Specific Science Metadata ODL Files

The production rule-specific science metadata ODL files provide specific information to PDPS about production rules used by a PGE. They are needed only when the PGE is subject to one of the following conditions:

Is executed on the basis of a satellite orbit.

Needs to know the orbital path of a satellite.

Requires data based on geographic tiling of the Earth.

Since not every PGE is based on orbits or tiles, not all PGEs require these files. The comments in the PGE_ODL.template describe when setting a specific parameter means that a production rule-specific science metadata ODL file needs to be created.

The production rule-specific science metadata ODL files are broken into three types, which are defined as follows:

Orbit ODL File.

- Defines the orbital period of the satellite from which the PGE's input data is created.
- Defines when a given orbit starts, how long it lasts, and the number of the orbit.
- PDPS uses the information in the orbit ODL file to extrapolate future orbits and is able to plan PGEs that are required to run every so many orbits of the satellite.

Pathmap ODL File.

- Defines the mapping between the cyclic 0-233 orbits that the satellite makes with the actual path number that the PGE requires.
- PDPS computes the path number from the orbit number (specified in the orbit ODL file) by incrementing it until it reaches the 233 maximum, then resetting it to zero.

- Many instruments expect the path number to be a fixed swath on the Earth, so it is not just incremented for each satellite pass.
- The pathmap ODL file creates a mapping from the sequential path numbers to the path numbers expected by the PGEs.

Tile ODL File.

- Defines the coordinates of the tiles used by some instruments to specify geographic locations on the Earth.
- The tile definitions are used by PDPS to schedule the PGE (one execution per tile) and to acquire the necessary data (using the geographic coordinates to acquire data for the tile being processed only).

Unlike the PGE science metadata ODL file, there is no tool to automatically generate a template production rule-specific science metadata ODL file. Because the files themselves tend to be small, this is not usually a problem. A template version of each kind of production rule-specific science metadata ODL file (e.g., ORBIT_ODL.template, TILE_ODL.template) exists in the /usr/ecs/<MODE>/CUSTOM/data directory on the AIT Workstation. The templates must be copied, named properly, and edited in order to create the appropriate production rule-specific science metadata ODL file.

Release 5 Production Rules

The following statements provide some simplified descriptions of production rules that are scheduled to be made available in Release 5:

Basic Temporal - Temporal (time) range of inputs matches the temporal range of outputs.

Advanced Temporal - Temporal range of inputs is offset from the expected temporal range of inputs and outputs.

Alternate Input - PGE is run with different inputs based on the availability of various alternate input data sets.

Optional Input - PGE is run with specified optional inputs if available; otherwise, PGE is run without them.

Minimum/Maximum Number of Granules - Minimum number of input granules needed for full data coverage and maximum number of input granules to search for may be specified. Minimum and maximum number of outputs expected from the PGE may be specified.

Optional DPRs – The only DPRs executed are those for which the non-routine key input data actually become available (i.e., are either produced in data processing or can be acquired from the archive).

Intermittent Activation - Every n^{th} DPR is activated; all other DPRs are skipped.

Metadata Checks - DPR is run only if input data's metadata value(s) meet(s) certain criteria.

Metadata Query - Input granule selection is based on metadata value.

Data Day - Input data selection is based on Data Day.

Spatial Query - Input granule selection is based on the spatial coverage of another input (i.e., the key input).

Tiling - Input data is chosen on the basis of Instrument Team-defined tiles (geographic areas).

Closest Granule – DPR is generated if a required input granule within a particular time range (rather than an exact time) is available; otherwise, no DPR is generated. (Supersedes the Most Recent Granule Production Rule)

Orbital Processing - Selection of input times is based on orbit information.

Basic Temporal Production Rule

The Basic Temporal Production Rule defines the timeframe for the PGE along with its input and output data. PGEs subject to the Basic Temporal Production Rule generally have the following characteristics in common:

Typically scheduled to run using input data that become available periodically (every hour, every day, etc.).

Use input data for a particular period of time.

Produce output for a specified length of time.

The data the PGE takes in (its input) and the data it produces (its output) have the same period (or some subset of the same period) as the PGE.

Example One:

- A MODIS PGE processes data for five-minute intervals, producing Level 1B granules.
- The PGE requires as input the specific five-minute Level 1A granule that is contemporaneous with (covers the same five-minute time period as) the Level 1B granule to be produced.
- Using the Basic Temporal Production Rule, a five-minute Level 1A granule is staged as input to the PGE and a five-minute Level 1B granule is expected as output, both matching the timeframe for which the PGE is run.

Example Two:

- A CERES PGE processes data for 24-hour intervals, producing 24-hour Level 1A granules as output.
- As input the PGE takes Level 0 data that is ingested every two hours.
- Using the Basic Temporal Production Rule, twelve two-hour Level 0 granules are staged as input to the PGE and a 24-hour Level 1A granule is expected as output, matching the timeframe for which the PGE is run.

The fundamental elements used to define the Basic Temporal Production Rule are “period: and “boundary.”

Period is the length of time for which a PGE processes data or the length of time for which input and output data is collected.

- A PGE that is subject to the Basic Temporal Production Rule only and that processes data in two-hour blocks, takes in data that relates to a particular two-hour interval and produces output data for that same two-hour period.
- Data that has a period of 15 minutes was collected or produced for a 15-minute time period.

Boundary is the starting point for the data or PGE. Depending on the characteristics of the data or PGE, the boundary may be the start of a minute or hour or day or week (etc.).

- If a PGE's boundary is the start of the hour, it processes data that starts every hour and runs on data for the length of its period.
- If data comes in every day, PDPS predicts that the data is going to be available at the start of the day and allows scheduling of PGEs that use the data as input accordingly.

Both the PGE itself and the input data have a boundary and period associated with them. That is how PDPS determines the frequency of processing for a Basic Temporal PGE and the time period for its inputs and outputs.

PDPS uses **period** and **boundary** in combination to plan the processing of each PGE, including determining its input requirements and anticipated output (which may be input to other PGEs). If a PGE has a period of one hour and a boundary of “start of day,” it is scheduled every hour, beginning at midnight. If an input has a period of 15 minutes and boundary of “start of hour,” PDPS predicts it every 15 minutes beginning on the hour.

Boundary offset is an addition to the Basic Temporal Production Rule that allows a PGE or data to start on an offset from a given boundary. For example, if a PGE would normally run every day but not start until two or three hours into the day (e.g., beginning at 3:00 a.m. instead of midnight), a boundary offset can be used to add three hours to the “start of day” boundary. This would mean the PGE would run on data that occurred three hours after the boundary.

Data with offset times refers to data where the start time is a few minutes off of the start time that PDPS expects. For example: if data is defined to PDPS as follows:

```
BOUNDARY = "START_OF_HOUR"
PERIOD   = "HOURS=1"
```

but the data actually starts at 1:05 and ends at 2:05, the data is said to have **offset times**. There is a flag in the production rule syntax that tells PDPS to shift granule time specifications to match the granules in the archive.

The **end-of-month anomaly** is an addition to the Basic Temporal Production Rule that allows a PGE or data to cover a specific number of days within a month. The month is broken into thirds. The first third is composed of the first 10 days of the month. The second third consists of days 11 through 20. And the last third varies in length depending on the total number of days in the

month (i.e., for November it would have 10 days; for December it would have 11 days). A specific **boundary** and **period** allow a PGE or its data to be scheduled into thirds of a month.

Figure 30 provides an illustration of the Basic Temporal Production Rule. The PGE has a boundary of "start of day" and a period of one hour, so it is scheduled for every hour through the day. If a Production Request were entered for two full days of processing, a DPR would be created for the PGE to run every hour; i.e., 48 DPRs total. If a Production Request were created for a four-hour period in the middle of a single day (for example, from 12:00 noon to 4:00 p.m.), then four DPRs would be created, one for 12:00-1:00, one for 1:00-2:00, one for 2:00-3:00, and one for 3:00-4:00.

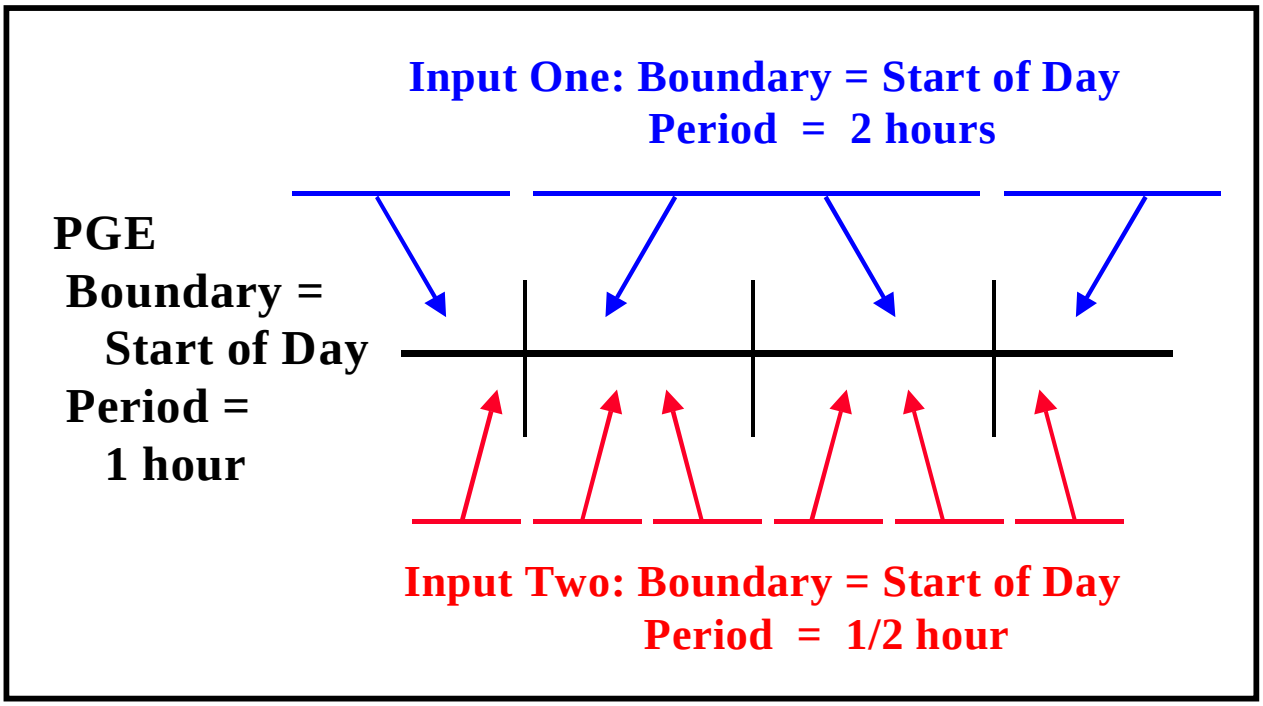


Figure 30. Example of the Basic Temporal Production Rule

In the example (Figure 30), Input One has a boundary of "start of day" and a period of two hours, so when PDPS plans for its availability, it expects a granule every two hours beginning at midnight. Consequently, each granule of Input One is associated with two DPRs for the PGE, because the PGE encompasses only one hour of the two-hour granule's period.

Input Two has a boundary of "start of day" and a period of ½ hour, so when PDPS plans for its availability, it expects a granule every ½ hour beginning at midnight. As a result two granules of Input Two are associated with each DPR for the PGE, because the PGE encompasses an hour of the ½-hour granule's Period. Thus, every DPR of the PGE will wait for two granules of Input Two to arrive before it can be processed.

PGE Science Metadata ODL File Parameters

The following parameters must be set properly in the applicable PGE science metadata ODL file in order to implement the Basic Temporal Production Rule:

SCHEDULE_TYPE.

PROCESSING_PERIOD.

PROCESSING_BOUNDARY.

The SCHEDULE_TYPE parameter specifies the type of scheduling that will be done for the PGE. The following values are applicable to the Basic Temporal Production Rule:

"Time"

- The PGE is scheduled on the basis of the specified boundary/period and the availability of data for that boundary/period.

"Snapshot"

- The PGE is scheduled for a single date/time.
- Note that PROCESSING_PERIOD and PROCESSING_BOUNDARY are **not** needed when "Snapshot" is specified.

Other values for SCHEDULE_TYPE apply to other production rules, such as the following values:

“Data"

- The PGE is scheduled on the basis of the availability of data produced by other PGEs.

"Tile"

- The PGE is scheduled based on the definition of geographic tiles.

"Orbit"

- PGE scheduling is based on the orbit of the spacecraft.

The PROCESSING_PERIOD parameter describes the length of time for which the PGE executes. Data will be acquired (barring any combination of Production Rules) for the specified period and output data will be planned for the given period. It is of the format "<Period Type>=<Length of Period>". Note that “length of period” can be specified as a positive integer only. The following values are acceptable “period type” entries for the Basic Temporal Production Rule:

"YEARS"

- PGE processes data applicable to a given year or years.
- "YEARS" might be specified for a PGE that computes a yearly average.
- For example, PROCESSING_PERIOD = "YEARS=1" relates to a PGE that processes one year’s worth of data.

"MONTHS"

- PGE processes data applicable to a particular month or several months.
- “MONTHS” is most likely to be used for some kind of averaging PGE.
- For example, PROCESSING_PERIOD = "MONTHS=2" relates to a PGE that processes two months’ worth of data at a time.

"THIRDS"

- PGE processes data applicable to some number of thirds of the month.
- For example, PROCESSING_PERIOD = "THIRDS=1" relates to a PGE that processes data applicable to 1/3 of the month.

"WEEKS"

- PGE processes data applicable to some number of weeks.
- For example, PROCESSING_PERIOD = "WEEKS=2" relates to a PGE that processes two weeks’ worth of data every time it runs.

"DAYS"

- PGE processes data applicable to some number of days.
- For example, PROCESSING_PERIOD = "DAYS=5" relates to a PGE that processes five days’ worth of data.

"HOURS"

- PGE processes data applicable to some number of hours.
- For example, PROCESSING_PERIOD = "HOURS=4" relates to a PGE that processes four hours’ worth of data when it is executed.

"MINS"

- PGE processes data applicable to some number of minutes.
- For example, PROCESSING_PERIOD = "MINS=5" relates to a PGE that processes five minutes’ worth of data.

"SECS"

- PGE processes data applicable to some number of seconds.
- For example, PROCESSING_PERIOD = "SECS=2" relates to a PGE that runs on two seconds’ worth of data.

There are other types of values for PROCESSING_PERIOD but they apply to other production rules (as described in the applicable sections of the lesson).

The PROCESSING_BOUNDARY parameter specifies the boundary (starting point in time) of the PGE. It tells when each instance of the PGE should start. Note that the

PROCESSING_BOUNDARY and PROCESSING_PERIOD are used in conjunction to schedule the PGE.

The following PROCESSING_BOUNDARY values are used for implementing the Basic Temporal Production Rule:

"START_OF_HOUR" – PGE processes data for each hourly interval.

"START_OF_6HOUR" - PGE processes data for every 6-hour interval.

"START_OF_DAY" - PGE processes data for every daily interval.

"START_OF_WEEK" - PGE processes data for every weekly interval.

"START_OF_ONE_THIRD_MONTH" - PGE processes data for every 1/3 of a month.

"START_OF_MONTH" - PGE processes data for every monthly interval.

"START_OF_YEAR" - PGE processes data for every yearly interval.

"START_DATE=*DD/MM/YYYY*" - PGE processes data for the specified date only.

There are other values for PROCESSING_BOUNDARY that apply to other production rules (as described in the applicable sections of the lesson).

Handling Data with Offset Times

When the ALIGN_DPR_TIME_WITH_INPUT_TIME flag is set to "Y" (i.e., ALIGN_DPR_TIME_WITH_INPUT_TIME = “Y”) PDPS shifts the expected times for input data to the actual times found in the archive. If the flag is NOT set, data with offset times can cause problems when generating Production Requests.

ESDT Science Metadata ODL File Parameters

The following parameters must be set properly in the applicable ESDT science metadata ODL file in order to implement the Basic Temporal Production Rule:

DYNAMIC_FLAG.

PERIOD.

BOUNDARY.

The DYNAMIC_FLAG describes the type of data that is defined in the ESDT science metadata ODL file. It specifies to PDPS what kind of data the PGE requires as input or produces as output. It can have any of the following four possible values, all of which are valid for Basic Temporal data:

"S"

- Static Data.
- Data do not change at regular intervals.
- The same granule can be used as input for many runs of the PGE.
- Calibration files are a good example of static data.

"I"

- Dynamic Internal.
- Data are produced by a PGE running at the local DAAC.
- All output products are either “dynamic internal” or “interim” kinds of data.

"E"

- Dynamic External.
- Data are produced by an external source (not a PGE running at the local DAAC).
- EDOS data is a primary example.
- Dynamic external can be set for PGE inputs only.

"T"

- Interim/Intermediate.
- Data are stored only temporarily by the Data Server Subsystem.

The PERIOD parameter specifies the length of time covered by the data. Data are expected to be either ingested or produced for the length of the PROCESSING_PERIOD described in PGE science metadata ODL files. However, the PERIOD of the data does **not** have to match the PROCESSING_PERIOD defined for the PGE. PDPS plans for data where the ESDT period is less or more than the processing period of the PGE that uses it. For example, if the PGE PROCESSING_PERIOD = "HOURS=1" and the input data PERIOD = "MINS=5", then PDPS plans to acquire twelve granules of the input data to cover the PROCESSING_PERIOD.

The following “period type” values are used for implementing the Basic Temporal Production Rule:

"YEARS"

- Data span a year or years.
- “YEARS” might be selected for a yearly average output product.
- For example, PERIOD = "YEARS=1" specifies data that cover a period of a year.

"MONTHS"

- Data span a month or several months.
- “MONTHS” is most likely used for some kind of averaging output product.
- For example, PERIOD = "MONTHS=2" specifies data that cover a period of two months.

"THIRDS"

- Data span some number of thirds of a month.
- For example, PERIOD = "THIRDS=1" specifies data that cover a period of 1/3 month.

"WEEKS"

- Data span some number of weeks.
- For example, PERIOD = "WEEKS=2" specifies data that cover a period of two weeks.

"DAYS"

- Data span some number of days.
- For example, PERIOD = "DAYS=5" specifies data that cover a period of five days.

"HOURS"

- Data span some number of hours.
- For example, PERIOD = "HOURS=4" specifies data that cover a period of four hours.

"MINS"

- Data span some number of minutes.
- For example, PERIOD = "MINS=5" specifies data that cover a period of five minutes.

"SECS"

- Data span some number of seconds.
- For example, PERIOD = "SECS=2" specifies data that cover a period of two seconds.

"ORBITS"

- Data span some number of orbits of the spacecraft.
- For example, PERIOD = "ORBITS=1" specifies data that cover one orbit.
- A PGE can be time-scheduled (using the Basic Temporal Production Rule) but use orbit-based data.

The BOUNDARY parameter is the starting point in time of the data granule. It tells when each data granule should start. Note that the BOUNDARY and PERIOD are used in conjunction to determine the starting and ending time for the granules.
The following values for BOUNDARY apply to the Basic Temporal Production Rule:

"START_OF_HOUR"

- Data granules start every hour.

"START_OF_6HOUR"

- Data granules start every six hours.

"START_OF_DAY"

- Data granules start every day.

"START_OF_WEEK"

- Data granules start every week.

"START_OF_ONE_THIRD_MONTH"

- Data granules start every 1/3 of a month.

"START_OF_MONTH"

- Data granules start every month.

"START_OF_YEAR"

- Data granules start every year.

"START_OF_ORBIT"

- Data granules start every orbit.

Advanced Temporal Production Rule

The Advanced Temporal Production Rule allows for input data to be acquired for a time period other than that of the PGE or its planned inputs/outputs. It provides an offset mechanism, specifying on an input basis that the data required for processing is some number of seconds earlier or later than the planned time period for the PGE.

Example One:

- A PGE requires data from its previous execution for interpolation purposes (e.g., one of its inputs is the output of the very same PGE the last time that it ran).
- If the PGE processes data for each one-hour interval (producing an hourly product), the Advanced Temporal Production Rule is specified with an offset of minus 3600 seconds (one hour) for the input of the ESDT produced by previous runs.

Example Two:

- A PGE takes as input two-hour Level 0 data to produce an L1A product.
- Because the edges of the Level 0 data can be difficult to process without preceding and succeeding data, the PGE requires three Level 0 granules, one

from the time period before it runs, one for the time period it is currently processing and one for the next time period.

- The PGE is defined as having three inputs, the first with and Advanced Temporal offset of minus 7200 seconds (two hours), the second with no Advanced Temporal offset and the third with an Advanced Temporal offset of plus 7200 seconds (two hours).

The Advanced Temporal Production Rule uses the times specified in the Basic Temporal Production Rule as a reference point for specifying offset(s) to request data from a “period” and/or “boundary” different from that of the DPR or its input. The offsets are specified as either negative or positive numbers to indicate whether the time period of the input data is before or after that of the DPR (a particular run of a PGE).

Begin Period Offset is an amount of time (in seconds) that is specified with respect to the DPR start time. A negative beginning offset requests data that was collected before the DPR start time. A positive beginning offset requests data with a collection time after the start time of the DPR.

End Period Offset is an amount of time (in seconds) that is specified with respect to the DPR end time. A negative ending offset requests data that ended collection before the DPR end time was reached. A positive ending offset requests data that ended collection after the end time of the DPR boundaries.

Note that the beginning and ending offsets are not absolute cut-offs for data. Overlapping granules (granules that start or end outside of the offsets) will be staged as inputs to the DPR.

Figure 31 provides an illustration of the Advanced Temporal Production Rule. The PGE shown in the example processes data for every one-hour interval. However, Input One comes in at two-hour intervals and Input Two is produced every 1/2 hour.

Both the Begin Period Offset and End Period Offset for Input One are -7200 seconds (minus two hours). Consequently, every DPR will stage the "previous" Input One. This could be used to get the "previous" or "next" granule of an input.

The Begin Period Offset for Input Two is zero, meaning that it will match the Start Time of the DPR. The End Period Offset is +1800 seconds (plus 1/2 hour). Therefore, all Input Two granules will be staged that fall within the time period of the DPR plus 1/2 hour. The effect is to acquire all Input Two granules within the time period of the DPR, plus the one from the next 1/2-hour time period, for a total of three granules. The additional granule acquired by means of the End Period Offset might be used for interpolation purposes at the end point.

The same types of parameter settings that apply to the Basic Temporal Production Rule apply to the Advanced Temporal Production Rule. In addition, there are some parameters in the PGE science metadata ODL file that apply specifically to the Advanced Temporal Production Rule. However, the values applicable to the Basic Temporal

Production Rule must be set before the Advanced Temporal Production Rule syntax is added.

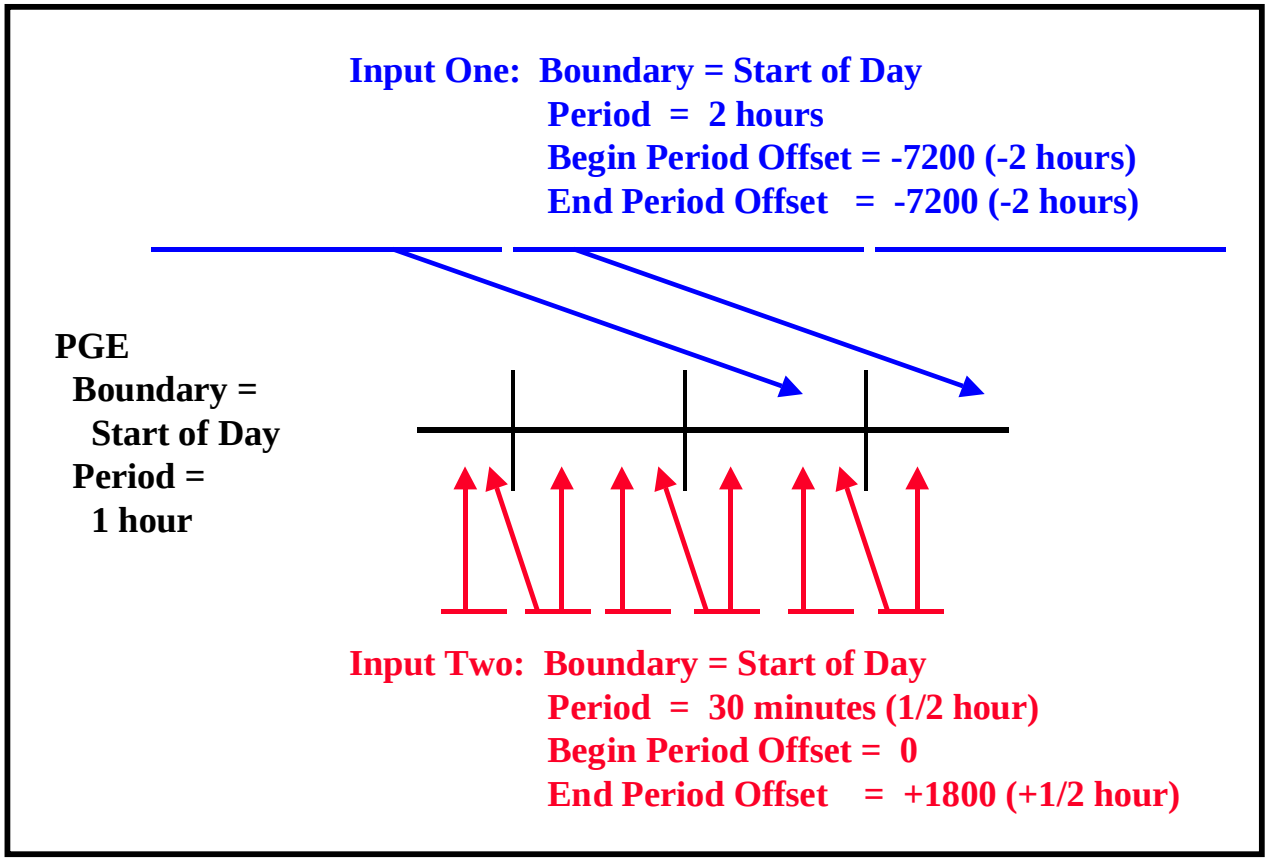


Figure 31. Example of the Advanced Temporal Production Rule

PGE Science Metadata ODL File Parameters

During the SSI&T process the PGE science metadata ODL file is generated from the PCF delivered with the science algorithm. A PCF_ENTRY object is generated for each file entry in the PCF. In order to implement the Advanced Temporal Production Rule the PCF_ENTRY object for each type of input file to which the rule applies uses the following syntax:

```
OBJECT = PCF_ENTRY
    .
    .
    .
    BEGIN_PERIOD_OFFSET =
    END_PERIOD_OFFSET =
    .
    .
    .
END_OBJECT = PCF_ENTRY
```

Accordingly, the following parameters must be set properly in order to implement the Advanced Temporal Production Rule:

BEGIN_PERIOD_OFFSET.

END_PERIOD_OFFSET.

BEGIN_PERIOD_OFFSET is the offset added to or subtracted from the Data Start Time of the DPR. The value assigned to BEGIN_PERIOD_OFFSET can be either a positive or negative value, specified in seconds. If the value is positive, it is added to the Data Collection Start Time (looking for the input forward in time). If the value is negative, it is subtracted from the Data Collection Start Time (looking backward in time). For example, BEGIN_PERIOD_OFFSET = -3600 requests data that was collected one hour (3600 seconds) before the DPR start time.

END_PERIOD_OFFSET is the offset added to or subtracted from the Data Collection End Time of the DPR. The value assigned to END_PERIOD_OFFSET can be either a positive or negative value, specified in seconds. If the value is positive, it is added to the Data Collection End Time (looking for the input forward in time). If the value is negative, it is subtracted from the Data Collection End Time (looking backward in time). For example, END_PERIOD_OFFSET = +2700 requests data that was collected 45 minutes (2700 seconds) after the DPR end time.

The BEGIN_PERIOD_OFFSET and END_PERIOD_OFFSET parameters can be specified for any input PCF_ENTRY in the PGE science metadata ODL file. If not specified, the parameters are set to zero (0) and the Advanced Temporal Production Rule does not apply to the PGE.

Alternate Input and Optional Input Production Rules

The Alternate Input and Optional Input Production Rules are very similar and use much the same processing in PDPS. Both rules allow a PGE to select various inputs based on timers and priority lists. The major difference is that Alternate Inputs requires that one of alternates on the list be used, whereas Optional Inputs allows successful execution of the PGE if no optional input on the list is available.

The Alternate Input Production Rule allows for a PGE to evaluate a list of inputs in priority order and be scheduled and executed with the best priority input that could be found. In essence, a PGE using Alternate Inputs is saying "I would like to run with Input A, but if it's not available, I am willing to run with Input B." A timer can be used to specify how long to wait for a given alternate choice before proceeding with a choice of lesser priority. The PGE is not executed until one of the alternate choices has been found.

Example:

- A PGE requires model wind data as an input but is capable of accepting wind data from a Data Assimilation Office (DAO) model, a National Centers for Environmental Prediction (NCEP) model, or (as a last resort) climatology.
- The PGE would use the Alternate Input Production Rule to list each input in priority order, giving a timer value for how long to wait before trying the next input.
- If the DAO data are most desirable, DAO would be listed as first choice or "primary" data.

- NCEP would be the second choice.
- Climatology would be the last choice.
- If a timer value is specified for DAO data, the PGE will wait for that timer to expire before running with either NCEP data or climatology.
- If a timer had been placed on the NCEP input, the PGE would wait before running with the climatology data.

The Optional Input Production Rule allows for a PGE to list inputs that are desired but not required for it to execute. The inputs are ranked as previously stated and timers are set to wait before choosing a lower-priority type of input. However, if none of the inputs on the list becomes available, the PGE starts because the alternatives are classified as "optional." In essence the PGE is saying "I would like to run with Input A, but if its not available, I can run (and produce reasonable output) without it."

Example:

- It would be preferable to run a particular MODIS PGE with the output of a MISR PGE as input.
- However, the MISR output may not be produced every day.
- So the MODIS PGE lists the MISR input as optional with a two-hour timer.
- On those occasions when no MISR output is produced, the MODIS PGE waits for two hours and then is executed without the MISR input.

Figure 32 provides an illustration of the Alternate Input Production Rule. The PGE in the illustration has two inputs that are “required” so they must available for the PGE to be run. It also has one input that is “alternate.” The alternate input can be one of three choices, the first choice is the **primary**, then there are second and third choices.

After the pair of required inputs has become available, the alternate inputs are evaluated as follows:

If the primary alternate is available, it is used as input and the PGE is scheduled for execution.

There is a one-hour timer on the primary alternate. If the primary alternate is unavailable, the PGE waits until the primary alternate becomes available or the one-hour timer expires, whichever occurs first.

If the second alternate is available after the timer for the primary alternate has expired, the second alternate is used as input and the PGE is scheduled for execution.

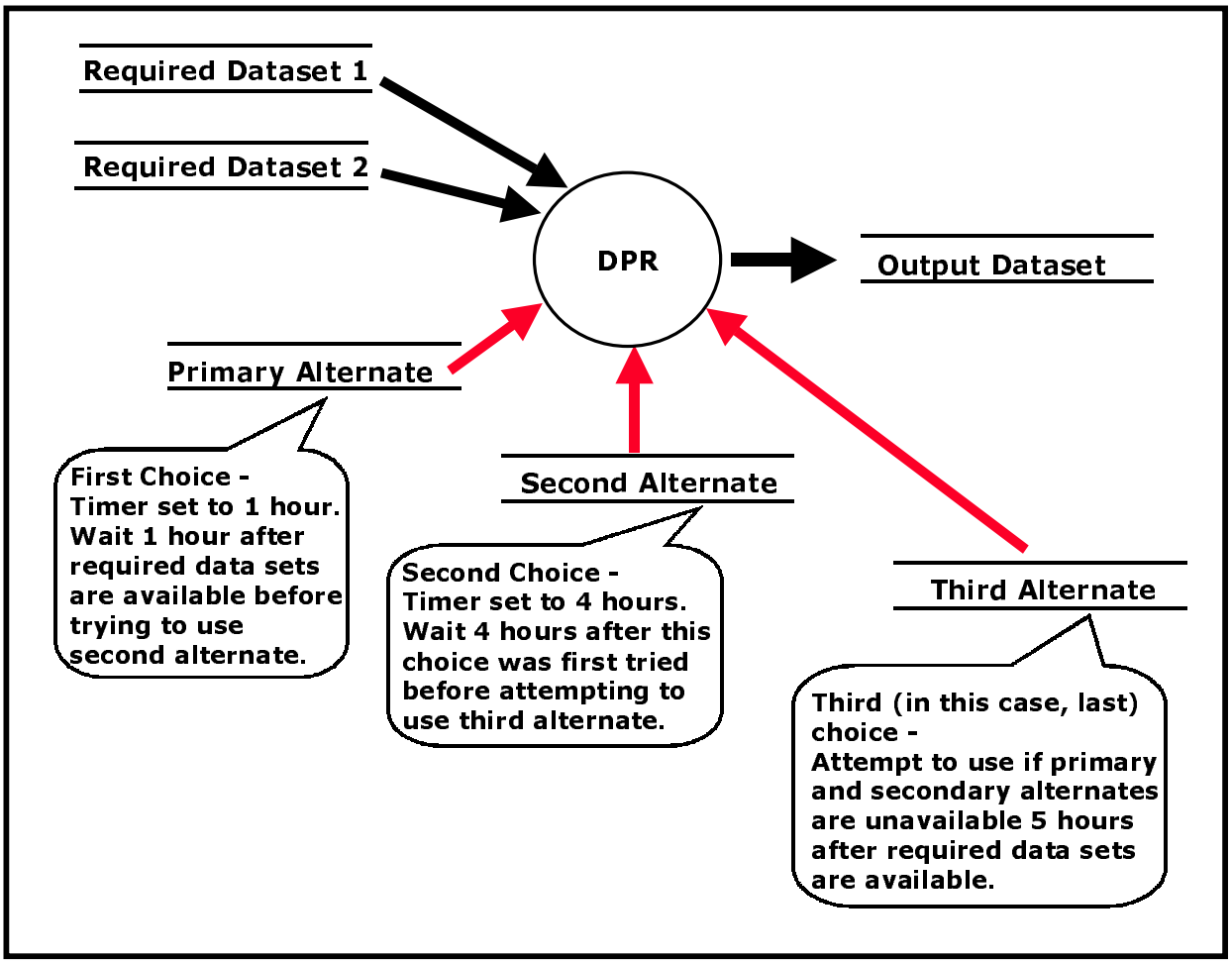


Figure 32. Example of the Alternate Input Production Rule

There is a four-hour timer on the second alternate. If the second alternate is unavailable, the PGE waits until either the primary alternate or the secondary alternate becomes available or the four-hour timer expires, whichever occurs first.

If the third alternate is available after the timer for the second alternate has expired, the third alternate is used and the PGE is scheduled for execution.

There is no timer on the third alternate. If the third alternate is not available, the PGE waits until either the primary alternate, the secondary alternate, or the third alternate becomes available, whichever occurs first.

The PGE will not start processing until one of the alternates becomes available.

If instead of an alternate the third input for the PGE had been defined as an optional input, the preceding scenario would have been the same, except that if neither the primary alternate, the second alternate nor the third option was available after the timers had expired, the PGE would

not wait; it would be scheduled for execution without the third input. It would run with the two required inputs only.

The Alternate Input and Optional Input Production Rules are additions to settings/syntax put into the ODL files for other production rules. Inputs deemed “optional” or “alternate” can be searched for and acquired by other production rules (e.g., Basic Temporal or Metadata Checks/Query). The syntax for the rules used to search for the inputs have to be filled out in addition to the syntax required to make the input an alternate or optional input.

PGE Science Metadata ODL File Parameters

The following parameter must be set properly in the applicable PGE science metadata ODL file in order to implement the Alternate Input or Optional Input Production Rule:

INPUT_TYPE.

In addition, one of the following two ODL objects is used within a PCF_ENTRY to define either the Alternate Input Production Rule or the Optional Input Production Rule:

ALTERNATE_INPUT object.

OPTIONAL_INPUT object.

INPUT_TYPE is a type of data defined by a PCF_ENTRY object (i.e., between OBJECT = PCF_ENTRY and END_OBJECT = PCF_ENTRY). It can have one of four possible values, only three of which are used to define an alternate or optional inputs:

"Required"

- A required input.
- The data must be available or the PGE does not execute.
- It is the "normal" value for the parameter (i.e., INPUT_TYPE = “Required”); consequently, the input is neither an alternate input nor an optional input.

"Primary"

- The primary alternate input.
- The data is the first choice in a list of alternates.

"Alternate"

- An alternate input (except the primary alternate) in a list of alternates.
- The data is not the first choice in a list of alternates; it is a subsequent choice if the primary (or a higher-priority alternate) is not available.

"Optional"

- An optional input.
- Availability of the data will be checked and if a timer has been specified, execution of the PGE will wait.
- The PGE can be executed without the data if it is not available.

Although the Alternate Input and Optional Input Production Rules are similar, there are two different ODL objects used to define them within a PCF_ENTRY; i.e., the ALTERNATE_INPUT object and the OPTIONAL_INPUT object.
The ALTERNATE_INPUT object has the following syntax:

```
OBJECT = PCF_ENTRY
    .
    .
    .
    .
    .
    OBJECT = ALTERNATE_INPUT
        .
        .
        .
    END_OBJECT = ALTERNATE_INPUT
END_OBJECT = PCF_ENTRY
```

The ALTERNATE_INPUT ODL object surrounds an Alternate Input definition. An OBJECT/END_OBJECT pair separates the parameters defining the Alternate Input from the rest of the parameters defining the PCF_ENTRY. The following parameters define an ALTERNATE_INPUT object:

CLASS.

CATEGORY.

ORDER.

RUNTIME_PARM_ID.

TIMER.

WAITFOR.

TEMPORAL [not implemented].

CLASS is a simple counter used to differentiate the different ALTERNATE_INPUT objects within the file. Since each ALTERNATE_INPUT object resides within a different PCF_ENTRY object, the CLASS for an ALTERNATE_INPUT object can always be 1.
CATEGORY is the name of the list of alternates to which the ALTERNATE_INPUT belongs. The PDPS uses CATEGORY to associate different alternates within a list. CATEGORY can be set to any string value of 20 characters or less (e.g., CATEGORY = "Snow Ice"). Alternates that are part of the same list should have matching CATEGORY values.
ORDER is the numerical place that the particular alternate holds in the list of alternates. The first choice or Primary Alternate (with the INPUT_TYPE = "Primary") should have ORDER = 1.
RUNTIME_PARM_ID specifies the Logical ID (in the PCF) for which the PGE will find the Logical ID of the alternate chosen. Since all alternates must be contained within different PCF_ENTRY objects, they all must have different Logical IDs (but all alternates within the same CATEGORY should have the same value of RUNTIME_PARM_ID). The

RUNTIME_PARM_ID parameter specifies the Logical ID of a runtime parameter that the PGE may read to find out which alternate was chosen for the particular execution of the PGE.
The TIMER parameter specifies how long to wait for the particular alternate before checking for the next alternate in the list. The parameter value is expressed in the format "<Period Type>=<Length of Period>". Note that “Length of Period” can be specified as a positive integer only. The Alternate Input Production Rule accepts the following “Period Type” values:

"WEEKS"

- PDPS should wait for some number of weeks before searching for the next alternate in the list.
- For example, TIMER = "WEEKS=2" would make PDPS wait two weeks before checking for the next alternate input.

"DAYS"

- PDPS should wait for some number of days before searching for the next alternate in the list.
- For example, TIMER = "DAYS=5" would make PDPS wait five days before checking for the next alternate input.

"HOURS"

- PDPS should wait for some number of hours before searching for the next alternate in the list.
- For example, TIMER ="HOURS=4" would make PDPS wait four hours before checking for the next alternate input.

"MINS"

- PDPS should wait for some number of minutes before searching for the next alternate in the list.
- For example, TIMER = "MINS=5" would make PDPS wait five minutes before checking for the next alternate input.

"SECS"

- PDPS should wait for some number of seconds before searching for the next alternate in the list.
- For example, TIMER = "SECS=2" would make PDPS wait two seconds before checking for the next alternate input.

The WAITFOR parameter specifies whether or not the PGE can be run without the alternate input. Setting WAITFOR = "N" means that the PGE can run without the input if it cannot be found. In a list of alternate inputs, this would have meaning for the last choice only. If WAITFOR = "Y", the PGE is not executed (even after the last alternate timer expires) until one of the alternates in the list can be found.

The TEMPORAL parameter is an unimplemented feature that would allow for searching for alternates from the same time period but a different date. It is currently stored in the PDPS database but is not used.

The OPTIONAL_INPUT object has the following syntax:

```
OBJECT = PCF_ENTRY
    .
    .
    .
    .
    .
    OBJECT = OPTIONAL_INPUT
        .
        .
        .
    END_OBJECT = OPTIONAL_INPUT
END_OBJECT = PCF_ENTRY
```

The OPTIONAL_INPUT ODL object surrounds an Optional Input definition. An OBJECT/END_OBJECT pair separates the parameters defining the Optional Input from the rest of the parameters defining the PCF_ENTRY. The following parameters define an OPTIONAL_INPUT object:

CLASS.

CATEGORY.

ORDER.

RUNTIME_PARM_ID.

TIMER.

TEMPORAL [not implemented].

The parameters that apply to the Optional Input Production Rule are defined in the same way that the corresponding parameters are defined for the Alternate Input Production Rule. However, note that the Optional Input Production Rule has no WAITFOR parameter. It is irrelevant; in fact, the very essence of the Optional Input Production Rule depends on not “waiting for” the last option but going ahead with the execution of the PGE without the unavailable optional input(s).

Table 15. Extract of PGE Metadata ODL File Template Showing Alternate Inputs

```
>OBJECT = PCF_ENTRY
>   CLASS = 16
>     LOGICAL_ID = 1500
>   PCF_FILE_TYPE = 1
>   DATA_TYPE_NAME = "MOD10L2G"        [MODIS Level 2G Snow Cover]
```

Table 15. Extract of PGE Metadata ODL File Template Showing Alternate Inputs

```
>   DATA_TYPE_VERSION = "1"                [ESDT versioning in release B.0]
>   DATA_TYPE_REQUIREMENT = 1
>   SCIENCE_GROUP = ""
>     OBJECT = FILETYPE
>         CLASS = 1
>     FILETYPE_NAME = "Single File Granule"
>   END_OBJECT = FILETYPE
>   INPUT_TYPE = "Primary"
>   NUMBER_NEEDED = 1
>   OBJECT = ALTERNATE_INPUT
>         CLASS = 1
>         CATEGORY = "Snow Ice"                  set [undefined]
>         ORDER = 1                              [This data type is sought first]
>       RUNTIME_PARM_ID = 1509                   [Run-time parameter holds LID of
alternate]
>       IMTER = "HOURS=6"
>        AWTFOR = "N"                            [Force time-out on wait]
>       EMTPORAL = "N"                           [Use most currently produced]
>   END_OBJECT = ALTERNATE_INPUT
>END_OBJECT = PCF_ENTRY
>
>OBJECT = PCF_ENTRY
>   CLASS = 17
>     LOGICAL_ID = 1501
>   PCF_FILE_TYPE = 1
>   DATA_TYPE_NAME = "MOD10A1"             [MODIS Level 3 Daily Gridded Snow
Cover data set]
>   DATA_TYPE_VERSION = "1"
>   DATA_TYPE_REQUIREMENT = 1
>   SCIENCE_GROUP = ""
>     OBJECT = FILETYPE
>         CLASS = 2
>     FILETYPE_NAME = "Single File Granule"
>   END_OBJECT = FILETYPE
>   INPUT_TYPE = "Alternate"
```

Table 15. Extract of PGE Metadata ODL File Template Showing Alternate Inputs

```
>   OBJECT = ALTERNATE_INPUT
>         ACCESS = 2
>         CATEGORY = "Snow Ice"                    [Ice defined]
>         ORDER = 2                                [This data type is sought last]
>       RUNTIME_PARM_ID = 1509                     [Run-time parameter holds LID of
alternate]
>       MTER = "HOURS=6"                           [Wait 6 additional hours]
>       AWFOR = "N"
>       EMPORAL = "N"
>   END_OBJECT = ALTERNATE_INPUT
>END_OBJECT = PCF_ENTRY
>
>OBJECT = PCF_ENTRY
>   CLASS = 18
>     LOGICAL_ID = 1502
>   PCF_FILE_TYPE = 1
>    DATA_TYPE_NAME = "MIANTASC"          [MISR Terrestrial Atmosphere and
Surface Climatology]
>   DATA_TYPE_VERSION = "1"
>   DATA_TYPE_REQUIREMENT = 1
>   SCIENCE_GROUP = ""
>     OBJECT = FILETYPE
>         ACCESS = 3
>     FILETYPE_NAME = "Single File Granule"
>   END_OBJECT = FILETYPE
>   INPUT_TYPE = "Alternate"
>   OBJECT = ALTERNATE_INPUT
>         ACCESS = 3
>         CATEGORY = "Snow Ice"                    [Ice defined]
>         ORDER = 3                                [This data type is sought last]
>       RUNTIME_PARM_ID = 1509                     [Run-time parameter holds LID of
alternate]
>       IMTER = ""                                 [Do not wait for this one]
>       AWFOR = "Y"                                [Start anyway]
>       EMPORAL = "N"
```

Table 15. Extract of PGE Metadata ODL File Template Showing Alternate Inputs

```
>   END_OBJECT = ALTERNATE_INPUT
>END_OBJECT = PCF_ENTRY
```

Minimum/Maximum Number of Granules Production Rule

The Minimum/Maximum Number of Granules Production Rule makes it possible to specify a range of possible granules for a given input or output for a PGE.

Inputs.

- Minimum number of granules the PGE needs for full data coverage.
- Maximum number of granules for the time period.

Outputs.

- Minimum number of outputs that the PGE is expected to produce.
- Maximum number of outputs that the PGE is expected to produce.

For example, a PGE processes data for every 90-minute interval, has a period of 90 minutes, and takes as input a granule with a period of two hours.

In many instances one granule of the input will satisfy the PGE.

In other instances, because of the way the two-hour and 90-minute periods overlap, the PGE needs two input granules to cover the time period.

Therefore,…

- Minimum Number of Granules = 1.
- Maximum Number of Granules = 2.

The Minimum/Maximum Number of Granules Production Rule is different from most production rules because it works for both input and output granules. It allows the PGE to request of a range of inputs (i.e., 1-10 granules), so that it runs with as few as one granule but with as many as ten granules. If a PGE needs at least three granules of a particular input, the minimum number of granules is defined as three and the PGE is not executed until at least three granules are available.

Optional outputs are defined when the Minimum Number of Granules is set to zero. In such cases the PGE can produce none of the particular type of output and still be considered to have executed successfully. If a PGE has a non-zero value for a Minimum Number of Granules associated with an output, and fails to produce any granules of that output type, it is marked as failed.

Figure 33 provides an illustration of the Minimum/Maximum Number of Granules Production Rule. In the example the PGE processes data related to a one-hour period and takes in both

Input 1 and Input 2. Since Input 1 has a PERIOD of 1/2 hour, every PGE run requires two Input 1 granules. Input 2 has a PERIOD of 15 minutes, so there are four Input 2 granules for every PGE run.

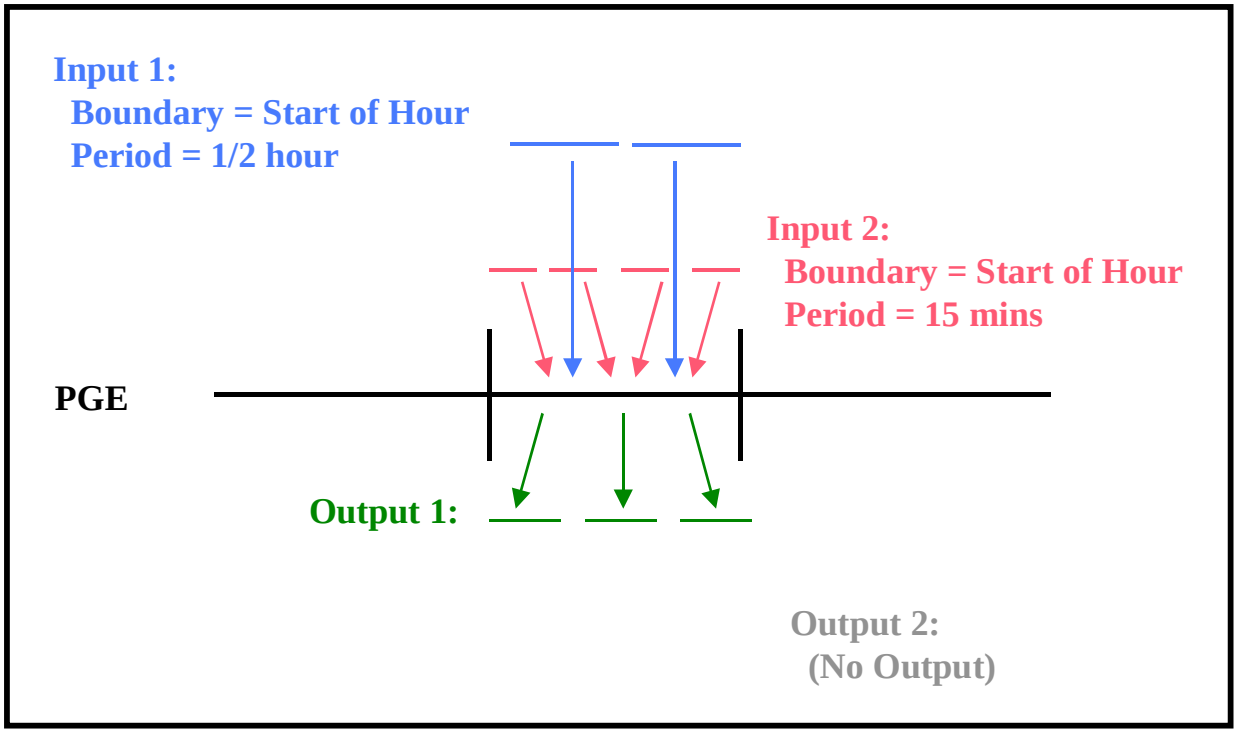


Figure 33. Example of the Minimum/Maximum Number of Granules Production Rule

The PGE produces three Output 1 granules for each run. In this case it does not produce any Output 2 granules.
Minimum and maximum values can affect each input and output as follows:

Input 1:

- If Minimum Granules is set to anything equal to or less than two for Input 1, the PGE is scheduled and executed.
- If Minimum Granules is set to three, the PGE is not scheduled because there are not enough Input 1 granules to make the minimum.
- If Maximum Granules is set to anything equal to or greater than two for Input 1, the PGE is scheduled and executed.
- If Maximum Granules is set to one, the PGE is not scheduled because there are too many Input 1 granules (the number exceeds the maximum that the PGE can process).

Input 2:

- If the Minimum Granules is set to anything equal to or less than four for Input 2, the PGE is scheduled and executed.
- If Minimum Granules is set to five, the PGE is not scheduled because there are not enough Input 2 granules to make the minimum.
- If Maximum Granules is set to anything equal to or greater than four for Input 2, the PGE is scheduled and executed.
- If Maximum Granules is set to three, the PGE is not scheduled because there are too many Input 2 granules (the number exceeds the maximum that the PGE can process).

Output 1:

- If Minimum Granules is set to anything equal to or less than three for Output 1, the PGE is scheduled and executes successfully.
- If Minimum Granules is set to four, the PGE is marked as failed because it did not produce the expected number of output granules.
- If Maximum Granules is set to anything equal to or greater than three for Output 1, the PGE is scheduled and executes successfully.
- If Maximum Granules is set to two, the PGE is marked as failed because it produced too many output granules.

Output 2:

- If Minimum Granules is set to anything other than zero, the PGE is marked as failed because it did not produce the expected number of output granules.
- If Maximum Granules is set to anything equal to or greater than zero for Output 2, the PGE is scheduled and executes successfully.

The Minimum/Maximum Granules Production Rules are additions to settings/syntax put into the ODL files for other production rules. All Production Rules have a Minimum and Maximum Granule setting for both inputs and outputs, even though both values may be set to one (1).

PGE Science Metadata ODL File Parameters

The PGE science metadata ODL file syntax for implementing the Minimum/Maximum Production Rule for **input** data includes the following types of entries:

```
OBJECT = PCF_ENTRY
    .
    PCF_FILE_TYPE =
    .
    .
    MIN_GRANULES_REQUIRED =
```

```
        MAX_GRANULES_REQUIRED =
        .
        .
        .
    END_OBJECT = PCF_ENTRY
```

Accordingly, the following parameters must be set properly in order to implement the Minimum/Maximum Production Rule:

PCF_FILE_TYPE.

MIN_GRANULES_REQUIRED.

MAX_GRANULES_REQUIRED.

The PCF_FILE_TYPE parameter is defined by integers in the range of 1 to 8 (inclusive). The integers are codes for the following types of files:

1 - product input files.

2 - product output files.

3 - support input files.

4 - support output files.

5 - user defined runtime parameters.

6 - interim/intermediate input files.

7 - interim/intermediate output files.

8 - temporary input/output.

For inputs (any PCF_ENTRY with a PCF_FILE_TYPE equal to 1, 3 or 6) the following pair of values must be set for each PCF_ENTRY:

MIN_GRANULES_REQUIRED

- Minimum number of granules required for the input.
- A value of zero (MIN_GRANULES_REQUIRED = 0) would mean that the PGE could execute if no granules for that particular input could be found (in effect, the input is an **optional input**).
- A value of three (for example) would mean that the PGE must have at least three granules of the input before the PGE can be executed.

MAX_GRANULES_REQUIRED

- Maximum number of granules for the input that the PGE is able to successfully process.
- A value of four (for example) would mean that the PGE would process at most four granules for the input.

- If MAX_GRANULES_REQUIRED = 4 and more than four granules are found for the given input, the PGE is not executed.

The PGE science metadata ODL file syntax for implementing the Minimum/Maximum Production Rule for **output** data includes the following types of entries:

```
OBJECT = PCF_ENTRY
    .
    PCF_FILE_TYPE =
    .
    .
    MIN_GRANULE_YIELD =
    MAX_GRANULE_YIELD =
    .
    .
    .
END_OBJECT = PCF_ENTRY
```

For outputs (any PCF_ENTRY with a PCF_FILE_TYPE equal to 2, 4 or 7) the following pair of values must be set for each PCF_ENTRY.

MIN_GRANULE_YIELD

- Minimum number of granules that the PGE produces for the output.
- A value of zero (MIN_GRANULE_YIELD = 0) means that the PGE produces no granules for the output (the output is an **optional output**).
- A value of three (for example) means that the PGE produces at least three granules of the output during a successful execution.

MAX_GRANULE_YIELD

- Maximum number of granules that the PGE produces for this output.
- A value of four (for example) means that at most the PGE produces four granules for the output.
- Note that sizing of disk space is based on this number, so making it too small could cause problems on the science processor disks.

Optional DPRs Production Rule

The Optional DPRs Production Rule (also called the Data-Scheduled Production Rule) makes the execution of a PGE subject to the availability of a **key input**. The system generates DPRs for every possible instance of the key input data but executes only the DPRs for which data are either produced in data processing or can be acquired from the archive.

The Optional DPRs Production Rule applies to PGEs that process certain kinds of **non-routine data**.

Routine Data

- Data that can be predicted, that come in at specific intervals and are always of a specified length.
- Routine data makes it possible for the Basic Temporal Production Rule to schedule PGEs based on their input data.

Non-Routine Data

- Data that cannot be predicted because they come in at random periods and/or their length is variable.
- Examples include an "optional" output of an upstream PGE, or data that are archived at random periods (e.g., some forms of ASTER data).

An Optional DPR has as its **key input** a non-routine data type. There are two sets of circumstances that lead to the scheduling of Optional DPRs:

Every possible time that the input is produced in data processing (i.e., the key input is produced as an "optional" output by an upstream PGE).

Whenever a new granule (of a particular data type) can be acquired from the archive (e.g., archived data that were inserted at unpredictable times).

An example of the first condition starts with a MODIS PGE that produces a certain product only when the input data were collected during the satellite's "Day" mode. A second MODIS PGE is scheduled to use the optional ("Day"-mode) product from the first MODIS PGE as its key input. The second MODIS PGE is scheduled to run after every instance of the first MODIS PGE; however, only the DPRs that can use the optional products resulting from runs of the first MODIS PGE are executed. The remaining DPRs cannot be executed because there is no input data for them.

The second condition is illustrated by ASTER routine processing, which makes use of the Optional DPRs Production Rule to schedule and execute ASTER PGEs for new data that have been archived. (Note that the DAAC ingests and archives ASTER production data from tapes supplied by the ASTER Ground Data System on a frequent but not entirely predictable basis.) When the Production Planner creates a Production Request for an ASTER PGE, it is necessary to specify the **insertion time** range (i.e., the time period when the desired data were archived) as opposed to the **collection time** (when the satellite instrument gathered the data). DPRs specifying the ASTER PGE are scheduled and executed for the data granules that were actually inserted in the archive during the time period specified in the Production Request.

An illustration of the Optional DPRs production rule is presented in Figure 34. In the figure there are two DPRs (i.e., DPR-1 and DPR-2) for the upstream PGE and two DPRs (i.e., OPT-1 and OPT-2) for the PGE subject to the Optional DPRs Production Rule. The "Optional DPRs" PGE takes as input the optional output of the upstream PGE. When it is executed, DPR-1 produces the optional output, so the dependent DPR (OPT-1) is executed. However, OPT-2 is not executed because DPR-2 (on which OPT-2 depends) does not produce the optional output.

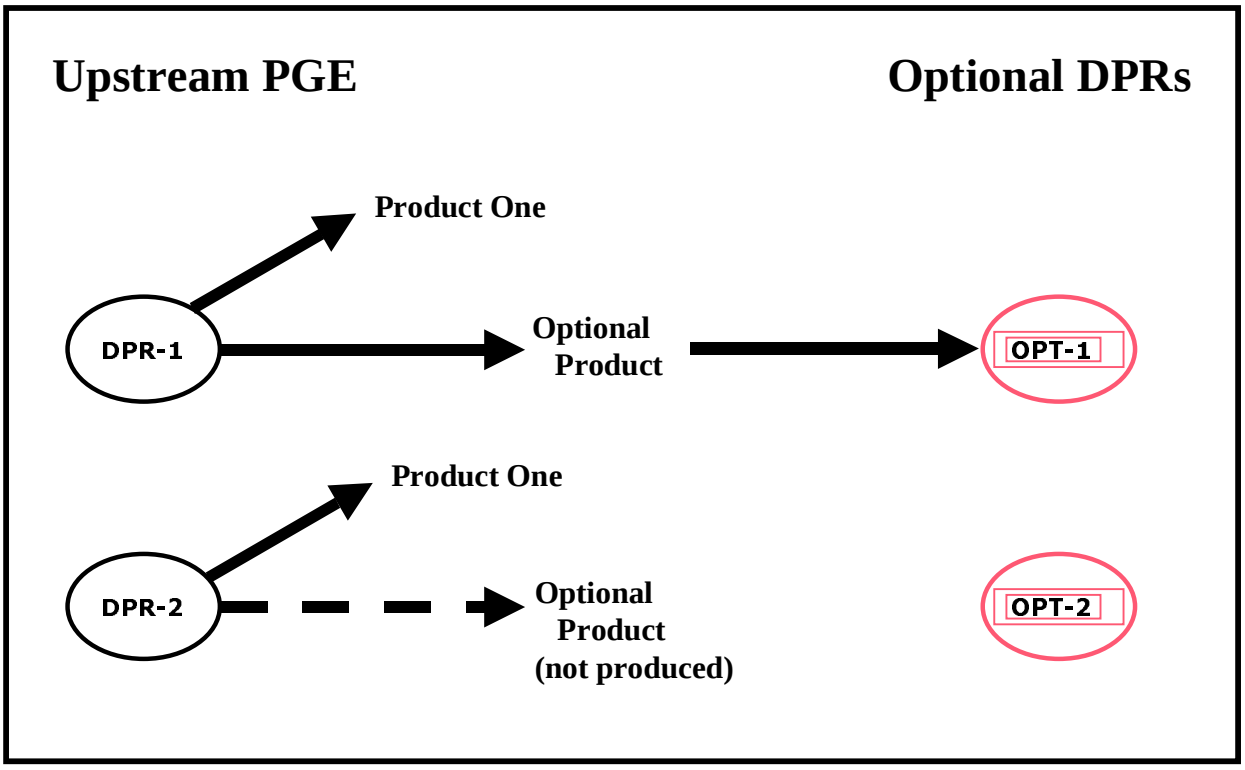


Figure 34. Example of the Optional DPRs Production Rule

The Optional DPRs Production Rule is set up during the SSI&T process. It uses many of the same parameter settings as the Basic Temporal Production Rule so the values specified in the Basic Temporal Production Rule (or other production rules) are set first, then the Optional DPRs Production Rule syntax is added.

PGE Science Metadata ODL File Parameters

The following two types of PGE science metadata ODL file entries must be made in order to set up the Optional DPRs Production Rule:

SCHEDULE_TYPE.

KEY_INPUT.

The SCHEDULE_TYPE parameter is set as follows:

SCHEDULE_TYPE = “Data”

- This demonstrates the appropriateness of the term “Data-Scheduled Production Rule.”
- Other schedule types include Time, Tile, Orbit, and Snapshot.

The key input is designated by including the following parameter in the PCF_ENTRY for whichever input is to be the key input:

KEY_INPUT = “Y”

- Assigning a value of “Y” to the KEY_INPUT parameter identifies the data as a key input and it is subsequently treated as such.
- Either assigning a value of “N” to the KEY_INPUT parameter or leaving out the parameter entirely identifies the non-key input data.
- Only one key input is allowed per PGE profile.

The Production Planner’s role in the implementation of the Optional DPRs Production Rule was described in the MODIS and ASTER examples previously described and varies with the kind of key input:

Optional output of an upstream PGE (MODIS example).

- Production Planner creates Production Requests for the PGE subject to the Optional DPRs Production Rule and specifies the same date/time range as for the upstream PGE.
- Some of the DPRs generated as a result of the Production Request will never run due to lack of input data.

Ingested on an irregular time schedule (ASTER example).

- Production Planner specifies the data **insertion time** range when creating Production Requests.
- All DPRs generated as a result of the Production Requests should be capable of running.

Intermittent Activation Production Rule

The conditions for executing most PGEs are well defined. The most common activation condition is the availability of all input data sets. Similarly, the frequency of execution is usually well defined (e.g., run once for every granule or run monthly averages once a month). However, some PGEs have additional or different constraints on when they are run.

A PGE can be set up to run on every n^{th} instance of input data. For example, a QA PGE that is run on a daily product may need to be run only every fifth day to provide a spot check. Note that this does **not** refer to the common case of running a weekly averaging PGE only once each week, which would be handled by the Basic Temporal Production Rule and the time ranges specified for the input and output ESDTs. Rather, this is a special case where a PGE **can** be run every day

(or hour, week, etc.), but for some reason (such as a QA check) it is desired to run the PGE only every n^{th} day.

To implement the Intermittent Activation Production Rule the Production Planner supplies the following information (via the Production Request Editor) when creating a production request:

Number to Skip

- Number of DPRs to be skipped (not executed).
- Entered in the **Skip** field on the Production Request Editor.

Number to Keep

- After skipping the specified number of DPRs, how many are to be kept?
- Entered in the **Keep** field on the Production Request Editor.
- The number to keep is usually one but could be any number.

Skip First

- Button on the Production Request Editor.
- Selected to skip the first DPR.
- Not selected if the first DPR is to be run.

The Planning Subsystem uses the preceding information to establish a pattern of execution. The pattern is effective for the single PR in which the "number to skip" and the "number to keep" are specified; it is not maintained between PRs.

The following example of the Intermittent Activation Production Rule is shown in Figure 35:

The Production Planner prepares a production request for a 14-day period, generating 14 DPRs.

The Production Planner made the following selections on the Production Request Editor:

- Entered "4" in the **Number to Skip** field.
- Entered "1" in the **Number to Keep** field.
- Did **not** select the **Skip First** button.

Consequently, the following results are obtained:

- First DPR runs.
- Four DPRs (second through fifth) are skipped.

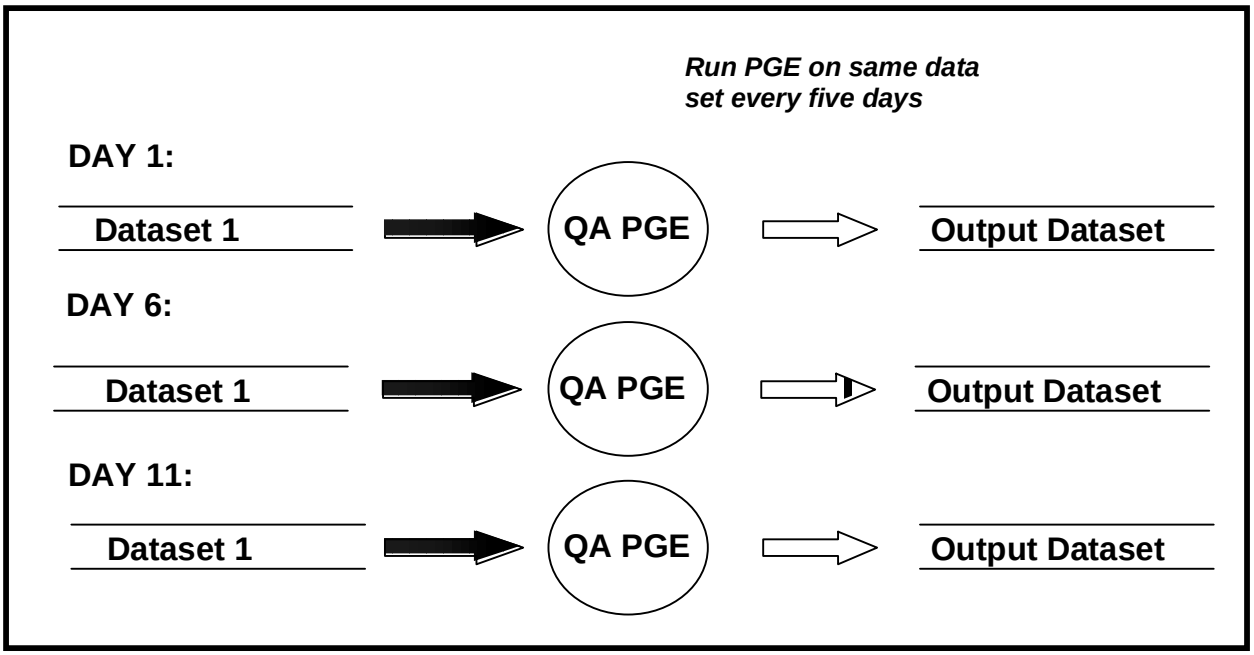


Figure 35. Example of the Intermittent Activation Production Rule

- Sixth DPR runs.
- Four DPRs (seventh through tenth) are skipped.
- Eleventh DPR runs.
- Remaining three DPRs (twelfth through fourteenth) are skipped.

Metadata Checks and Metadata Query Production Rules

The Metadata Checks and Metadata Query Production Rules are similar in definition and use. Both production rules allow the PGE to specify granule-level metadata values that define whether the PGE can accept one (or more) of its inputs. The rules differ only in the results of metadata search performed.

Metadata Checks Production Rule.

- When PLS requests the Science Data Server to search for the input(s), the Science Data Server "checks" the metadata of all granules that match the time frame with respect to the value(s) allowed by the PGE.
- If any granule fails to match the specified value(s), the PGE is not executed.

Metadata Query Production Rule.

- When PLS requests the Science Data Server to search for the input(s), the Science Data Server adds to the query the metadata value(s) desired by the PGE.

- Only the granules that match the time frame of the PGE plus the granule-level metadata value(s) specified by the PGE are staged for the PGE to use as input.
- If no granules are found matching the conditions and the input is not optional, the PGE is not executed.

Example of Metadata Checks:

- A MODIS PGE is run when the Percent Cloud Cover of its inputs is greater than 25 percent.
- The Metadata Checks Production Rule is used to specify the granule-level metadata value of greater than 25.
- When the PGE is scheduled and is ready to start, two granules match the timeframe of the Production Request for the input with the Metadata Check.
- If both granules have a Percent Cloud Cover greater than 25 percent, execution of the PGE starts and both granules are staged.
- If one of the granules has a Percent Cloud Cover of 15 percent, the PGE is not executed.

Example of Metadata Query:

- A MODIS PGE is run when as many granules as possible of one of its inputs have a QA Value = "Good".
- The Metadata Query Production Rule is used to specify the granule-level metadata value = "Good".
- When the PGE is scheduled and is ready to start, two granules match the time frame of the production request for the input with the Metadata Query.
- If both granules have a QA Value = "Good", execution of the PGE starts and both granules are staged.
- If one of the granules has a QA Value = "Bad", the PGE executes but with only one granule (the one with QA Value = "Good").

The Metadata Checks and Metadata Query Production Rules are used in conjunction with the times specified in the Basic Temporal Production Rule or other production rules. The Metadata Check or Query is added information that further refines what granules are sought by the PGE.

Multi-Granule ESDTs are a special case of the Metadata Query Production Rule. Multi-Granule ESDTs are used for PGE inputs or outputs when more than one granule of the same ESDT exists for the same temporal range (time period). The Multi-Granule ESDT mechanism employs a metadata parameter to differentiate between the "equal in time" granules. A metadata parameter is selected that is unique across granules for the same time period and that is used by PDPS to keep track of which granule is which when the granules are produced. Later, if only one of a pair of granules for a particular time period is needed as input to the PGE, the Metadata Query is used to ensure that PDPS schedules the correct granule as input.

The **Data Day Production Rule** is actually an addition to the Metadata Query Production Rule involving runtime parameter values. There is a pair of settings (Start Data Day and End Data Day) that allow a PGE to perform a Metadata Query for the start of the Data Day and the end of the Data Day. A separate section of this lesson is devoted to the Data Day Production Rule.

Using runtime parameter values is a capability of the Metadata Query and Metadata Checks Production Rules. Rather than use a hard-coded value for the check or query, a value computed from one of the other production rules can be used.

Figure 36 illustrates the Metadata Checks and Metadata Query Production Rules. If no Metadata Check or Query were applicable, the PGE shown in the figure would use three granules of input (i.e., Granules A through C). However, let us assume that the metadata value to be checked/queried is %CloudCover. Each granule has a different value for %CloudCover.

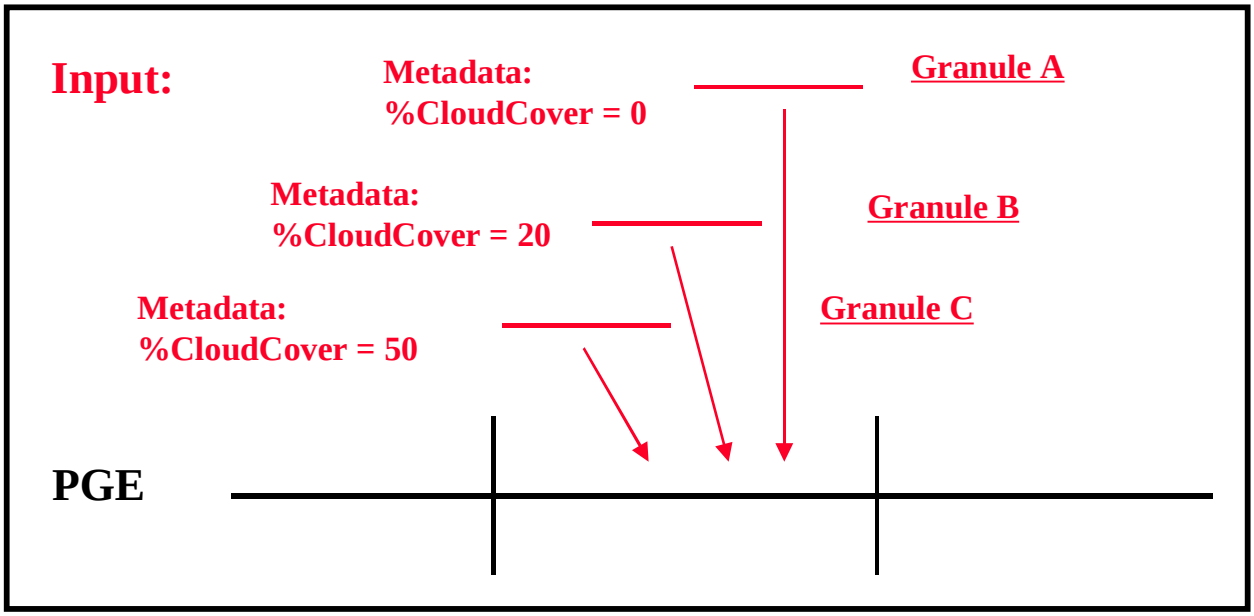


Figure 36. Example of the Metadata Checks and Query Production Rules

The following results demonstrate the differences between the Metadata Checks and Metadata Query Production Rules, especially with respect to the number of inputs that the PGE receives when different values are specified:

Metadata Check of %CloudCover < 80:

- In this case all three granules are acquired and the PGE is scheduled and executed.

Metadata Query of %CloudCover < 80:

- All three granules are acquired and the PGE is scheduled and executed.

Metadata Check of %CloudCover = 50:

- The PGE is not scheduled because only one of the three granules (Granule C) meets the criterion.

Metadata Query of %CloudCover = 50:

- Granule C is found and if the PGE's Min/Max Granules parameters are set to allow one granule, that one granule is acquired and the PGE is scheduled and executed.

Metadata Check of %CloudCover < 50:

- The PGE is not scheduled because only two of the three granules (Granule A and B) meet the criterion.

Metadata Query of %CloudCover < 50:

- Granules A and B are found and if the PGE's Min/Max Granules parameters are set to allow two granules, the granules are acquired and the PGE is scheduled and executed.

Metadata Check of %CloudCover <= 50:

- The PGE is scheduled and executed because all three granules meet the criterion.

Metadata Query of %CloudCover <= 50:

- All three granules are found and acquired and the PGE is scheduled and executed.

Metadata Check of %CloudCover = 20:

- The PGE is not scheduled because only one of the three granules (Granule B) meets the criterion.

Metadata Query of %CloudCover = 20:

- Granule B is found and if the PGE's Min/Max Granules parameters are set to allow one granule, the granule is acquired and the PGE is scheduled and executed.

Metadata Check of %CloudCover < 20:

- The PGE is not scheduled because only one of the three granules (Granule A) meets the criterion.

Metadata Query of %CloudCover < 20:

- Granule C is found and if the PGE's Min/Max Granules parameters are set to allow one granule, the granule is acquired and the PGE is scheduled and executed.

Metadata Check of %CloudCover = 10:

– The PGE is not scheduled because none of the three granules meets the criterion.

Metadata Query of %CloudCover = 10:

– The PGE is not scheduled because no granules are returned from the query (unless Minimum Granules is set to 0).

Note that there can be more than one Metadata Check or Metadata Query on a given input. In the preceding example, a Metadata Check on %CloudCover can be combined with a Metadata Query on another parameter to further limit the input.

The Metadata Checks and Metadata Query Production Rules are additions to settings/syntax put into the ODL files for other production rules. The addition of a Metadata Check or a Metadata Query to an input means that other production rules used to evaluate that input will be applied in combination with the Metadata Check or Metadata Query.

PGE Science Metadata ODL File Parameters

Although the Metadata Checks and Metadata Query Production Rules are similar, there are two different ODL objects used to define them within a PCF_ENTRY in the PGE science metadata ODL file; i.e., the METADATA_CHECKS object and the METADATA_QUERY object.

The METADATA_CHECKS object has the following syntax:

```
OBJECT = PCF_ENTRY
    .
    .
    .
    .
    .
    OBJECT = METADATA_CHECKS
        .
        .
        .
    END_OBJECT = METADATA_CHECKS
END_OBJECT = PCF_ENTRY
```

The METADATA_QUERY object has the same syntax except “METADATA_QUERY” replaces “METADATA_CHECKS” in every instance.

Most of the following parameters must be set in the PGE science metadata ODL file within the METADATA_CHECKS or METADATA_QUERY ODL object (as applicable) in order to implement either the Metadata Checks or Metadata Query Production Rule:

CLASS.

PARM_NAME.

OPERATOR.

VALUE.

DATABASE_QUERY.

KEY_PARAMETER_NAME (optional).

KEY_PARAMETER_VALUE (optional).

CLASS is a simple counter used to differentiate the different Metadata Checks or Metadata Query objects within the file. Since each Metadata Checks or Metadata Query object resides within a different PCF_ENTRY object, the CLASS for an METADATA_CHECKS or METADATA_QUERY object can always be 1 (e.g., CLASS = 1).

PARM_NAME is the name of the metadata parameter on which the check or query is to be performed. The value specified for PARM_NAME (e.g., PARM_NAME = “%CloudCover”) must be part of the granule-level metadata of the ESDT. In addition, it must match the parameter name specified in the ESDT science metadata ODL file.

OPERATOR is the operator (e.g., OPERATOR = “==”) on which the check/query is to be performed. The following values are valid for OPERATOR:

">"

- Value in metadata must be greater than.

"<"

- Value in metadata must be less than.

">="

- Value in metadata must be greater than or equal to.

"<="

- Value in metadata must be less than or equal to.

"=="

- Value in metadata must be equal to.

"!="

- Value in metadata must be **not** equal to.

VALUE is the value (e.g., VALUE = 50) against which the metadata parameter (defined by PARM_NAME) is compared (using the operator specified by the OPERATOR parameter). The value for the VALUE parameter should be the type of data (e.g., integer, string) as defined in the ESDT ODL metadata for the parameter.

DATABASE_QUERY indicates whether the value for the Metadata Check or Query should be retrieved from the PDPS database rather than through the use of the VALUE parameter. Specifying DATABASE_QUERY permits **runtime parameter values** to be used for Metadata Query or Metadata Checks. The following values are valid for the DATABASE_QUERY parameter:

"NONE"

- Use the value in the VALUE parameter; no value from the PDPS database is used.

"PATH NUMBER"

- Use the Path Number (0-233) of the orbit for which the PGE is scheduled.

"ORBIT NUMBER"

- Use the Orbit Number of the orbit for which the PGE is scheduled.

"TILE ID"

- Use the Tile ID of the current Data Processing Request.

"START DATA DAY"

- Use the Start Data Day for the current Data Processing Request.

"END DATA DAY"

- Use the End Data Day for the current Data Processing Request.

KEY_PARAMETER_NAME is an optional parameter that is used to specify the container within a multi-container metadata group (i.e., the MeasuredParameters metadata group in most ESDTs). The KEY_PARAMETER_NAME (e.g., KEY_PARAMETER_NAME = "ParameterName" for metadata checks or queries within the MeasuredParameters group) in conjunction with the KEY_PARAMETER_VALUE allows PDPS to determine which container within the multi-container group is to be the object of the check or query. KEY_PARAMETER_NAME is **not** used for product-specific attributes.

KEY_PARAMETER_VALUE is an optional parameter that is used to specify the **value** (e.g., KEY_PARAMETER_VALUE = "LandCoverage") for the container within a multi-container metadata group (i.e. the MeasuredParameters metadata group in most ESDTs). The KEY_PARAMETER_VALUE in both the PGE science metadata ODL file and ESDT science metadata ODL file must match.

Multi-Granule ESDTs are created by adding the following parameter to the PCF_ENTRY in the PGE science metadata ODL file:

DISTINCT_VALUE.

The DISTINCT_VALUE must be set to the value of the metadata parameter that is used to differentiate granules within the Multi-Granule ESDT. In addition, the input or output defined by the PCF entry must have a corresponding DISTINCT_PARAMETER entry in the ESDT science metadata ODL file.

ESDT Science Metadata ODL File Parameters

The METADATA_DEFINITION ODL object surrounds the definition for Metadata Checks or Metadata Query information within the ESDT science metadata ODL file. An OBJECT/END_OBJECT pair is needed to separate the parameters defining the Metadata Definition from the rest of the parameters defining the ESDT with the following syntax:

```
OBJECT = METADATA_DEFINITION
    .
    .
    .
END_OBJECT = METADATA_DEFINITION
```

A METADATA_DEFINITION object can match multiple Metadata Checks or Metadata Query objects in various PGE science metadata ODL files. There is no difference between the two production rules with respect to the parameters that need to be set in the ESDT science metadata ODL file. Most of the following parameters must be set:

CLASS.

PARM_NAME.

CONTAINER_NAME.

TYPE.

KEY_PARAMETER_NAME (optional).

KEY_PARAMETER_VALUE (optional).

CLASS is a simple counter used to differentiate the different Metadata Definition objects within the file. Each Metadata Definition object within the file must have a **different** CLASS value.

PARM_NAME is the name of the Metadata parameter on which the check or query will be performed. The value specified for PARM_NAME must be part of the granule-level metadata of the ESDT. It must also match the parameter name specified in the PGE science metadata ODL file(s).

CONTAINER_NAME is the name of the Metadata Group within which the metadata parameter defined by PARM_NAME is contained. For product-specific attributes CONTAINER_NAME is set to the string "AdditionalAttibutes" (i.e., CONTAINER_NAME = "AdditionalAttibutes").

TYPE indicates the type of data within the metadata parameter. The following values are valid for TYPE:

"INT"

- Integer data.

"FLOAT"

- Floating point data.

"STR"

- String or character data.
- Note that dates and times are considered string data.

KEY_PARAMETER_NAME is an optional parameter that is used to specify the container within a multi-container metadata group (i.e., the MeasuredParameters metadata group in most ESDTs). The KEY_PARAMETER_NAME allows PDPS to determine which container within the multi-container group is to be the object of the check or query.

KEY_PARAMETER_VALUE is an optional parameter that is used to specify the value for the container within a multi-container metadata group (i.e., the MeasuredParameters metadata group in most ESDTs). The KEY_PARAMETER_VALUE in both the ESDT science metadata ODL file and PGE science metadata ODL file must match.

The ESDT science metadata ODL file for an input specifying Multi-Granule ESDTs needs to have the following parameter added:

DISTINCT_PARAMETER.

The DISTINCT_PARAMETER must be set to the name of the metadata parameter that is used to differentiate granules within the Multi-Granule ESDT. A corresponding METADATA_DEFINITION must be created to help PDPS find the specified metadata parameter when querying the Science Data Server.

Data Day Production Rule

The Data Day Production Rule is an addition to the Metadata Query Production Rule involving runtime parameter values. The Data Day Production Rule uses a query to the PDPS database for the time period for the DPR and a Metadata Query for data matching the Data Day. The Data Day is defined as a day within twelve hours of the current day. There is a pair of settings (Start Data Day and End Data Day) that provide parameters for the Metadata Query.

The Start Data Day and End Data Day values are calculated by subtracting twelve hours from the starting day for which the PGE is executing and adding twelve hours onto the ending day for which the PGE is running.

Data Day for a PGE is running on data from 07/04 00:00:00 to 07/05 00:00:00 is defined as follows:

- START_DATA_DAY = 07/03 12:00:00
- END_DATA_DAY = 07/06 12:00:00.

Spatial Query Production Rule

The Spatial Query Production Rule allows a PGE to select input(s) based on the spatial coverage of another input (called the **key input**). The PDPS queries the Science Data Server for the spatial coverage of the key input, then uses it in acquiring any subsequent inputs that the PGE has requested that have the same spatial coverage.

Example:

- Level 0 input data for an ASTER DPR covers a small section of the Earth.
- The PGE requires ancillary data that covers the same area to complete its processing.
- The PGE uses the Spatial Query Production Rule to mark the geographic input as its key input.
- The PGE specifies that the ancillary input is to be retrieved for the same spatial coverage as that of the key input.
- When PDPS finds an input granule for the PGE, it performs a Spatial Query to acquire the ancillary input with the same spatial coverage as that of the key input.

Without specifying coordinates, PDPS can match inputs against the spatial constraint of the key input, and give to a PGE only those granules which overlap in area.

For Release 6A Spatial Pad will be added to the Spatial Query Production Rule. Spatial Pad is a means of padding the spatial constraints of the key input. The specified pad is added to all sides of the key input's spatial shape. All granules that intersect the expanded area are retrieved.

Figure 37 is an illustration of the Spatial Query Production Rule. The figure shows a PGE that has two input types, one of which is the key input. The other type of input has granules labeled with the names of various colors. One granule (i.e., “green”) of the key input is found. The spatial coordinates of the granule are retrieved and all inputs of the second ESDT are checked for overlap with the key input’s coordinates.

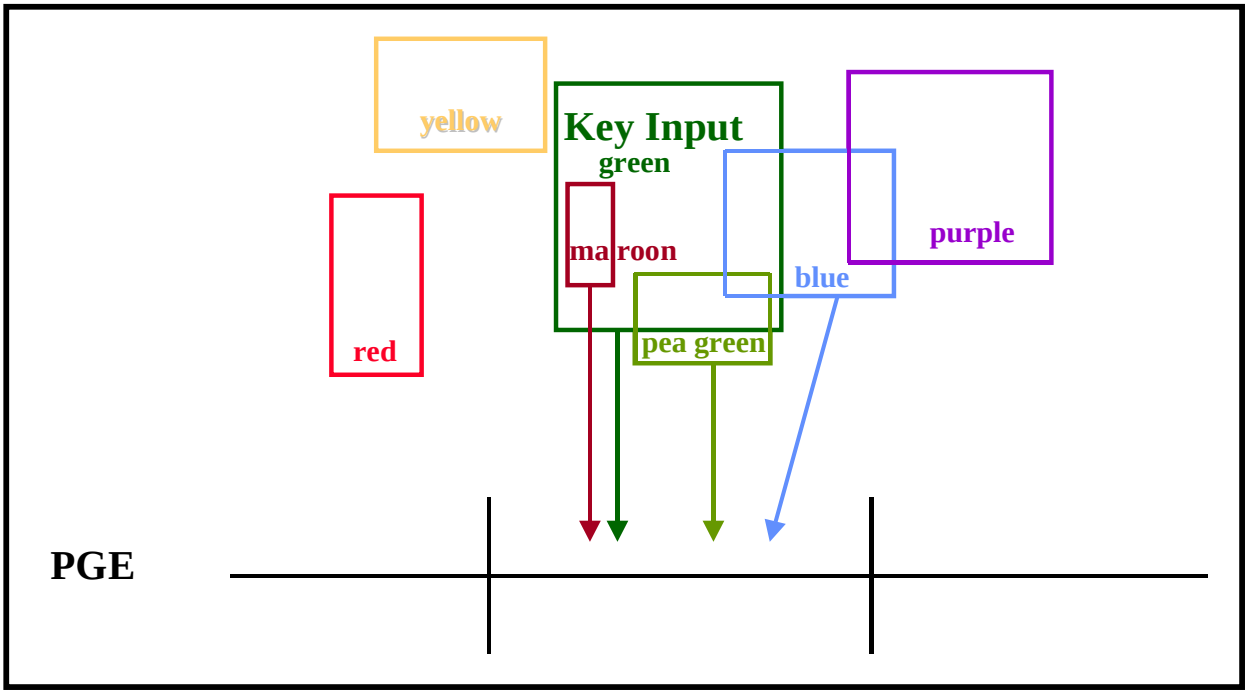


Figure 37. Example of the Spatial Query Production Rule

Assuming that all granules relate to the same time period, the granules are evaluated as follows:

The “yellow” granule is not retrieved as an input because its spatial coordinates do not overlap with those of the key input.

The “red” granule is not retrieved as an input because its spatial coordinates do not overlap with those of the key input.

The “blue” granule is retrieved as an input because its spatial coordinates overlap with those of the key input. Part of its spatial constraint is within the constraint of the key input.

The “maroon” granule is retrieved as an input because its spatial coordinates overlap with those of the key input. The spatial constraint of this granule is completely within the constraint of the key input.

The “pea green” granule is retrieved as an input because its spatial coordinates overlap with those of the key input. Part of its spatial constraint overlaps with that of the key input.

The “purple” granule is not retrieved as an input because its spatial coordinates do not overlap with those of the key input. It does not matter that it overlaps with another input that is accepted (i.e., the “blue” granule).

The Spatial Query Production rule is somewhat of an addition to other production rules. As such, it needs the same parameter settings as the Basic Temporal Production Rule. The values specified in the Basic Temporal Production Rule (or other production rules) are set first, then the Spatial Query Production Rule syntax is added.

PGE Science Metadata ODL File Parameters

In order to implement the Spatial Query Production Rule the following two parameters must be defined in the applicable PCF_ENTRY (each input is defined by a separate PCF_ENTRY in the PGE science metadata ODL file):

KEY _INPUT.

QUERY_TYPE.

The entries are made in the following format:

```
OBJECT = PCF_ENTRY
    .
    .
    .
    QUERY_TYPE = “Spatial”
    KEY_INPUT = “Y”
    .
    .
    .
END_OBJECT = PCF_ENTRY
```

QUERY_TYPE indicates what type of query is to be done to acquire the input defined by the PCF_ENTRY object. Valid values are as follows:

"Temporal" - Input is acquired based on time.

- The Basic Temporal and/or the Advanced Temporal Production Rules is/are used to get the input.
- “Temporal” is the value that is assumed if the parameter is left out of the PCF_ENTRY object.

"Spatial" - Input is acquired based on spatial coordinates (as well as time).

- An input must be designated the key input to be used in determining the spatial constraints of the search.
- “Spatial” is the value specified for each input that uses the Spatial Query Production Rule.

"Tile" - Input is acquired by the spatial definition of a tile.

- Refer to the Tiling Production Rule for additional information.

"Already Created Tile" - Input is acquired based on the tile ID of an already created tile.

- Refer to the Tiling Production Rule for additional information.

The KEY_INPUT is the input on which the spatial queries for other inputs will be based. When a KEY_INPUT parameter is assigned a value of “Y” the corresponding input is designated a key input and is treated as such. A value of “N” or leaving out the parameter entirely specifies a non-key input. Only one (1) key input is allowed per PGE Profile.

Tiling Production Rule

The Tiling Production Rule allows a PGE to run over a series of specific geographic locations called "tiles". The tiles are defined before the PGE is scheduled, specifying the longitude and latitude of four points that outline each tile. When the PGE is scheduled, it is scheduled for an entire day, and data is queried based on both a timeframe and the geographic location specified. Each run of the PGE for that day is for a specific tile, and only data that overlap or fit within the geographical coordinates of the tile are staged for the PGE.

Example:

- A MODIS PGE is designed to run on data for a specific geographic location every day.
- The location is expressed as a polygon defined by latitude and longitude coordinates.
- The MODIS PGE is scheduled every day, and data are retrieved that match the time period (the day for which the PGE is being executed) and some part of it falls within the geographic constraints of the tile.
- The PGE runs and produces data that define information about the particular tile.

Period and **boundary** are used to specify the timing of input data and provide indications of how often the PGE should be executed. But at least some of the input data are retrieved on the basis of the coordinates defined for the tile on which the PGE is executing. In fact there are really two kinds of tiling:

The PGE takes in data based on geographic shapes (tiles) and produces an output or outputs for the specified geographical coverage.

The PGE takes in an already tiled product as input.

- This form of tiling is more like a Metadata Query using a runtime parameter value to acquire the correct tiled data.

There are some possible future enhancements to the Tiling Production Rule but they have not been scheduled yet:

Zonal Tiling supports tiles that cover a band around the Earth between two given latitudes.

Tile Clustering involves grouping tiles that cover nearby geographic locations together so that data that span the tiles may be staged only once.

- Intended to improve the performance of Tiling.
- Also provides for the ability to prioritize one group of tiles over others (so specific geographic outputs are produced before other geographic outputs).

Runtime parameters can be set to the ID of the tile being processed. Since PDPS schedules a Tiling PGE to run once per tile, it can pass the identifier of the tile to the PGE. The identifier can be placed under a specified runtime parameter in the PCF, or it can be used in a Metadata Query for a PGE that would use already tiled data as input.

Figure 38 provides an example of the Tiling Production Rule. The PGE runs once per defined tile. So for every tile in the Tile Scheme a Data Processing Request is created to run using data that match the geographic extent of the tile. The PDPS sends the coordinates of the tiles (e.g., Tiles 1 through 3 in Figure 38) to the Science Data Server when requesting data and acquires only the granules that fall fully or partially within the defined tile.

The PGE itself must be set up to handle the fact that the entire area of the tile may not be covered by available data. In addition, because PDPS does not keep track of tiles once they have been produced, the PGE must set the metadata of the output products so a downstream Tiling PGE can acquire the correct granules for a given tile. The PDPS matches up the granules needed for a downstream PGE via a query to the Data Server Subsystem.

Tiling Based on Already Tiled Data

As previously stated, the second form of Tiling concerns PGEs based on tiles that have already been created by other PGEs. Tiling based on already tiled data is really a combination of the Metadata Query Production Rule and the Tiling Production Rule. The latter is used in running the PGE(s) once per tile, just like any other Tiling PGE. The Metadata Query Production Rule is used in acquiring the previously tiled data by querying the Science Data Server for metadata that match the tile ID that is currently being executed. The query depends on the “runtime parameters” function of Tiling to provide the tile ID relevant to the PGE that is currently being executed.

The Tiling Production Rule is based (at least for the PGE science metadata ODL file) on the same fields used for the Basic Temporal Production Rule. A PGE that performs Tiling still needs a **boundary** and **period** and other such parameters. The difference is that values specified for some of the fields provide Tiling information. Furthermore, Tiling requires that a tile scheme be identified in the PGE science metadata ODL file. The tile scheme is defined in a tile science metadata ODL file.

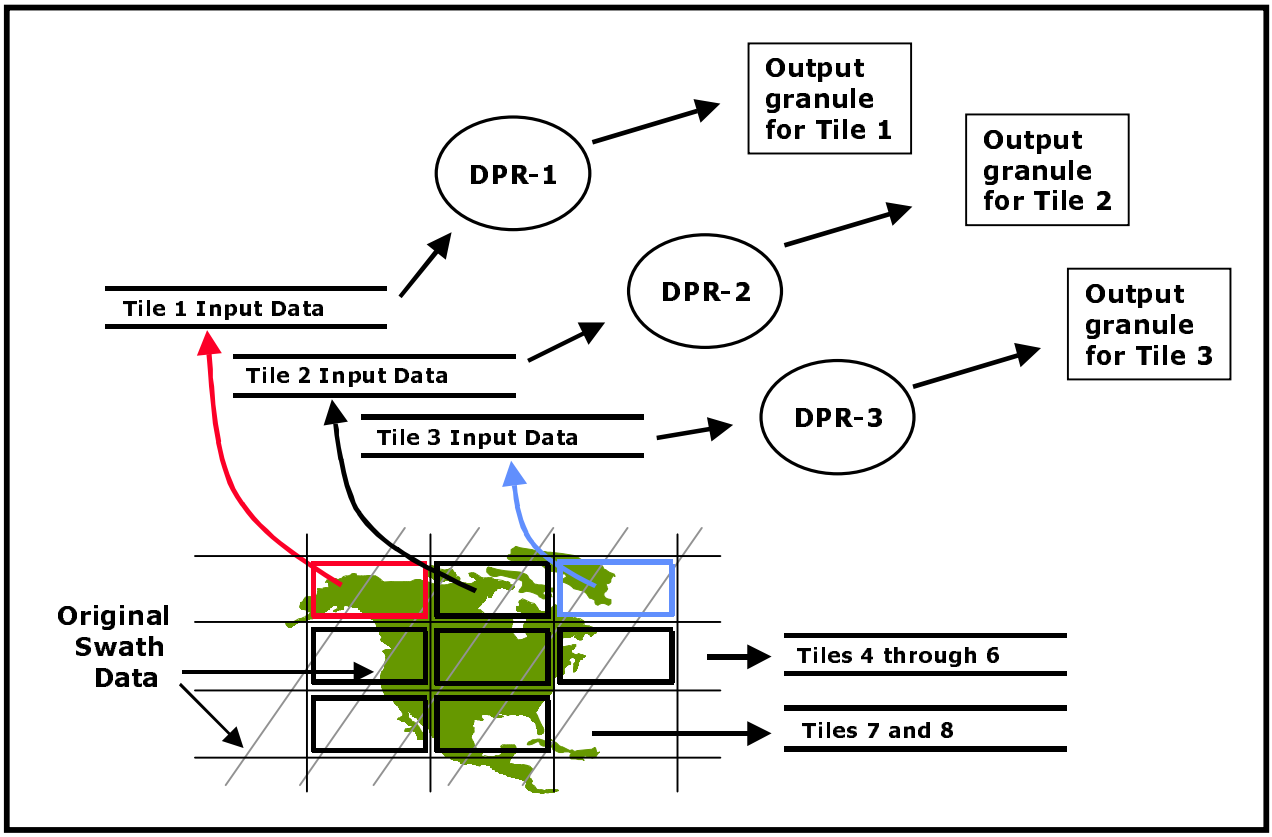


Figure 38. Example of the Tiling Production Rule

PGE Science Metadata ODL File Parameters

The following parameters must be set in the PGE science metadata ODL file in order to implement the Tiling Production Rule:

SCHEDULE_TYPE.

TILE_SCHEME_NAME.

In addition, the following parameter is used within a PCF_ENTRY when defining the Tiling Production Rule:

QUERY_TYPE.

The SCHEDULE_TYPE parameter defines the type of scheduling that will be done for the PGE. Values for the Tiling Production Rule are:

"Tiling"

- Tile-Scheduled.
- The PGE is scheduled based on the specified PROCESSING_PERIOD and PROCESSING_BOUNDARY, but a DPR is created for each defined tile.

The TILE_SCHEME_NAME parameter is the name of the Tile Scheme to be used by PDPS when scheduling and executing PGEs for each defined tile. There must be a tile ODL file that matches the specified scheme name.
The QUERY_TYPE parameter specifies the type of query to be performed on the input defined by the PCF_ENTRY Object. It uses the following syntax:

```
OBJECT = PCF_ENTRY
    .
    .
    .
    QUERY_TYPE =
    .
END_OBJECT = PCF_ENTRY
```

For Tiling PGEs there are two possible values for QUERY_TYPE:

"Tile"

- The data for the input are acquired on the basis of the spatial constraints of the current tile.
- Used for a PGE that takes in raw data and produces one or more tiles of data.

"Already Created Tile"

- The input is a tiled output of another Tiling PGE.
- Used for a PGE that takes input from one or more other Tiling PGEs.
- A Metadata Query must be added to this PCF_ENTRY in order for the correct tiled input to be acquired.

Tile Science Metadata ODL File Parameters

The following parameter must be set in the Tile science metadata ODL file in order to implement the Tiling Production Rule:

TILE_SCHEME_NAME.

In addition, the following ODL objects are used within a PCF_ENTRY to define the Tiling Production Rule:

TILE object.

TILE_COORDINATE object.

The TILE_SCHEME_NAME parameter identifies the tile scheme for which the tile information is being specified. Values are limited by the following constraints:

The string specified can be no more than 20 characters.

The string specified should match the string specified for TILE_SCHEME in the PGE science metadata ODL file.

The TILE object is an ODL object that surrounds each tile definition. An OBJECT/END_OBJECT pair (as shown in the example that follows) is needed for each tile that is going to be expressly defined:

```
OBJECT = TILE
    .
    .
    .
END_OBJECT = TILE
```

The following parameters are set in the TILE object in order to implement the Tiling Production Rule:

CLASS.

TILE_ID.

TILE_DESCRIPTION.

CLASS is a simple counter used to differentiate the different TILE objects within the file. Each TILE object needs to have a different CLASS value.

TILE_ID is the tile identifier for the tile being defined. The TILE_ID must be an integer (e.g., TILE_ID = 12) and must be greater than zero but less than the maximum integer. If a Tile ID is defined in other tile schemes, it must have the same coordinates and description.

TILE_DESCRIPTION is a string of characters (255 characters maximum) that describes what the tile is for, such as its geographic location or area that it covers (e.g., TILE_DESCRIPTION = "Upper North America").

The TILE_COORDINATE object is an ODL object that defines a coordinate (latitude and longitude) for a tile. An OBJECT/END_OBJECT pair is needed for each coordinate that is defined. Each tile must have four TILE_COORDINATE objects defined. (Currently only four-sized polygons are allowed; however, a possible future enhancement would provide for polygons with more than four points.) Coordinate objects must follow a clockwise sequence so that if lines were drawn between the points in the order they are given the desired shape would be drawn.

Coordinate objects conform to the following format:

```
OBJECT = TILE
    .
    .
    .
    OBJECT = TILE_COORDINATE
       CLASS =
       LATITUDE =
       LONGITUDE =
    END_OBJECT = TILE_COORDINATE
    .
    .
    .
```

```
END_OBJECT = TILE
```

The following parameters are set in the TILE_COORDINATE object in order to implement the Tiling Production Rule:

CLASS.

LATITUDE.

LONGITUDE.

The CLASS parameter (e.g., CLASS = 1) is an object counter that is used only to distinguish objects. The value assigned to CLASS must be an integer greater than zero and must be unique in the file for the particular type of object.
The LATITUDE parameter (e.g., LATITUDE = 12.15) describes the latitude component of the tile coordinate. There is one LATITUDE entry per TILE_COORDINATE object.
The LONGITUDE parameter (e.g., LONGITUDE = -43.22) describes the longitude component of the tile coordinate. There is one LONGITUDE entry per TILE_COORDINATE object.

Closest Granule Production Rule

The Closest Granule Production Rule allows a PGE to request the nearest input granule from the Data Processing Request time. The PDPS requests a search forward or backward for a specified period of time until it finds a granule that matches the request. However, there is a limit to the number of queries that are performed. The number of queries and the period length of the query are specified during SSI&T.

Example:

- A PGE processes data at daily intervals and could use a particular type of calibration granule that would allow it to determine the nearest parameters of the instrument.
- Although most calibration coefficients are defined as static granules, in this case there is a dynamic granule that is received about once a month.
- The closest such granule would be optimum, so the PGE uses the Closest Granule Production Rule to search forward or backward from the time of the DPR to find the nearest calibration granule.

The Closest Granule Production Rule supersedes the Most Recent Granule Production Rule. The latter allowed the search for inputs to go backward in time from the start of the DPR. The Closest Granule Production Rule allows the search for input granules to go either backward or forward in time, increasing the flexibility of the rule. The Closest Granule Production Rule has all of the ability of Most Recent Granule, plus the ability to search forward in time for input data. The Closest Granule Production Rule uses two values to determine the period of the query. The two values are concerned with the direction of the query and the number of queries allowed.

Offset.

- Tells the PDPS software the query duration.
- The sign (+ or -) indicates whether the query goes forward (positive) or backward (negative) in time.

Retries.

- Tells the PDPS software how many time periods (as defined by the offset) to search (either forward or backward) in time for matching granule.

The PDPS does a Basic Temporal query before using Closest Granule to find the input. If the desired input is not found within the time period of the DPR, PDPS performs a query against the Science Data Server for the period defined by the offset. Again, if no matching granule is found, PDPS repeats the query, going backward or forward in time by the value specified in the offset. If no acceptable granule has been found before the maximum number of queries is reached, PDPS fails to generate the DPR due to insufficient input data.

Figure 39 illustrates the Closest Granule Production Rule. In the example, the PGE has a boundary of "start of day" and a period of one hour, so it is scheduled to run for one hour's worth of input data. The input has a period of one hour, and can come in at any hour of the day. Consequently, the PGE requests one granule of input.

The PGE has defined the Closest Granule Production rule with a –6-hour period of the query, meaning that it queries back in time in six-hour intervals. The number of retries is two. The PDPS performs a query for the input based on the time period of the DPR. Not finding any matching data, it uses the Closest Granule information to query for a six-hour period beginning six hours before the start time of the DPR. Again nothing is found, so a second Closest Granule query is performed, this one six hours before the last Closest Granule query. The second query results in the discovery of two matching granules. The PDPS selects the granule that is later in time and schedules the PGE to use it as input.

If the Closest Granule Production Rule were used in conjunction with the Minimum/Maximum Number of Granules Production Rule, it might be possible for both granules to be selected in the previously described Closest Granule query. If the example included setting the Maximum Number of Granules to two, both granules would be selected as input to the PGE.

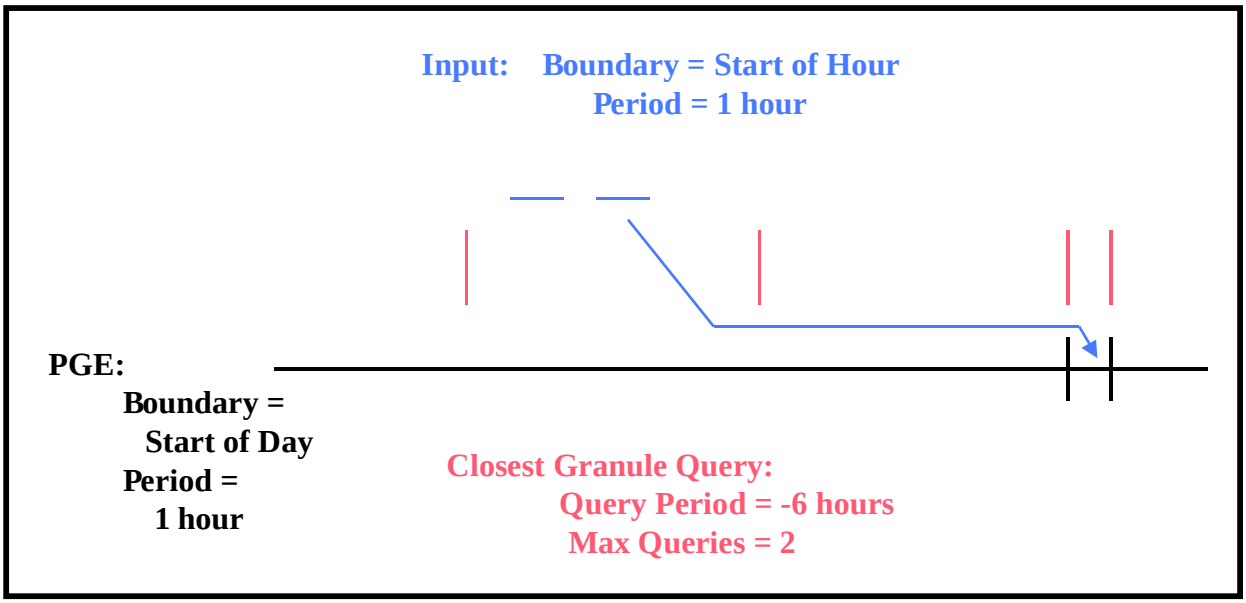


Figure 39. Example of Closest Granule Production Rule

The Closest Granule Production Rule needs the same parameter settings as the Basic Temporal Production Rule. The values needed for the Basic Temporal Production Rule must be set before the Closest Granule Production Rule syntax is added.

PGE Science Metadata ODL File Parameters

In addition to the parameter settings for the Basic Temporal Production Rule, the following parameters must be set within the appropriate PCF_ENTRY in the PGE science metadata ODL file in order to implement the Closest Granule Production Rule:

CLOSEST_QUERY_OFFSET.

CLOSEST_QUERY_RETRIES.

CLOSEST_QUERY_OFFSET is the offset added to or subtracted from the Data Start Time of the DPR and uses as the query for the requested input data type. The specified value has the format "<Period Type>)=<Length of Period>" (e.g., CLOSEST_QUERY_OFFSET = “HOURS=6”). The following “Period Type” values are used in implementing the Closest Granule rule:

"WEEKS"

- Offset is some number of weeks.
- For example, "WEEKS=2" would be a 14-day offset.

"DAYS"

- Offset is some number of days.
- For example, "DAYS=5" would be a 120-hour offset.

"HOURS"

- Offset is some number of hours.
- For example, "HOURS=4" would be a 240-minute offset.

"MINS"

- Offset is some number of minutes.
- For example, "MINS=5" would be a 300-second offset.

"SECS"

- Offset is some number of seconds.

CLOSEST_QUERY_RETRIES is the maximum number of Closest Granule queries before the DPR fails due to insufficient input data. The specified value is an integer value that is limited only by the maximum size of an integer on the executing hardware.
Note that the longer the offset value or the greater the number of retries, the more time that each query requires due to search time at the Science Data Server and processing time of any granules returned. The combination of a large offset with a large number of retries, can (if no data granules are found) consume a lot of time while failing to generate a DPR.

Orbital Processing Production Rule

The Orbital Processing Production Rule is similar to the Basic Temporal Production Rule in that both define the time period for the inputs and outputs of the PGE. The difference is that the Orbital Processing Production Rule uses the orbit of the spacecraft to determine that time period. A PGE that processes data related to every orbit of a satellite uses data related to a time period that is computed from the orbit of that satellite.

Example:

- A PGE processes Level 0 data related to each orbit of the AM-1 satellite.
- The AM-1 satellite has an orbital period of 98 minutes so the PGE is scheduled to process data for each 98-minute interval.
- Since Level 0 data are received every two hours, the data staged for the PGE include every Level 0 granule that falls within the 98 minute PGE interval.
- Only one granule of Level 0 data is relevant to some 98-minute orbits.
- Two granules of Level 0 data are relevant to other 98-minute orbits.

The Orbital Processing Production Rule uses the “period” and “boundary” concept just like the Basic Temporal Production Rule. The difference is that for Orbital Processing, the orbit of the spacecraft is taken into account when a PGE or its data are marked as **orbit scheduled**.
When responding to a Production Request for orbit-scheduled processing, PDPS determines the orbit of the satellite via information provided during the SSI&T process. The information (stored in the PDPS database) gives the start time and length of a particular orbit or set of orbits. PDPS extrapolates (or interpolates in the case of an orbit between two orbital periods stored in the database) the start and end times of the PGE that is specified in the Production Request. Data

are sought on the basis of the derived start and stop times and the appropriate data granule(s) is/are staged before the PGE is executed.

Orbital path is the path of the satellite over the Earth. It is a number from 0-233 that indicates the region of the Earth covered by a particular orbit. Note that because of the implementation of Orbital Path, there needs to be a mapping between the orbital path calculated by PDPS and the orbital path number expected by the PGEs.

Runtime parameters can be set to values associated with Orbital Processing. The following list of orbital parameters can be placed under runtime parameters:

Orbit Number.

- The number of the orbit (starting from zero) and continually increasing.

Orbital Path Number.

- The number of the path that maps to the orbit number.
- The orbital path number is the 0-233 orbital path traversed by the satellite.

Orbit Number within the Day.

- The number of the orbit within the given day.
- It includes any orbit that starts within the given day.
- This is a **Release 6A** capability.

Granule Number within the Orbit.

- The number of the granule within a given orbit.
- It includes any granule that starts within the given orbit.
- This is a **Release 6A** capability.

Figure 40 provides an illustration of the Orbital Processing Production Rule. The PGE in the diagram takes a two-hour input, but is scheduled based on the orbit time and period of the satellite. PDPS uses the data collected at SSI&T to predict the time of the orbit and performs the query to the Science Data Server for the input based on that extrapolated or interpolated orbital time. Granules of input data are allocated to DPRs based on their ability to cover the time period relevant to the DPR.

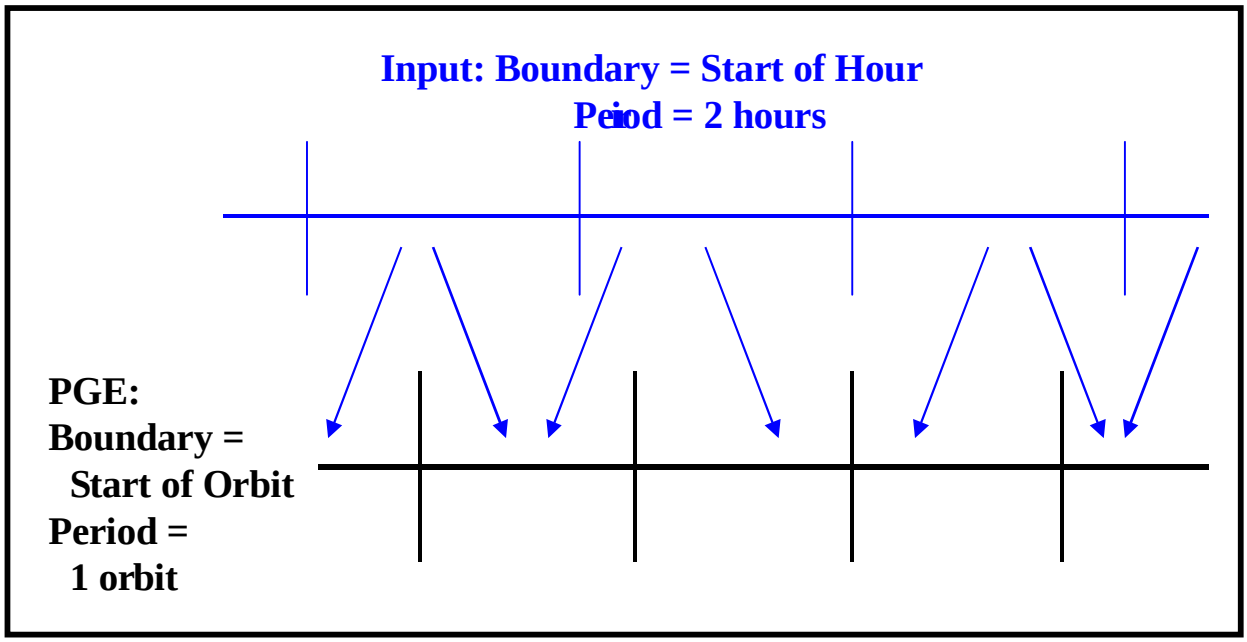


Figure 40. Example of the Orbital Processing Production Rule

In the example shown in Figure 40 the length of an orbit is less than the period of the two-hour input, so sometimes a single granule may cover the input time range of a PGE execution and at other times two granules are required. The production rule would work equally well if the data were of a shorter period (e.g., 1/2 hour) than the orbit of a satellite (e.g., 90 minutes). In such a case three granules would be staged for every execution of the PGE.

The Orbital Processing Production Rule is based (at least for the PGE science metadata ODL file) on the same fields used for the Basic Temporal Production Rule. However, the values specified for the parameters provide orbit information rather than time-period information.

PGE Science Metadata ODL File Parameters

The following parameters must be set in the PGE science metadata ODL file in order to implement the Orbital Processing Production Rule:

PLATFORM.

SCHEDULE_TYPE.

PROCESSING_PERIOD.

PROCESSING_BOUNDARY.

The PLATFORM parameter is the name of the platform (satellite) for which the PGE is processing data. Information concerning the orbits of a satellite are stored in the PDPS database. Values that can be assigned to the parameter are subject to the following constraints:

The string specified can have no more than 25 characters.

The string specified should match the string specified in the orbit science metadata ODL file. If no matching file is found, an error is reported during SSI&T.

The SCHEDULE_TYPE parameter describes the type of scheduling that is required for the PGE. "Orbit" is the value used for Orbital Processing. As a result, the PGE is scheduled based on the start time and period of the satellite's orbit. Note that PROCESSING_PERIOD and PROCESSING_BOUNDARY must be set correspondingly.

The PROCESSING_PERIOD is the time interval for the data that the PGE processes. Assuming no combination of production rules that would affect the period, data are acquired for the specified PROCESSING_PERIOD and output data are planned for the given period. The value assigned to PROCESSING_PERIOD is of the format "<Period Type>=<Length of Period>". The “Period Type” applicable to the Orbital Processing Production Rule is "ORBITS". For example, "ORBITS=1" would be applied to a PGE that processes data related to one orbit’s worth of data.

The PROCESSING_BOUNDARY is the boundary (starting point in time) of the PGE. It specifies when each instance of the PGE should start. Note that the PROCESSING_BOUNDARY and PROCESSING_PERIOD are used in conjunction when scheduling the PGE. Consequently, "START_OF_ORBIT" is the acceptable PROCESSING_BOUNDARY for the Orbital Processing Production Rule. It indicates that the PGE processes data related to each satellite orbit. It must be used in conjunction with a PROCESSING_PERIOD that specifies a “Period Type” of “ORBITS”.

Orbit Science Metadata ODL File Parameters

The following parameter must be set in the orbit science metadata ODL file in order to implement the Orbital Processing Production Rule:

PLATFORM.

In addition, the following ODL object is used in defining orbits for the Orbital Processing Production Rule:

ORBIT_MODEL object.

The value assigned to the PLATFORM parameter in the orbit science metadata ODL file must be exactly the same as that specified for the same parameter in the PGE science metadata ODL file.

The ORBIT_MODEL object is an ODL object that surrounds each orbit definition. An OBJECT/END_OBJECT pair (as shown in the example that follows) is needed for each orbit that is to be expressly defined: PDPS extrapolates or interpolates orbits that are not specifically defined within the file.

```
 OBJECT = ORBIT_MODEL
CLASS = 1
ORBIT_NUMBER = 1000
ORBIT_PATH_NUMBER = 68
ORBIT_PERIOD = "MINS=98"
ORBIT_ START = “09/21/1999 14:50:00”
END_OBJECT = ORBIT_MODEL
```

The following parameters are set in the ORBIT_MODEL object in order to implement the Orbital Processing Production Rule:

CLASS.

ORBIT_NUMBER.

ORBIT_PATH_NUMBER.

ORBIT_PERIOD.

ORBIT_START.

CLASS is a simple counter used to differentiate the different ORBIT_MODEL objects within the file. Each ORBIT_MODEL object needs to have a different CLASS value.

ORBIT_NUMBER is simply the number of the orbit being specified. Each orbit of the satellite has a sequential number associated with it. This is the integer value of the orbit number for the orbit being defined in the ORBIT_MODEL object.

ORBIT_PATH_NUMBER is value of the path for the specified orbit. The orbital path is a number from 0-233 that repeats every 16 days. This is the integer value of the orbital path number for the orbit being defined in the ORBIT_MODEL object.

ORBIT_PERIOD is the length of time it takes for the satellite to complete one orbit. The value assigned to ORBIT_PERIOD has the format "<Period Type>=<Length of Period>" (e.g., “MINS=98”). Note that the “Length of Period” is specified as a positive integer only.

Period Type values for the orbit model science metadata OFL file are:

"WEEKS"

- Orbit spans some number of weeks.
- For example, "WEEKS=2" would be an orbit that takes two weeks to complete.

"DAYS"

- Orbit spans some number of days.
- For example, "DAYS=5" would be an orbit that takes five days to complete.

"HOURS"

- Orbit spans some number of hours.
- For example, "HOURS=4" would be an orbit that takes four hours to complete.

"MINS"

- Orbit spans some number of minutes.
- For example, "MINS=85" would be an orbit that takes eighty-five minutes to complete.

"SECS"

- Orbit spans some number of seconds.
- For example, "SECS=7200" would be an orbit that takes 7200 seconds (two hours) to complete.

ORBIT_START is the start date and time for the orbit defined by the particular ORBIT_MODEL object. Its format is either "MMM DD YYYY HH:MM:SS" or “MM/DD/YYYY HH:MM:SS”.

Production Planning Considerations

During normal operations it is expected that the Production Planner will not have to add PRs to the PDPS database very frequently. The frequency of this activity is, to some extent, determined by the SCF responsible for the science software.

The PR is a template request to generate a particular data product and results in a production run of the associated SCF-provided PGE.

- PR specifies a range (temporal, orbit, or tile) over which the data products are to be produced or the PGEs are to be scheduled.
  - PR might request that the data product be produced for only a single day’s data.
  - PR might request that data products be produced for every opportunity of input data for several months, resulting in several hundred jobs being planned and run as the input data become available.
- Early in a mission the SCF may prefer to request processing for a short time period only (e.g., a week or less).
  - At that time the SCF is gaining an understanding of the on-orbit behavior of the instrument, the resulting data, and the interaction of the science processing software with real data.
  - SCF reviews the quality of the products and notifies the Production Planner of the need for any changes to the PR (e.g., discontinue the PR, change time ranges, or modify input parameters).
- When the SCF has developed a good understanding of the instrument’s behavior, the team may be comfortable requesting processing for months at a time.
- DAAC operations may have operational reasons for wanting to issue processing requests for a more limited time period.

The Production Planner has to balance the various considerations when determining whether or not to create or update a PR.

Planning decisions are made on the basis of locally defined planning strategies for supporting the SCFs’ data processing needs. The production planning tools are intended to be flexible enough in their design to support the particular planning and scheduling cycles of the operations organization at each DAAC.

Before planning production the Production Planner must coordinate with the Resource Planner to resolve all resource allocation issues. The Resource Planner notifies the Production Planner of

the resources available for use in processing. Furthermore, the Production Planner may well have direct access to the Resource Plan.
The Production Planner prepares monthly and weekly production plans. In addition, the Production Planner develops a daily production schedule from the most current weekly plan. However, the first step in the planning process is creating production requests using the Production Request Editor.

DPREP Considerations

DPREP (data preprocessing) is a set of three PGEs that are supplied by ECS, unlike most PGEs, which are provided by the Science Computing Facilities that ECS supports. DPREP consists of the following three PGEs:

EcDpPrAm1EdosEphAttDPREP_PGE (Step 1).

EcDpPrAm1FddAttitudeDPREP_PGE (Step 2).

EcDpPrAm1FddEphemerisDPREP_PGE (Step 3).

The PGEs run separately and in a particular sequence.
Two files describe the PGEs and how to run them:

"DPREP_README"

"HowtoRunDPREP"

The files are installed on the science processor hosts (e.g., e0spg01, g0spg01, l0spg01, n0spg03) in the **/usr/ecs/*MODE*/CUSTOM/data/DPS directory**.

The DPREP PGEs process Level Zero (L0) Terra (AM-1) spacecraft data (e.g., ESDT AM1ANC) provided by EDOS. The output files/granules of the DPREP PGEs are subsequently used in the processing of data from various instruments on the satellite. They provide the following types of ancillary (non-science) data:

Ephemeris

- Spacecraft location: ephemeris (or orbit) data include: latitude, longitude, and height.

Attitude

- Orientation of the satellite, including yaw, pitch, and roll angles; and angular rates about three axes.

There are two profiles for DPREP PGEs:

Profile 1 runs routinely at the DAACs using previous DPREP output in addition to new Terra ancillary (e.g., AM1ANC) data.

Profile 2 (the boot-up procedure) takes in the Terra ancillary data only and is run under two sets of conditions:

- First run of DPREP (because there is no previous output) to initialize DPREP processing.

- Following any long period of time during which EDOS L0 ancillary data are unavailable. (Short gaps in the ephemeris data are filled by EcDpPrAm1EdosEphemerisRepair, one of the executables in the EcDpPrAm1EdosEphAttDPREP_PGE.)

In order to run Profile 2 successfully following a long period of data unavailability, DPREP must be told where to resume orbit counting. The initial orbit number in the Step 1 process control file (PCF), must be set to the orbit number corresponding to the timestamp at which data availability resumes.

Until an automated process can be implemented, whenever there is a telemetry drop-out, a member of the DAAC science support team takes the following actions:

Calls the Flight Operations Team (FOT).

Asks for the on-line engineer.

Requests the orbit number that coincides with the start time of the first L0 ancillary data set that follows the data drop-out.

Sets the orbit number in the Step 1 PCF.

Then Profile 2 can be run successfully. Afterward, routine operations can be resumed using Profile 1 PGEs.

Production Rules Technical Information Sources

1 ECS Baseline Information System :

2 **PDPS Home Page: http://dmserver.gsfc.nasa.gov/ecsdev/relb/pdps/index.html**

http://pete.hitc.com/baseline/ -choose drop4, you will find latest patch documentation and much more.

3 EOS Instrument Team Science Software (PGE's):

http://ecsinfo.hitc.com/iteams/Science/science.html. Production Rules White Paper and much more.

4 MODIS - Science Data Processing Software Version 2.0 System Description, SDST-104 dated August 25, 1998. **http://ltpwww.gsfc.nasa.gov/MODIS/SDST/docs.html**

5 Test Scenarios for selected Production Rules can be viewed by accessing the SCF at: **</home/dheroux/DPS/TESTBED/MISR/SSIT/V2/ODL/Scenarios**>

DPREP

Introduction

This section contains information to run DPREP.

DPREP is made up of three PGE's each run separately.

The PGE's are titled Step1 DPREP, Step 2 DPREP and Step 3 DPREP.

The input files normally come from INGEST.

These files are depicted in two of the three step DPREP process in Table 16. DPREP Step 3 will be available with Drop 5A.

The output files generated from each of the DPREP PGE's contains Ancillary Attitude, and Ephemeris data that becomes new inputs to Instrument PGE's.

These Instrument PGE's will then process its satellite data with similar time span files created by DPREP. The DPREP registration process for each of the three PGE's creates in the Science Data Server Archive a subscription for each of the DPREP PGE's. PGE execution then takes place in the PDPS.

The SSI&T effort for DPREP PGE's is similar in effort to what would be required to register any other PGE.

SSI&T Activity for DPREP

The Level Zero datasets are received in 2 hour chunks. The file processes in Table 16, depicts the minimal time span allowable for a DPREP run. In a normal operation of DPREP, a twenty four hour time span would be prepared for. This would require additional 2 hour chunks and thus additional files of data would need to be registered. Before the registration process can take place a number of files will have to be updated to process a block of data for a particular time period. Therefore, DPREP input files will have to be identified and various templates for the SSI&T process will require annotation.

The sections that follow in the have been highlighted with notations as to what SSI&T process applies in the preparation of each template and the function required to register each section. With a particular function identified, other portions of this manual can be referred to for more detailed procedures to be used to carry out the full SSI&T process.

Whenever new input files are introduced or updated executables are re-introduced it is wise to first set up the PGE to run from the Command Line. This will determine if what has been introduced will run error free. After a successful Command Line run it is advisable then to complete the SSI&T effort to run from the PDPS. Command Line Runs include the use the PCF to run from in the Science Data Server. PDPS runs include ESDT's and ODL files to generate internal PCF's.

DPREP File Processes

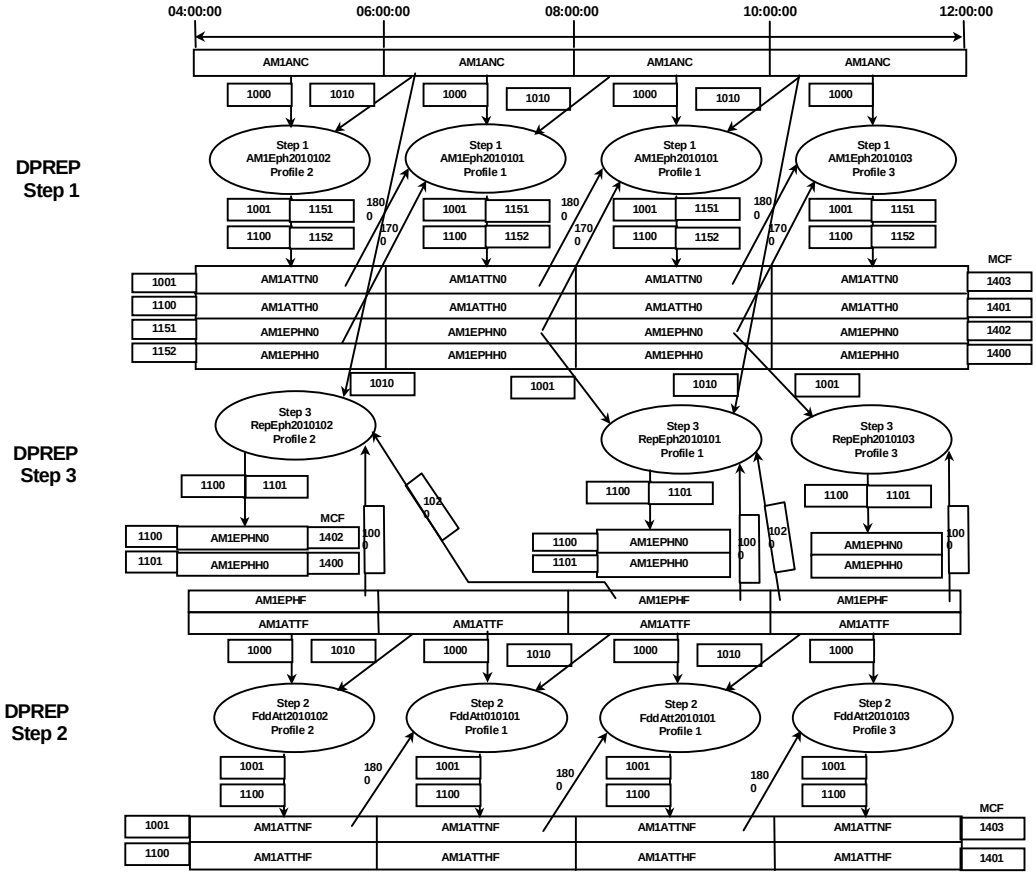


Table 16. DPREP File Processes

DPREP Processes and Procedures Used in Testing

Processes and procedures are provided in the following files on an SGI machine used to support SSI&T:

DPREP README and HowToRunAM1DPREP files located at :/usr/ecs/OPS/CUSTOM/data/DPS/

DPREP binary located: :/usr/ecs/OPS/CUSTOM/bin/DPS/

The following files are taken from an SSI&T SGI machine using the /data/DPS/ thread for referencing DPREP files in this section.

/data3/ecs/OPS/CUSTOM/data/DPS

p0spg01{emcleod}21: ls -lrt

total 13568

```
-rwxr-xr-x    1 cmops    cmops     449920 Jul 16  1998
P0420004AAAAAAAAAAAAAAA96182015958001
-rwxr-xr-x    1 cmops    cmops        376 Jul 16  1998
P0420004AAAAAAAAAAAAAAA96182015958000
-rwxr-xr-x    1 cmops    cmops      93408 Jul 16  1998
EOS_AM1_Ephemeris_FDF.eph
-rwxr-xr-x    1 cmops    cmops     451104 Jul 16  1998
EOS_AM1_Ephemeris_EDOS.eph
-rwxr-xr-x    1 cmops    cmops     449920 Jul 16  1998
P0420004AAAAAAAAAAAAAAA96182035958001
-rwxr-xr-x    1 cmops    cmops        376 Jul 16  1998
P0420004AAAAAAAAAAAAAAA96182035958000
-rwxr-xr-x    1 cmops    cmops      93152 Jul 16  1998 EOSAM1_1998-01-13.eph
-rwxr-xr-x    1 cmops    cmops      93184 Jul 16  1998 EOSAM1_1998-01-14.eph
-rwxr-xr-x    1 cmops    cmops      92672 Jul 16  1998 EOSAM1_1998-01-13.att
-rwxr-xr-x    1 cmops    cmops      29430 Jul 16  1998 DpPrPreProcess.pcf
-rwxr-xr-x    1 cmops    cmops      15058 Jul 16  1998 DpPrEphReplacer.pcf
-rwxr-xr-x    1 cmops    cmops      13840 Feb  8 15:21 PGS_101
-rwxr-xr-x    1 cmops    cmops       6207 Feb  8 15:21 HowToRunDPREP
-rwxr-xr-x    1 cmops    cmops       2834 Feb  8 15:21 HowToCreateDprepTarFile
-rwxr-xr-x    1 cmops    cmops        384 Feb  8 15:21 EDOS_LEVEL_ZERO_00.curr
-rwxr-xr-x    1 cmops    cmops     449344 Feb  8 15:21 EDOS_LEVEL_ZERO_01.curr
-rwxr-xr-x    1 cmops    cmops        384 Feb  8 15:21 EDOS_LEVEL_ZERO_00.next
-rwxr-xr-x    1 cmops    cmops     449984 Feb  8 15:21
EDOS_LEVEL_ZERO_01.next_2
-rwxr-xr-x    1 cmops    cmops     449984 Feb  8 15:21
EDOS_LEVEL_ZERO_01.next_1
-rwxr-xr-x    1 cmops    cmops     449280 Feb  8 15:21 EDOS_LEVEL_ZERO_01.next
-rwxr-xr-x    1 cmops    cmops        384 Feb  8 15:21
EDOS_LEVEL_ZERO_00.next_2
-rwxr-xr-x    1 cmops    cmops        384 Feb  8 15:21
EDOS_LEVEL_ZERO_00.next_1
-rwxr-xr-x    1 cmops    cmops        384 Feb  8 15:21
EDOS_LEVEL_ZERO_00.next.step3
```

```
-rwxr-xr-x   1 cmops    cmops     449984 Feb  8 15:21
EDOS_LEVEL_ZERO_01.next.step3
-rwxr-xr-x   1 cmops    cmops     225280 Feb  8 15:21
AM1_DEFINITIVE_ATT.fdd.next_1
-rwxr-xr-x   1 cmops    cmops     225280 Feb  8 15:21
AM1_DEFINITIVE_ATT.fdd.next
-rwxr-xr-x   1 cmops    cmops     225280 Feb  8 15:21
AM1_DEFINITIVE_ATT.fdd.curr
-rwxr-xr-x   1 cmops    cmops       5909 Feb  8 15:21 FDDAttitude.tar.met
-rwxr-xr-x   1 cmops    cmops      14000 Feb  8 15:21 AM1_REPAIR_EPH.fdd
-rwxr-xr-x   1 cmops    cmops       1681 Feb  8 15:21
AM1_DEFINITIVE_ATT.fdd.next_1.met
-rwxr-xr-x   1 cmops    cmops       1681 Feb  8 15:21
AM1_DEFINITIVE_ATT.fdd.next.met
-rwxr-xr-x   1 cmops    cmops       1681 Feb  8 15:21
AM1_DEFINITIVE_ATT.fdd.curr.met
-rwxr-xr-x   1 cmops    cmops       1654 Feb  8 15:21 AM1ANC.next_2.met
-rwxr-xr-x   1 cmops    cmops       1654 Feb  8 15:21 AM1ANC.next_1.met
-rwxr-xr-x   1 cmops    cmops       1654 Feb  8 15:21 AM1ANC.next.met
-rwxr-xr-x   1 cmops    cmops       1654 Feb  8 15:21 AM1ANC.curr.met
-rwxr-xr-x   1 cmops    cmops      26819 Feb  8 15:21
EcDpPrAm1FddEphemerisDPREP_PGEDPREP.Step3.PCF.curr
-rwxr-xr-x   1 cmops    cmops      28627 Feb  8 15:21
EcDpPrAm1FddAttitudeDPREP_PGE.Step2.PCF.next
-rwxr-xr-x   1 cmops    cmops      29026 Feb  8 15:21
EcDpPrAm1FddAttitudeDPREP_PGE.Step2.PCF.curr
-rwxr-xr-x   1 cmops    cmops      30520 Feb  8 15:21
EcDpPrAm1EdosAncillary.Steps1ab.PCF.next_1
-rwxr-xr-x   1 cmops    cmops      30553 Feb  8 15:21
EcDpPrAm1EdosAncillary.Steps1ab.PCF.next
-rwxr-xr-x   1 cmops    cmops      30485 Feb  8 15:21
EcDpPrAm1EdosAncillary.Steps1ab.PCF.curr
-rwxr-xr-x   1 cmops    cmops       5909 Feb  8 15:21
AM1_Ancillary_DPREP.tar.met
-rwxr-xr-x   1 cmops    cmops      26819 Feb  8 15:21
EcDpPrAm1FddEphemerisDPREP_PGE.Step3.PCF.next
-rwxr-xr-x   1 cmops    cmops       5816 Feb  8 15:21 DPREP_README
```

DPREP consists of three PGE's each run separately.

1 The first step is a **ksh** script called **EcDpPrAm1EdosEphAttDPREP_PGE**, which serves as a driver for three executables:

EcDpPrAm1EdosAncillary

EcDpPrAm1EdosEphemerisRepair

EcDpPrAm1ToolkitHdf

2 The second step is **EcDpPrAm1FddAttitudeDPREP_PGE.**

3 The third step is **EcDpPrAm1FddEphemerisDPREP_PGE**.

PGE Name-STEP ONE

EcDpPrAm1EdosAncillary reads in AM-1 L0 (EDOS) Ancillary Dataset (logical id 1000, ESDT AM1ANC). It also reads another set of AM-1 L0 (EDOS) Ancillary Dataset (logical id 1010, ESDT AM1ANC). The second set of L0 data is required to insure that incomplete orbits in the first data set get complete orbit metadata records. The only data that will be extracted from the second data set is the descending node time and longitude.

EcDpPrAm1EdosAncillary also reads in ephemeris and attitude data (toolkit native format) under logical ids 1500 (ESDT AM1EPHN0) and 1502 (ESDT AM1ATTN0). These would be the last ephemeris/attitude data sets generated from a previous run of this PGE.

EcDpPrAm1EdosEphemerisRepair

If EcDpPrAm1EdosAncillary signals a short gap was detected then EcDpPrAm1EdosEphemerisRepair reads the scratch file created by EcDpPrAm1EdosAncillary and performs the gap fill and writes a gap filled native format ephemeris file. If no short gap is signaled then the scratch file is simply renamed to the native format ephemeris file.

EcDpPrAm1ToolkitHdf

This process takes the native format ephemeris file and produces a corresponding HDF file and a metadata file.

This PGE produces reformatted attitude and ephemeris data sets. The attitude data sets are produced using logical ids 1001 (ESDT AM1ATTN0) and 1100 (ESDT AM1ATTH0). The ephemeris data sets are produced using logical ids 1151 (ESDT AM1EPHN0) and 1152 (ESDT AM1EPHH0). The corresponding MCFs are accessed using logical ids 1400 (ESDT AM1EPHH0), 1401 (ESDT AM1ATTH0), 1402 (ESDT AM1EPHN0) and 1403 (ESDT AM1ATTN0).

The PGE produces an ASCII report file under logical id 2000

EcDpPrAm1EdosAncillary.Steps1ab.PCF.curr is a sample pcf used by this PGE. This pcf has directory locations and file names which may not be valid in all environments. Be careful before directly using this pcf. A sample set of input files for this PGE are

1000 EDOS_LEVEL_ZERO_00.curr
1000 EDOS_LEVEL_ZERO_01.curr
1010 EDOS_LEVEL_ZERO_00.next
1010 EDOS_LEVEL_ZERO_01.next

and a copy of each of these files is provided. The files in this group do contain small data gaps which will cause EcDpPrAm1EdosEphemerisRepair to be run.

The remaining pcfs for Step 1 DPREP:

EcDpPrAm1EdosAncillary.Steps1ab.PCF.next
EcDpPrAm1EdosAncillary.Steps1ab.PCF.next_1

will move through the sequence of the Level Zero files provided.

example; EDOS_LEVEL_ZERO_00.next is treated as current and EDOS_LEVEL_ZERO_00.next_1 is treated as next etc...These pcfs can be utilized at the testers discretion and are not necessary for the running of steps 2 and 3. The input files indicated in EcDpPrAm1EdosAncillary.Steps1ab.PCF.next_1 contain no gaps so testers shouldn't be concerned that EcDpPrAm1EdosEphemerisRepair is not invoked when using this pcf.

Note:

The Level Zero datasets are received in 2 hour chunks but processing can't be performed on the most recently available 2 hour chunk. Step 1 processing needs to look forward in the time stream in order to complete orbit metadata processing.

PGE Name-STEP TWO

EcDpPrAm1FddAttitudeDPREP_PGE reads in the current FDD Attitude Dataset under logical ID 1000 and the next FDD Attitude Dataset under logical ID 1010. It also reads in the attitude data set it produced with it's last run under logical ID 1502. The output of this process is a native format attitude file (logical ID 1001) and an HDF format attitude file (logical ID 1100). A .met file is also produced for each.
EcDpPrAm1FddAttitudeDPREP_PGE.Step2.PCF.curr is a sample pcf used by this PGE. This pcf has directory locations and file names which may not be valid in all environments. Be careful before directly using this pcf. A sample set of input files for this PGE are
1000 AM1_DEFINITIVE_ATT.fdd.curr
1010 AM1_DEFINITIVE_ATT.fdd.next
(IMPORTANT: These files contain incorrect data and were delivered only as a placeholder. EcDpPrAm1FddAttitudeDPREP_PGE will not run with these files as input. When valid FDD Definitive Attitude files become available they should be moved to these filenames and EcDpPrAm1FddAttitudeDPREP_PGE should run successfully.)
and a copy of each of these files is provided.

Note: Step 2 processing must follow 2 hours behind step 1 processing.

PGE Name-STEP THREE

If step1 finds too many missing data points in the ephemeris data it signals that a definitive ephemeris file is needed from FDD which EcDpPrReplaceEphemeris will use to replace the ephemeris dataset that was generated from the Level Zero data.
EcDpPrReplaceEphemeris reads in the definitive Ephemeris dataset received from FDD (logical ID 1000) and the EOS_AM1 Ephemeris data (logical id 1001) in toolkit native format.
It produces Replacement ephemeris datasets (logical id 1101, ESDT AM1EPHH0) and (logical id 1100, ESDT AM1EPHN0). The corresponding MCFs are accessed using logical ids 1400 (ESDT AM1EPHH0) & 1402 (ESDT AM1EPHN0)
EcDpPrAm1FddEphemerisDPREP_PGE.Step3.PCF.curr is a sample pcf used by this PGE. This pcf has directory locations and file names which may not be valid in all environments. Be careful before directly using this pcf. A sample FDD input file for this PGE is
1000 AM1_REPAIR_EPH.fdd
and a copy of this file is provided.
Toolkit Initialization Settings
SGI Application Binary Interface New 32
Disk Space Used for PGE Run 50.000 MB
Shared Memory ON
Use Test Files if SM Fails ON
Use Log Files ON
Continue if Logging Fails OFF

HowToRunDPREP

DPREP is made up of 3 PGEs that are named:
Step1 DPREP, Step2 DPREP, Step3 DPREP.
Step1 DPREP is what used to be called **Am1Ancillary DPREP**.
Step2 DPREP is the **FDD Attitude DPREP**.
Step3 DPREP is **a combination of FDD Ephemeris and the replace DPREPs**.
Note: Each PGE will use both the **Science Metadata update tool** and the **Operational Metadata Update tool** to incorporate the specific parameters listed in each of the direction sections below.
If Tar files are not available for registering the three Terra/AM-1 DPREP PGEs, follow the instructions found in the **HowToCreateDprepTarFiles** file to generate the necessary tar files.

HowToCreateDprepTarFiles

p0spg01:/usr/ecs/OPS/CUSTOM/data/DPS[46] > more HowToCreateDprepTarFile

This file provides information to create the DPREP pge tar files

Objective : To create AM1_Ancillary_DPREP.tar & FddAttitude.tar

and using SSIT Manager archive the PGE tar files

Files needed :

PGS_101

EcDpPrAm1EdosEphAttDPREP_PGE

EcDpPrAm1EdosAncillary

EcDpPrAm1EdosEphemerisRepair

EcDpPrAm1ToolkitToHdf

EcDpPrAm1FddAttitudeDPREP_PGE

EcDpPrAm1FddEphemerisDPREP_PGE

Steps to create AM1_Ancillary_DPREP.tar

---

1 Login to the science processing machine **(SGI)**

2 Check to make sure that the executables

EcDpPrAm1EdosEphAttDPREP_PGE

EcDpPrAm1EdosAncillary

EcDpPrAm1EdosEphemerisRepair

EcDpPrAm1ToolkitToHdf

are delivered and reside in **$ECS_HOME/<MODE>/CUSTOM/bin/DPS**

3 Check to make sure that the data file **PGS_101** is delivered and resides in **$ECS_HOME/<MODE>/CUSTOM/data/DPS**

4 cd to $ECS_HOME/<MODE>/CUSTOM/data/DPS

5 /bin/tar cvf AM1_Ancillary_DPREP.tar PGS_101 -C $ECS_HOME/<MODE>/CUSTOM/bin/DPS EcDpPrAm1EdosAncillary EcDpPrAm1EdosEphAttDPREP_PGE EcDpPrAm1EdosEphemerisRepair EcDpPrAm1ToolkitToHdf

Step 5 will create the **AM1_Ancillary_DPREP.tar** file in **$ECS_HOME/<MODE>/CUSTOM/data/DPS**

This file then needs to be ftped to the SSIT machine from where it will be archived

The corresponding met file for this tar file is **AM1_Ancillary_DPREP.tar.met** and is delivered to **$ECS_HOME/<MODE>/CUSTOM/data/DPS**

This file then needs to be ftped to the SSIT machine from where it (and the tar file itself) will be archived

Steps to create FddAttitude.tar

---

1 Login to the science processing machine **(SGI)**

2 Check to make sure that the executable **EcDpPrAm1FddAttitudeDPREP_PGE** is delivered and resides in **$ECS_HOME/<MODE>/CUSTOM/bin/DPS**

3 Check to make sure that the data file **PGS_101** is delivered and resides in **$ECS_HOME/<MODE>/CUSTOM/data/DPS**

4 cd to $ECS_HOME/<MODE>/CUSTOM/data/DPS

5 **/bin/tar cvf FDDAttitude.tar PGS_101 -C $ECS_HOME/<MODE>/CUSTOM/bin/DPS EcDpPrAm1FddAttitudeDPREP_PGE**

Step 5 will create the **FDDAttitude.tar file** in **$ECS_HOME/<MODE>/CUSTOM/data/DPS**

This file then needs to be ftped to the SSIT machine from where it will be archived

The corresponding met file for this tar file is **FDDAttitude.tar.met** and is delivered to **$ECS_HOME/<MODE>/CUSTOM/data/DPS**

This file then needs to be ftped to the SSIT machine from where it (and the corresponding tar file) will be archived**.**

Steps to create ReplaceEphemeris.tar

---

1 **Login to the science processing machine (SGI)**

2 Check to make sure that the executable **EcDpPrAm1FddEphemerisDPREP_PGE** is delivered and resides in **$ECS_HOME/<MODE>/CUSTOM/bin/DPS**

3 Check to make sure that the data file **PGS_101** is delivered and resides in **$ECS_HOME/<MODE>/CUSTOM/data/DPS**

4 **cd to $ECS_HOME/<MODE>/CUSTOM/data/DPS**

5 **/bin/tar cvf ReplaceEphemeris.tar PGS_101 -C $ECS_HOME/<MODE>/CUSTOM/bin/DPS EcDpPrAm1FddEphemerisDPREP_PGE**

Step 5 will create the ReplaceEphemeris.tar file in **$ECS_HOME/<MODE>/CUSTOM/data/DPS**

This file then needs to be ftped to the SSIT machine from where it will be archived

The corresponding met file for this tar file is ReplaceEphemeris.tar.met and is delivered to **$ECS_HOME/<MODE>/CUSTOM/data/DPS**

This file then needs to be ftped to the SSIT machine from where it

(and the corresponding tar file) will be archived.

HowToRunDPREP

p0spg01:/usr/ecs/OPS/CUSTOM/data/DPS[48] > more HowToRunDPREP

This document contains information on how to schedule and run **AM-1 DPREP**

AM-1 DPREP is made up of **3 PGEs** that this document refers to as **Step 1 DPREP, Step 2 DPREP**, and **Step 3 DPREP:**

Step 1 DPREP is the **EDOS-**supplied ephemeris and attitude preprocessor**.**

Step 2 DPREP is the **FDD-**supplied attitude preprocessor**.**

Step 3 DPREP is the **FDD-**supplied ephemeris preprocessor**.**

Operationally, Steps 1 and 2 are scheduled daily and run independently of one another. Step 3 is scheduled and run on an as-needed basis.

In order to run, **DPREP** processing expects data to be available not only from the current segment, but also from the preceding and following segments as well. Data from the preceding and following segments are used to consistency check ephemeris and attitude data streams when the data streams bridge segment boundaries. The availability of data from the preceding and following segments is not guaranteed, however. Four data processing profiles have been developed to handle the various permutations of preceding and following data segments that could be available to **DPREP:**

Profile 1 expects data from the preceding, current, and following segments.

Profile 2 expects data from the current and following segments only.

Profile 3 expects data from the preceding and current segments only.

Profile 4 expects data from the current segment only.

These profiles are recognized by all **DPREP Steps.** The **ESDT** that is provided to **DPREP** from the preceding, current, and following data segments depends on the Step being run, however.

Profile 2 (no preceding data, but following data is available) initializes **DPREP's** processing of a given Step's ephemeris and/or attitude data stream. **Once Profile 2** has been run on a data segment, **Profile 1** (preceding and following data available) assumes processing responsibility on all data segments thereafter until data dropout or mission end is encountered. **Profile 3** (preceding data available, but no following data) then processes data segment that immediately precedes data dropout and, therefore, terminates processing on a given Step's ephemeris and/or attitude data stream**.**

In the big picture then, DPREP processing requires a single **Profile 2** to run on the first data segment, is followed by an indeterminate number of **Profile 1** processes, and is terminated by a single **Profile 3** process**.**

Profile 4 processes isolated data segments and is not likely to be scheduled operationally. More information on the **AM-1 DPREP** can be found in document **"TERRA Spacecraft Ephemeris** and **Attitude Data Preprocessing"** (document number **184-TP-001-001).**

Note: Each PGE will use both the **Science Metadata update tool** and the **Operational Metadata Update tool,** which are located in the **SSIT Manager,** to incorporate the specific parameters listed in each of the direction sections below.

Registering DPREP PGEs

So that the following tests can be conducted, all four Profiles must be registered for all three DPREP Steps. Each DPREP PGE is registered once.

Step 1 DPREP Test Instructions

For Step 1, **current** and **following** granules are EDOS-supplied L0 Ancillary (APID 4, ShortName AM1ANC), the preceding granule is the output Toolkit format ephemeris and attitude products produced by the preceding run of Step 1 (ShortNames AM1EPHN0, AM1ATTN0). DPREP expects both the L0 Ancillary header granule and the data granule (hence, 2 granules) for the current and following L0 Ancillary. Step 1 requires 2 preceding data sets, one ephemeris product and one attitude product from the preceding Step 1 run (two granules, one for ephemeris and one for attitude).

Step 1 DPREP Specifications

Tar File Name : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_Ancillary_DPREP.tar
Contents : EcDpPrAm1EdosEphAttDPREP_PGE

EcDpPrAm1EdosAncillary
EcDpPrAm1EdosEphemerisRepair
EcDpPrAm1ToolkitToHdf
PGS_101

PGE Met File : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_Ancillary_DPREP.tar.met

Top Level Exe : EcDpPrAm1EdosEphAttDPREP_PGE

PGE Name : AM1Eph
Version Number : 30001

Profile 1 ODL : PGE_AM1Eph#3000101.odl
Profile 2 ODL : PGE_AM1Eph#3000102.odl
Profile 3 ODL : PGE_AM1Eph#3000103.odl
Profile 4 ODL : PGE_AM1Eph#3000104.odl

Test Scenario

The Step 1 DPREP test is divided into two parts. The first part exercises Profiles 1, 2, and 3 over four consecutive data segments. The first data segment is scheduled for Profile 2 processing, the next two segments are scheduled for Profile 1 processing, and the last scheduled for Profile 3 processing. The data segments to be processed follow.

Segment 1: 1997 July 31 04:00:00 to 1997 July 31 06:00:00 (Profile 2)
Segment 2: 1997 July 31 06:00:00 to 1997 July 31 08:00:00 (Profile 1)
Segment 3: 1997 July 31 08:00:00 to 1997 July 31 10:00:00 (Profile 1)
Segment 4: 1997 July 31 10:00:00 to 1997 July 31 12:00:00 (Profile 3)

The second Step 1 DPREP test exercises Profile 4 on an isolated data segment.

Segment 5: 1997 July 31 18:00:00 to 1997 July 31 20:00:00 (Profile 4)

There is one dynamic granule (2 files for each granule) for each of the data segments described above. Insert all five dynamic granules into the archive using the SSIT Manager's insert dynamic data tool.

Segment 1:

ESDT Short Name : AM1ANC
File Names : EDOS_LEVEL_ZERO_00.Profile_2
EDOS_LEVEL_ZERO_01.Profile_2
Met file : AM1ANC.Profile_2.met
Time Range : 1997 July 31 04:00:00 to 1997 July 31 06:00:00

Segment 2:

ESDT Short Name : AM1ANC
File Names : EDOS_LEVEL_ZERO_00.Profile_1.A
EDOS_LEVEL_ZERO_01.Profile_1.A
Met file : AM1ANC.Profile_1.A.met
Time Range : 1997 July 31 06:00:00 to 1997 July 31 08:00:00

Segment 3:
ESDT Short Name : AM1ANC
File Names : EDOS_LEVEL_ZERO_00.Profile_1.B
EDOS_LEVEL_ZERO_01.Profile_1.B
Met file : AM1ANC.Profile_1.B.met
Time Range : 1997 July 31 08:00:00 to 1997 July 31 10:00:00

Segment 4:
ESDT Short Name : AM1ANC
File Names : EDOS_LEVEL_ZERO_00.Profile_3
EDOS_LEVEL_ZERO_01.Profile_3
Met file : AM1ANC.Profile_3.met
Time Range : 1997 July 31 10:00:00 to 1997 July 31 12:00:00

Segment 5:
ESDT Short Name : AM1ANC
File Names : EDOS_LEVEL_ZERO_00.Profile_4
EDOS_LEVEL_ZERO_01.Profile_4
Met file : AM1ANC.Profile_4.met
Time Range : 1997 July 31 18:00:00 to 1997 July 31 20:00:00

Create three production requests for the following time ranges. The number of DPRs that should be created are given below.

Request	Timespan	Profile	DPRs
1	1997 July 31 04:00:00 to 06:00:00	2	1
2	1997 July 31 06:00:00 to 10:00:00	1	2
3	1997 July 31 10:00:00 to 12:00:00	3	1

Schedule all three production requests to run as a single production plan. This completes the first part of the Step 1 DPREP test.

Next, create one production request that for the time range that follow.

Request	Timespan	Profile	DPRs
1	1997 July 31 18:00:00 to 20:00:00	4	1

Schedule this a separate production plan. This completes the second part of the Step 1 DPREP test.

Results

Find output granules for each of the timespans described above for ESDTs AM1EPHN0, AM1EPHH0, AM1ATTN0, and AM1ATTH0. ESDT AM1EPHN0 is the Toolkit format ephemeris granule, while ESDT AM1EPHH0 is the HDF format ephemeris granule. ESDT AM1ATTN0 is the Toolkit format attitude granule, while ESDT AM1ATTH0 is the HDF format attitude granule.

Step 2 DPREP Test Instructions

For Step 2, current and following granules are FDD-supplied attitude (ShortName AM1ATTF), the preceding granule is the output Toolkit format attitude product produced by the preceding run of Step 2 (ShortName AM1ATTNF).

Step 2 DPREP Specifications
Tar File Name : $ECS_HOME/<MODE>/CUSTOM/data/DPS/FDDAttitude.tar
Contents : EcDpPrAm1FddAttitudeDPREP_PGE PGS_101
PGE Met File : $ECS_HOME/<MODE>/CUSTOM/data/DPS/FDDAttitude.tar.met
Top Level Exe : EcDpPrAm1FddAttitudeDPREP_PGE
PGE Name : FddAtt
Version Number : 30001
Profile 1 ODL : PGE_FddAtt#3000101.odl
Profile 2 ODL : PGE_FddAtt#3000102.odl
Profile 3 ODL : PGE_FddAtt#3000103.odl
Profile 4 ODL : PGE_FddAtt#3000104.odl

Test Scenario

The Step 2 DPREP test, like the Step 1 test scenario, is divided into two parts. The first part exercises Profiles 1, 2, and 3 over four consecutive data segments. The first data segment is scheduled for Profile 2 processing, the next two segments are scheduled for Profile 1 processing, and the last scheduled for Profile 3 processing. The data segments to be processed follow.

Segment 1: 1997 July 31 04:00:00 to 1997 July 31 06:00:00 (Profile 2)
Segment 2: 1997 July 31 06:00:00 to 1997 July 31 08:00:00 (Profile 1)
Segment 3: 1997 July 31 08:00:00 to 1997 July 31 10:00:00 (Profile 1)
Segment 4: 1997 July 31 10:00:00 to 1997 July 31 12:00:00 (Profile 3)

The second Step 2 DPREP test exercises Profile 4 on an isolated data segment.

Segment 5: 1997 July 31 18:00:00 to 1997 July 31 20:00:00 (Profile 4)

There is one granule for each of the data segments described above. Insert all five granules into the archive using the SSIT Manager's insert dynamic data tool.

Segment 1:
ESDT Short Name : AM1ATTF
File Names : AM1_DEFINITIVE_ATT.fdd.Profile_2
Met file : AM1_DEFINITIVE_ATT.fdd.Profile_2.met
Time Range : 1997 July 31 04:00:00 to 1997 July 31 06:00:00

Segment 2:
ESDT Short Name : AM1ATTF
File Names : AM1_DEFINITIVE_ATT.fdd.Profile_1.A
Met file : AM1_DEFINITIVE_ATT.fdd.Profile_1.A.met
Time Range : 1997 July 31 06:00:00 to 1997 July 31 08:00:00

Segment 3:
ESDT Short Name : AM1ATTF
File Names : AM1_DEFINITIVE_ATT.fdd.Profile_1.B
Met file : AM1_DEFINITIVE_ATT.fdd.Profile_1.B.met
Time Range : 1997 July 31 08:00:00 to 1997 July 31 10:00:00

Segment 4:

ESDT Short Name : AM1ATTF
File Names : AM1_DEFINITIVE_ATT.fdd.Profile_3
Met file : AM1_DEFINITIVE_ATT.fdd.Profile_3.met
Time Range : 1997 July 31 10:00:00 to 1997 July 31 12:00:00
Segment 5:
ESDT Short Name : AM1ATTF
File Names : AM1_DEFINITIVE_ATT.fdd.Profile_4
Met file : AM1_DEFINITIVE_ATT.fdd.Profile_4.met
Time Range : 1997 July 31 18:00:00 to 1997 July 31 20:00:00

Create three production requests for the following time ranges. The number of DPRs that should be created are given below.

Request	Timespan	Profile	DPRs
1	1997 July 31 04:00:00 to 06:00:00	2	1
2	1997 July 31 06:00:00 to 10:00:00	1	2
3	1997 July 31 10:00:00 to 12:00:00	3	1

Schedule all three production requests to run as a single production plan. This completes the first part of the Step 2 DPREP test.

Next, create one production request that for the time range that follow.

Request	Timespan	Profile	DPRs
1	1997 July 31 18:00:00 to 20:00:00	4	1

Schedule this a separate production plan. This completes the second part of the Step 2 DPREP test.

These plans can be scheduled and run completely independently of the Step 1 DPREP plans described earlier.

Results

Find output granules for each of the timespans described above for ESDTs AM1ATTNF and AM1ATTHF. ESDT AM1ATTNF is the Toolkit format attitude granule, while ESDT AM1ATTHF is the HDF format attitude granule.

Step 3 DPREP Test Instructions

For Step 3, the current granule is FDD-supplied ephemeris (ShortName AM1EPHF), the preceding granule is the output Toolkit format ephemeris product produced by preceding run of Step 1, and the following granule is EDOS-supplied L0 Ancillary (ShortName AM1ANC).

Step 3 DPREP Specifications
Tar File Name : $ECS_HOME/<MODE>/CUSTOM/data/DPS/ReplaceEphemeris.tar
Contents : EcDpPrAm1FddAttitudeDPREP_PGE
PGS_101
PGE Met File : $ECS_HOME/<MODE>/CUSTOM/data/DPS/ReplaceEphemeris.tar.met

Top Level Exe : EcDpPrAm1FddEphemerisDPREP_PGE

PGE Name : RepEph
Version Number : 30001

Profile 1 ODL : PGE_RepEph#3000101.odl
Profile 2 ODL : PGE_RepEph#3000102.odl
Profile 3 ODL : PGE_RepEph#3000103.odl
Profile 4 ODL : PGE_RepEph#3000104.odl
Test Scenario
Operationally, Step 3 AM-1 DPREP will most likely be scheduled to run on single data segments. Hence, the Step 3 test scenario consists of scheduling and running four plans that exercise the four Profiles individually on a single data segments. The data segments to be processed follow.

Segment 1: 1997 July 31 04:00:00 to 1997 July 31 06:00:00 (Profile 2)
Segment 2: 1997 July 31 08:00:00 to 1997 July 31 10:00:00 (Profile 1)
Segment 3: 1997 July 31 10:00:00 to 1997 July 31 12:00:00 (Profile 3)
Segment 4: 1997 July 31 18:00:00 to 1997 July 31 20:00:00 (Profile 4)
There is one granule for each of the data segments described above. Insert all four granules into the archive using the SSIT Manager's insert dynamic data tool.
Segment 1:
ESDT Short Name : AM1EPHF
File Names : AM1_REPAIR_EPH.fdd.Profile_2
Met file : AM1_REPAIR_EPH.fdd.Profile_2.met
Time Range : 1997 July 31 04:00:00 to 1997 July 31 06:00:00
Segment 2:
ESDT Short Name : AM1EPHF
File Names : AM1_REPAIR_EPH.fdd.Profile_1
Met file : AM1_REPAIR_EPH.fdd.Profile_1.met
Time Range : 1997 July 31 08:00:00 to 1997 July 31 10:00:00
Segment 3:
ESDT Short Name : AM1EPHF
File Names : AM1_REPAIR_EPH.fdd.Profile_3
Met file : AM1_REPAIR_EPH.fdd.Profile_3.met
Time Range : 1997 July 31 10:00:00 to 1997 July 31 12:00:00
Segment 4:
ESDT Short Name : AM1EPHF
File Names : AM1_REPAIR_EPH.fdd.Profile_4
Met file : AM1_REPAIR_EPH.fdd.Profile_4.met
Time Range : 1997 July 31 18:00:00 to 1997 July 31 20:00:00
Create four individual production plans consisting of a single DPR for each of the time ranges that follow. Schedule and allow a given plan to complete before

scheduling the next.

Request	Timespan	Profile	DPRs
1	1997 July 31 04:00:00 to 06:00:00	2	1
2	1997 July 31 06:00:00 to 10:00:00	1	1
3	1997 July 31 10:00:00 to 12:00:00	3	1
4	1997 July 31 18:00:00 to 20:00:00	4	1

These plans can be scheduled and run only after the Step 1 AM-1 DPREP test has been successfully completed. The Step 1 DPREP test scenario must be completed before CONTINUING with the Step 3 test scenario. Continuation of the Step 3 test scenario requires the archive environment that was produced from the successful completion of the Step 1 DPREP test scenario.

Results

Find output granules for each of the timespans described above for ESDTs AM1EPHN0 and AM1EPHH0. ESDT AM1EPHN0 is the Toolkit format ephemeris granule, while ESDT AM1EPHH0 is the HDF format ephemeris granule. Output from Step 3 processing is placed in the same ESDTs as the Step 1 products.

Step1 DPREP directions used from earlier tests:

There are 2 profiles for DPREP. Profile 1 takes in previous DPREP output and is expected to be run most of the time at the DAACs. Profile 2 takes in only the AM1 Ancillary data and will be run only for the first run of DPREP (because there is no previous output). Both profiles should be registered and executed.

PGENAME : AM1Eph
PGEVERSION : 2.0
PROFILE : 1
TopLevelShellName : EcDpPrAm1EdosEphAttDPREP_PGE
PGENAME : AM1Eph
PGEVERSION : 2.0
PROFILE : 2
TopLevelShellName : EcDpPrAm1EdosEphAttDPREP_PGE (same executable, only insert it once)
PGE Tar file location : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_Ancillary_DPREP.tar
PGE Met file location : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_Ancillary_DPREP.tar.met
Science Software Release 6A

The PGE tar file contains the executables: EcDpPrAm1EdosEphAttDPREP_PGE, EcDpPrAm1EdosAncillary, EcDpPrAm1EdosEphemerisRepair, EcDpPrAm1ToolkitHdf.and the toolkit message file PGS_101

This PGE has no static data. This means Insert Static from SSIT can be skipped for this PGE.

However, there are 4 dynamic granules (2 files each) that this PGE needs as input. These 4 granules can be inserted from SSIT **Insert Test Dynamic Tool**

Granule 1:
ESDT Short Name : AM1ANC
Version : 001

Multi File granule : Y
Directory : $ECS_HOME/<MODE>/CUSTOM/data/DPS
File Names : EDOS_LEVEL_ZERO_00.curr, EDOS_LEVEL_ZERO_01.curr
Met file : AM1ANC.curr.met
Granule 2:
ESDT Short Name : AM1ANC
Version : 001
Multi File granule : Y
Directory : $ECS_HOME/<MODE>/CUSTOM/data/DPS
File Names : EDOS_LEVEL_ZERO_00.next, EDOS_LEVEL_ZERO_01.next
Met file : AM1ANC.next.met
Granule 3:
ESDT Short Name : AM1ANC
Version : 001
Multi File granule : Y
Directory : $ECS_HOME/<MODE>/CUSTOM/data/DPS
File Names : EDOS_LEVEL_ZERO_00.next_1, EDOS_LEVEL_ZERO_01.next_1
Met file : AM1ANC.next_1.met
Granule 4:
ESDT Short Name : AM1ANC
Version : 001
Multi File granule : Y
Directory : $ECS_HOME/<MODE>/CUSTOM/data/DPS
File Names : EDOS_LEVEL_ZERO_00.next_2, EDOS_LEVEL_ZERO_01.next_2
Met file : AM1ANC.next_2.met
3 PRs need to be created, two for Profile 1 and one for Profile 2. Create the PR for Profile 2 first (it runs on earlier data) and then create the PR for Profile 1.

The time frame for Production request editor for Profile 2 is 1998 June 30 00:00:00 to 1998 June 30 02:00:00

The time frame for Production request editor for Profile 1 is 1998 June 30 02:00:00 to 1998 June 30 04:00:00

The time frame for Production request editor for the second Profile 1 is 1998 June 30 04:00:00 to 1998 June 30 06:00:00

Step2 DPREP directions:
****
Step 2 DPREP can run ONLY after all three PRs for Step 1 DPREP have completed
****
There are 2 profiles for Step 2 DPREP. Profile 1 takes in previous Step 2 DPREP output and is expected to be run most of the time at the DAACs. Profile 2 takes in only the Fdd Att data and will be run only for the first run of DPREP (because there is no previous output). Both profiles should be registered and executed.
PGENAME : FddAtt
PGEVERSION : 1.0
PROFILE : 1

TopLevelShellName : EcDpPrAm1FddAttitudeDPREP_PGE
PGENAME : FddAtt
PGEVERSION : 1.0
PROFILE : 2
TopLevelShellName : EcDpPrAm1FddAttitudeDPREP_PGE (same executable, only insert it once)
PGE Tar file location : $ECS_HOME/<MODE>/CUSTOM/data/DPS/FDDAttitude.tar
PGE Met file location : $ECS_HOME/<MODE>/CUSTOM/data/DPS/FDDAttitude.tar.met
Science Software Version: 1.0
The PGE tar file contains the executable EcDpPrAm1FddAttitudeDPREP_PGE and the toolkit message file PGS_101
This PGE has no static data. This means Insert Static from SSIT can be skipped for this PGE.
However, there are 3 dynamic granules that this PGE needs as input. These 3 granules can be inserted from SSIT insert test dynamic tool
Granule 1:
ESDT Short Name : AM1ATTF
Version : 001
Multi File granule : N
File Name : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_DEFINITIVE_ATT.fdd.curr
Met file : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_DEFINITIVE_ATT.fdd.curr.met
Granule 2:
ESDT Short Name : AM1ATTF
Version : 001
Multi File granule : N
File Name : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_DEFINITIVE_ATT.fdd.next
Met file : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_DEFINITIVE_ATT.fdd.next.met
Granule 3:
ESDT Short Name : AM1ATTF
Version : 001
Multi File granule : N
File Name : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_DEFINITIVE_ATT.fdd.next_1
Met file : $ECS_HOME/<MODE>/CUSTOM/data/DPS/AM1_DEFINITIVE_ATT.fdd.next_1.met
2 PRs need to be created, one for Profile 1 and one for Profile 2. Create the PR for Profile 2 first (it runs on earlier data) and then create the PR for Profile 1.

The time frame for Production request editor for Profile 2 is 1998 June 30 02:00:00 to 1998 June 30 04:00:00

The time frame for Production request editor for Profile 1 is 1998 June 30 04:00:00 to 1998 June 30 06:00:00

Note: The Data files and tar files are delivered only to the Science Data Server and cannot be directly accessed by SSIT. In order to insert these files they must be moved to the SSIT server manually.

DPREP Step1 profile1 PCF

PRODUCT INPUT FILES
##
(**Initial Construction Record**:)
1000|P0420004AAAAAAAAAAAAAA98300080000000.PDS|/net/p0drg01/dss_amass/testdata/instrument_data/AM1/EDOS_HK_ANC/EDOSL0_ANC_PDS_0002||<A) insert UR here>||2
(**Current 2 hour Data File**)
1000|P0420004AAAAAAAAAAAAAA98300080000001.PDS|/net/p0drg01/dss_amass/testdata/instrument_data/AM1/EDOS_HK_ANC/EDOSL0_ANC_PDS_0002||<B) insert UR here>||1
: (**Future 2 hour Construction Record**)
1010|P0420004AAAAAAAAAAAAAA98300100000000.PDS|/net/p0drg01/dss_amass/testdata/instrument_data/AM1/EDOS_HK_ANC/EDOSL0_ANC_PDS_0002||<C)
insert UR here>||2
(**Future 2 hour Data File**)
1010|P0420004AAAAAAAAAAAAAA98300100000001.PDS|/net/p0drg01/dss_amass/testdata/instrument_data/AM1/EDOS_HK_ANC/EDOSL0_ANC_PDS_0002||<D)
insert UR here>||1

The last ephemeris/attitude data sets generated. **(Found in PGE template)**

Ephemeris (Toolkit native format) **(Previous Time Range)**
1500|EOS_AM1_Ephemeris.21206.step1b.eph|/home/cmts1/DPREP/demooutput||<E) insert UR here>||1
Attitude (Toolkit native format)
1502|EOS_AM1_Attitude.21206.step1.att|/home/cmts1/DPREP/demooutput||<F) insert UR here>||1
:
? PRODUCT OUTPUT FILES
##
(Current Time Range)

Toolkit attitude datasets (output of EcDpPrAm1EdosAncillary).

1001|EOS_AM1_Attitude.21208.step1.att|/home/cmts1/DPREP/demooutput||||1
1100|EOS_AM1_Attitude.21208.step1.hdf|/home/cmts1/DPREP/demooutput||||1

Toolkit ephemeris datasets (output of EcDpPrAm1EdosAncillary).

1102|EOS_AM1_Ephemeris.21208.hdf|/home/cmts1/DPREP/demooutput||||1

Gap-filled HDF ephemeris dataset (output of EcDpPrAm1EphemerisGapFill).

```
# ----------------------------------------------------------------------
1151|EOS_AM1_Ephemeris.21208.step1b.eph|/home/cmts1/DPREP/demooutput||||1
# ----------------------------------------------------------------------
# Gap-filled Toolkit ephemeris dataset (output of
# EcDpPrAm1FormatNativeEphemeris).
# ----------------------------------------------------------------------
1152|EOS_AM1_Ephemeris.21208.step1b.hdf|/home/cmts1/DPREP/demooutput||||1
:
SUPPORT OUTPUT FILES
##########################################################################
10100|LogStatus|/home/cmts1/DPREP/logs||||1
10101|LogReport|/home/cmts1/DPREP/logs||||1
10102|LogUser|/home/cmts1/DPREP/logs||||1
10103|TmpStatus|||||1
10104|TmpReport|||||1
10105|TmpUser|||||1
10110|MailFile|||||1
:
?   USER DEFINED RUNTIME PARAMETERS
##########################################################################
999|No Previous Data Set; 1=true, 0=false.|0          (Profile 1 set to 0, Profile 2 set to 1)
# ----------------------------------------------------------------------
# Toolkit version for which this PCF was intended.
# DO NOT REMOVE THIS VERSION ENTRY!
# ----------------------------------------------------------------------
10220|Toolkit version string|DAAC B.0 TK5.2.4     (Caution:  May need to change to higher
level Toolkit)
END
```

DPREP Step1 profile2 PCF (Profile 2 is start up PCF)

```
PRODUCT INPUT FILES
##########################################################################
1000|P0420004AAAAAAAAAAAAAAA98300060000000.PDS|/net/p0drg01/dss_amass/testdata/
instrument_data/AM1/EDOS_HK_ANC/EDOSL0_ANC_PDS_0002||<A) insert UR here>||2
1000|P0420004AAAAAAAAAAAAAAA98300060000001.PDS|/net/p0drg01/dss_amass/testdata/
instrument_data/AM1/EDOS_HK_ANC/EDOSL0_ANC_PDS_0002||<B) insert UR here>||1
1010|P0420004AAAAAAAAAAAAAAA98300080000000.PDS|/net/p0drg01/dss_amass/testdata/
instrument_data/AM1/EDOS_HK_ANC/EDOSL0_ANC_PDS_0002||<C) insert UR here>||2
1010|P0420004AAAAAAAAAAAAAAA98300080000001.PDS|/net/p0drg01/dss_amass/testdata/
instrument_data/AM1/EDOS_HK_ANC/EDOSL0_ANC_PDS_0002||<D) insert UR here>||1
# ----------------------------------------------------------------------
# The last ephemeris/attitude data sets generated.
# ----------------------------------------------------------------------
# Ephemeris (Toolkit native format)
#1500|EOS_AM1_Ephemeris.prev.eph|.||<E) insert UR here>||1
```

```
# Attitude (Toolkit native format)
#1502|EOS_AM1_Attitude.21206.step1.att|/home/cmts1/DPREP/output||<F) insert UR here>||1
:
?   PRODUCT OUTPUT FILES
##############################################################################
# ---------------------------------------------------------------------------
# Toolkit attitude datasets (output of EcDpPrAm1EdosAncillary).
# ---------------------------------------------------------------------------
1001|EOS_AM1_Attitude.21206.step1.att|/home/cmts1/DPREP/demooutput||||1
1100|EOS_AM1_Attitude.21206.step1.hdf|/home/cmts1/DPREP/demooutput||||1
# ---------------------------------------------------------------------------
# Toolkit ephemeris datasets (output of EcDpPrAm1EdosAncillary).
# ---------------------------------------------------------------------------
1102|EOS_AM1_Ephemeris.21206.hdf|/home/cmts1/DPREP/demooutput||||1
# ---------------------------------------------------------------------------
# Gap-filled HDF ephemeris dataset (output of EcDpPrAm1EdosEphemerisRepair).
# ---------------------------------------------------------------------------
1151|EOS_AM1_Ephemeris.21206.step1b.eph|/home/cmts1/DPREP/demooutput||||1
# ---------------------------------------------------------------------------
# Gap-filled Toolkit ephemeris dataset (output of EcDpPrAm1FormatNativeEphemeris).
# ---------------------------------------------------------------------------
1152|EOS_AM1_Ephemeris.21206.step1b.hdf|/home/cmts1/DPREP/demooutput||||1
?   SUPPORT OUTPUT FILES
##############################################################################
10100|LogStatus|/home/cmts1/DPREP/logs||||1
10101|LogReport|/home/cmts1/DPREP/logs||||1
10102|LogUser|/home/cmts1/DPREP/logs||||1
10103|TmpStatus|||||1
10104|TmpReport|||||1
10105|TmpUser|||||1
10110|MailFile|||||1
:
?   USER DEFINED RUNTIME PARAMETERS
##############################################################################
999|No Previous Data Set; 1=true, 0=false.|1
# ---------------------------------------------------------------------------
# Toolkit version for which this PCF was intended.
# DO NOT REMOVE THIS VERSION ENTRY!
# ---------------------------------------------------------------------------
10220|Toolkit version string|DAAC B.0 TK5.2.4
?   END
```

DPREP Step2 profile1 PCF

```
PRODUCT INPUT FILES
##########################################################################
1000|AM1_DEFATT_080000_212_1997_01.FDD|/home/cmts1/DPREP/am1defatt||<A)     insert
UR here>||1
:
1010|AM1_DEFATT_100000_212_1997_01.FDD|/home/cmts1/DPREP/am1defatt||<B)     insert
UR here>||1
# ----------------------------------------------------------------------
# The last ephemeris/attitude data sets generated.
# ----------------------------------------------------------------------
# Attitude (Toolkit native format)
1502|EOS_AM1_Attitude.21206.step2.att|/home/cmts1/DPREP/demooutput||<C)     insert     UR
here>||1
:
#1400|AM1EPHMH.mcf|.||||1
1401|AM1ATTHF.MCF|/home/cmts1/DPREP/mcf||||1
#1402|AM1EPHMN.mcf|.||||1
1403|AM1ATTNF.MCF|/home/cmts1/DPREP/mcf||||1
10250|MCF|||||1
10252|GetAttr.temp|||||1
10254|MCFWrite.temp|||||1
:
10501|EOS_AM1_Ephemeris.21208.step1b.eph|/home/cmts1/DPREP/demooutput||||2
10501|EOS_AM1_Ephemeris.21210.step1b.eph|/home/cmts1/DPREP/demooutput||||1
10502|INSERT_ATTITUDE_FILES_HERE|||||1
:
?   PRODUCT OUTPUT FILES
##########################################################################
:1001|EOS_AM1_Attitude.21208.step2.att|/home/cmts1/DPREP/demooutput||||1
1100|EOS_AM1_Attitude.21208.step2.hdf|/home/cmts1/DPREP/demooutput||||1
:
?   USER DEFINED RUNTIME PARAMETERS
##########################################################################
:
999|First Mission Data Set; 1=true, 0=false.|0
:10220|Toolkit version string|DAAC B.0 TK5.2.4
:END
```

DREP Step2 profile2 PCF

```
PRODUCT INPUT FILES
##########################################################################
1000|AM1_DEFATT_060000_212_1997_01.FDD|/home/cmts1/DPREP/am1defatt||<A)     insert
UR here>||1
```

```
:
1010|AM1_DEFATT_080000_212_1997_01.FDD|/home/cmts1/DPREP/am1defatt||<B)     insert
UR here>||1
# ----------------------------------------------------------------------
# The last ephemeris/attitude data sets generated.
# ----------------------------------------------------------------------
# Attitude (Toolkit native format)
1502|EOS_AM1_Attitude.21206.step1.att|/home/cmts1/DPREP/demooutput||<C)     insert     UR
here>||1
:
1400|AM1EPHMH.mcf|.||||1
1401|AM1ATTHF.MCF|/home/cmts1/DPREP/mcf||||1
1402|AM1EPHMN.mcf|.||||1
1403|AM1ATTNF.MCF|/home/cmts1/DPREP/mcf||||1
10250|MCF|||||1
10252|GetAttr.temp|||||1
10254|MCFWrite.temp|||||1
:
10501|EOS_AM1_Ephemeris.21206.step1b.eph|/home/cmts1/DPREP/demooutput||||2
10501|EOS_AM1_Ephemeris.21208.step1b.eph|/home/cmts1/DPREP/demooutput||||1
10502|INSERT_ATTITUDE_FILES_HERE|||||1
:
?   PRODUCT OUTPUT FILES
##########################################################################
1001|EOS_AM1_Attitude.21206.step2.att|/home/cmts1/DPREP/demooutput||||1
1100|EOS_AM1_Attitude.21206.step2.hdf|/home/cmts1/DPREP/demooutput||||1
:
?   USER DEFINED RUNTIME PARAMETERS
##########################################################################
:999|First Mission Data Set; 1=true, 0=false.|1
:
10220|Toolkit version string|DAAC B.0 TK5.2.4
:
?   END
```

Setups for DPREP

```
set path = ( /usr/ecs/TS1/CUSTOM/TOOLKIT/toolkit/bin/sgi_daac_f90 $path )
setenv PGS_PC_INFO_FILE /home/cmts1/DPREP/pcf/Step1DP21206.PCF
setenv PGSMSG /home/cmts1/DPREP/msg
setenv PGSHOME /usr/ecs/TS1/CUSTOM/toolkit
setenv PGS_PC_INFO_FILE /home/cmts1/DPREP/pcf/Step1DP21208.PCF
setenv PGSMSG /home/cmts1/DPREP/msg
setenv PGSHOME /usr/ecs/TS1/CUSTOM/toolkit
setenv PGS_PC_INFO_FILE /home/cmts1/DPREP/pcf/Step1DP21210.PCF
```

setenv PGSMSG /home/cmts1/DPREP/msg
setenv PGSHOME /usr/ecs/TS1/CUSTOM/toolkit
setenv PGS_PC_INFO_FILE /home/cmts1/DPREP/pcf/Step2DP21206.PCF
setenv PGSMSG /home/cmts1/DPREP/msg
setenv PGSHOME /usr/ecs/TS1/CUSTOM/toolkit
setenv PGS_PC_INFO_FILE /home/cmts1/DPREP/pcf/Step2DP21208.PCF
setenv PGSMSG /home/cmts1/DPREP/msg

setenv PGSHOME /usr/ecs/TS1/CUSTOM/toolkit

Note: The **setenv** parameters are used for the **Command Line PGE** test. Only the **(*)** are used for **PDPS PGE** runs.

DPREP Considerations

As previously mentioned DPREP (data preprocessing) consists of sets of PGEs that use a statistical approach to convert Level 0 (L0) attitude and ephemeris ancillary data for a particular satellite (e.g., Terra, Aqua) into SDP Toolkit native binary format without altering or modifying the scientific content of the data sets.
DPREP PGEs are supplied by ECS, unlike most PGEs, which are provided by the Science Computing Facilities that ECS supports. Release 5B DPREP supports Terra operations and Aqua SSI&T.

Terra DPREP

Terra DPREP consists of the following three PGEs:

AM1Eph or Step 1 (EcDpPrAm1EdosEphAttDPREP_PGE), a ksh script that serves as a driver for the following three executables:

- EcDpPrAm1EdosAncillary
- EcDpPrAm1EdosEphemerisRepair
- EcDpPrAm1ToolkitToHdf

FddAtt or Step 2 (EcDpPrAm1FddAttitudeDPREP_PGE).

RepEph or Step 3 (EcDpPrAm1FddEphemerisDPREP_PGE).

There are several sources of information on the Terra DPREP PGEs and how to run them:

184-TP-001-001, Terra Spacecraft Ephemeris and Attitude Data Preprocessing.

611-CD-510-001, Mission Operation Procedures for the ECS Project, Chapter 11.

Two files installed on the science processor hosts (e.g., e0spg01, g0spg01, l0spg01) in the /usr/ecs/*MODE*/CUSTOM/data/DPS directory.

- “DPREP_README”
- “HowtoRunDPREP”

The Terra DPREP PGEs process Level 0 Terra (AM-1) spacecraft data (e.g., ESDT AM1ANC) provided by EDOS. The output files/granules of the DPREP PGEs are subsequently used in the

processing of data from various instruments on the satellite. They provide the following types of ancillary (non-science) data:

Ephemeris.

- Spacecraft location: ephemeris (or orbit) data include: latitude, longitude, and height.

Attitude.

- Orientation of the satellite, including yaw, pitch, and roll angles; and angular rates about three axes.

AM1Eph (Step 1 DPREP: EDOS Level 0 Ancillary Data)

EcDpPrAm1EdosAncillary reads in an AM-1 L0 (EDOS) ancillary data set (ESDT AM1ANC). It also reads another set of AM-1 L0 (EDOS) ancillary data set. The second set of L0 data is required to ensure that incomplete orbits in the first data set get complete orbit metadata records. The only data that is extracted from the second data set is the descending node time and longitude. EcDpPrAm1EdosAncillary also reads in ephemeris and attitude data (ESDT AM1EPHN0 and ESDT AM1ATTN0). These would be the last ephemeris/attitude data sets generated from a previous run of the PGE.

If EcDpPrAm1EdosAncillary signals that a short gap was detected, EcDpPrAm1EdosEphemerisRepair reads the scratch file created by EcDpPrAm1EdosAncillary, fills the gap, and writes a gap-filled native-format ephemeris file.

EcDpPrAm1ToolkitToHdf takes the native format ephemeris file and produces a corresponding HDF file and a metadata file.

The PGE produces Toolkit- and HDF-format attitude (ESDTs AM1ATTN0 and AM1ATTH0) and ephemeris (ESDTs AM1EPHN0 and AM1EPHH0) data sets.

NOTE: The Level 0 (input) data sets represent two-hour periods of time but processing cannot be performed on the most recently available two-hour data set. Step 1 processing needs to look forward in the time stream in order to complete orbit metadata processing.

FddAtt (Step 2 DPREP: Definitive Attitude Data)

EcDpPrAm1FddAttitudeDPREP_PGE reads in both the current Flight Dynamics Division (FDD) attitude data set and the next FDD attitude data set. It also reads in the attitude data set it produced with its last run. The output of the process is a native-format attitude file (ESDT AM1ATTNF) and an HDF-format attitude file (ESDT AM1ATTHF). A metadata file is produced for each data file.

RepEph (Step 3 DPREP: Repaired Ephemeris Data)

If Step 1 finds too many missing data points in the ephemeris data (e.g., AM1EPHH0 and AM1EPHN0 granules have gaps of greater than 60 seconds), it signals that there are problems with the data. The Production Planner receives an e-mail message indicating the problems in the preprocessed Terra ephemeris granules and notifies the FDD (e.g., by telephone) that a repaired ephemeris file (AM1EPHF) is needed for the time span of the granule that has the gap.

When the repaired ephemeris file has been ingested, EcDpPrAm1FddEphemerisDPREP_PGE reads in the repaired ephemeris data set and the EOS_AM1 ephemeris data in toolkit native format. It produces replacement ephemeris data sets (AM1EPHH0 and AM1EPHN0).
Operationally, Steps 1 and 2 are scheduled daily and run independently of one another. Step 3 is scheduled and run on an as-needed basis.

Terra DPREP Profiles

As previously mentioned DPREP processing has data requirements beyond the current two-hour segment. Data from the preceding and following segments are used for performing consistency checks on the ephemeris and attitude data streams when the data streams bridge segment boundaries. However, there is no guarantee that data from the preceding and following segments will always be available. Consequently, four data processing profiles have been developed for each of the three DPREP steps to accommodate the various permutations of data availability:

Profile 1 is used when data are available from the preceding, current, and following segments.

Profile 2 is used when data are available from the current and following segments only.

Profile 3 is used when data are available from the preceding and current segments only.

Profile 4 is used when data are available from the current segment only.

Profile 1 is used for nominal DPREP operation. It is the profile of each DPREP step that is run on a routine basis.
Profile 2 (no preceding data, but following data is available) initializes DPREP's processing of a given step's ephemeris and/or attitude data stream. When Profile 2 has been run on a data segment, Profile 1 (preceding and following data available) assumes processing responsibility on all data segments thereafter until data dropout or mission end is encountered.
Profile 3 (preceding data available, but no following data) processes the data segment that immediately precedes data dropout and, therefore, terminates processing on a given step's ephemeris and/or attitude data stream.
Profile 4 is used for processing isolated data segments and is not likely to be scheduled operationally.
In the big picture of the mission, DPREP processing on the very first data segment would require running Profile 2. The next data segments would be processed using Profile 1 processes. The very last data segment of the mission would be processed using Profile 3.

Aqua DPREP

There is a slide presentation at this URL below that more clearly shows the files associated with Aqua processing. http://dmserver.gsfc.nasa.gov/ecsdev/relb/pdps/dprep/step1/sld001.htm

For Release 5B Aqua DPREP consists of the following PGEs:

PM1DefEph or PM1 Definitive Orbit PGE (EcDpPrPM1DefOrbitDPREP_PGE).

PM1PreEph or PM1 Predictive Orbit PGE (EcDpPrPM1PreOrbitDPREP_PGE).

One source of information on the Aqua DPREP PGEs and how to run them is the "HowtoRunPm1DPREP" file installed on the science processor hosts (e.g., e0spg01, g0spg01, l0spg01) in the /usr/ecs/*MODE*/CUSTOM/data/DPS directory.

PM1DefEph and PM1PreEph (EDOS Level 0 Ancillary Data)

The Aqua PM1DefEph and PM1PreEph PGEs process Aqua ancillary data (i.e., PM1EPHD, definitive ephemeris data and PM1EPHP, predictive ephemeris data) supplied by EDOS. The output files/granules of the Aqua DPREP PGEs provide satellite ephemeris data and are subsequently used in the processing of data from various instruments on the Aqua satellite.

The PM1 Definitive Orbit PGE reads in both the current PM1EPHD definitive ephemeris data set for Aqua and a previous PM1EPHND preprocessed Aqua platform definitive ephemeris data set in native format. The outputs of the process are preprocessed Aqua platform definitive ephemeris data sets in native format (PM1EPHND) and HDF format (PM1EPHHD).

The PM1 Predictive Orbit PGE reads in both the current PM1EPHP predictive ephemeris data set for Aqua and a previous PM1EPHND preprocessed Aqua platform definitive ephemeris data set in native format. The outputs of the process are preprocessed Aqua platform predictive ephemeris data sets in native format (PM1EPHNP) and HDF format (PM1EPHHP).

Aqua DPREP Profiles

Because there is no guarantee that data from the preceding and following segments will always be available, two data processing profiles have been developed for each of the Aqua DPREP PGEs to accommodate variations in data availability:

Profile 1 is used when both current and previous data sets are available.

Profile 2 is used the current data set only is available.

Profile 1 is used for nominal Aqua DPREP operation. It is the profile that is run on a routine basis.

Profile 2 (no previous or next data sets) initializes DPREP's processing of an ephemeris data stream. When Profile 2 has been run on a data segment, Profile 1 assumes processing responsibility.

Technical Notes:

1. The DPREP for Aqua ephemeris will be run separately for predictive and definitive orbit data (e.g. at GSFC and/or Langley)
2. The predictive data output will be used by the Aqua Attitude PGE to produce attitude for AIRS. This is being done because predictive is available sooner for AIRS need for an early update for their processing requirements.

3. The attitude DPREP will be run later when definitive orbit data is available to produce definitive attitude for Aqua.
4. The latest Toolkit requires the tag: EOSPMGIIS to be in the ephemeris and attitude files. This replaces EOSPM1.

Updating the Orbit Model

Introduction to Updating the Orbit Model

To determine realtime the latest Orbit Start times, Orbit Period, Path Number and Orbit Number, PDPS takes in specific information about the orbit of the satellite during initial SSI&T. This information then becomes the basis for predictions of future orbit start times and numbers. Because this value is accurate within a fraction of a second of time, the satellite may "drift" or a correction to orbit, known as a "burn" may have been applied. Therefore, the satellite Orbit Start Time can get out of sync either +/- with reality. The consequences are an elapse in time that will affect the Production Request Editor's ability to find a granule that should match with a DPR, or an incorrect Orbit Time could be passed to the PGE. The update of Orbit parameters will be done weekly at a specific time with scripts specifically written to extract the new Orbit Parameters from the most recent DPREP output file. These parameters intern will be inserted manually to the ORBIT.ODL file and then the re-registration of the Orbit.ODL file into the PDPS by SSI&T personnel. The M&O support Help Desk Team is responsible for knowing when changes to Orbit location have taken place from the Flight Dynamics Systems (FDS). A KnowledgeBase with backup procedures will be maintained by M&O for contingencies concerning Orbit Model updates. DPREP processing will be the most likely place to experience a failure due to Orbit time sync error encounters. The restoration of Orbit parameters with new values from FDS will most likely be necessary. The following procedures are provided to bring about an updated Orbit Model within ECS.

Procedures to Update the Orbit Model

Upon receipt of updated orbit parameters: ORBIT_ NUMBER,

ORBIT_PERIOD,

ORBIT Path Number and

ORBIT_START Time.

Proceed with the following steps.

---

1 Telnet or Rlogin to location (ais) system where ODL files are stored. ie; **"/usr/ecs/OPS/CUSTOM/data/DPS/ODL"**

2 Select the ORBIT.odl that is currently being used.

3 Using **vi, update the following files with the new parameter values received:**

- **ORBIT_AM1.odl** and/or **ORBIT_EOSAM1.odl** if they both are in use.

4 Have someone double check your entries for accuracy before preceding to the SSIT Manager for registering the new ODL file in the PDPS system.

5 For **Test Data** only; determine the Instrument PGE ODL that will be updated. MISR, MODIS etc.

- Using **vi, update the corresponding PATHMAP_Instrument_.odl** file with the new parameter values received.
- Ensure that the ABSOLUTE_PATH and MAPPED_PATH parameters agree with those in the new ORBIT_XXXX.odl.

6 SSI&T personnel will execute an Orbit Model Update by running a Dummy PGE established for this purpose at each of the DAAC's. Note: A dummy PGE is ran since a normal PGE cannot be re-registered if any DPRs exist in the system.

7 Notify DAAC Operations Supervisor that the Orbit Model has been updated. He will make a log entry of such action taken and may request the old computed values and the new replacement values be provided. The Supervisor will ensure that the orbital change is within several seconds, the expected change and not minutes!

PGE Chaining

1 Create PRs (so that DPRs) for the PGEs to be chained.

This can be done by using PR Editor. Follow the same procedure as creating independent PR.

A few points need to be noticed. Let's say among the chained PGEs, the output of PGE A will be the input of PGE B.1) In ESDT odl for this shared granule, "DYNAMIC_FLAG" has to be set to "I", i.e., dynamic internal. 2) First create PR for PGE A, then for PGE B. Otherwise PGE B PR may not be able to be generated.

2 Create the plan for a bunch of PRs which are chained.

In Work Bench GUI, 1) pull down "file" menu and select "new" to create the new plan; 2) highlight all PRs that are chain by clicking on their names on "unscheduled" area of Production Request area; 3) click schedule button to schedule these PRs.

3 Activate the plan.

In the Workbench GUI, click "activate" button, a GUI will pop up to ask for saving the plan. Answer "yes". Then another GUI will pop up to confirm whether to really activate the plan. Answer "yes" and the lowest level of DPR(s) in the chain will kick off.

In the pdps database, PlDataProcessingRequest table, when the PRs are successfully generated, the "completionState" for all DPRs in the chain are "NULL". When the plan is successfully activated, the "completionState" for lowest level of DPR(s) is changed from "NULL" to "STARTED". The high level of DPR(s) in the chain is changed from "NULL" to "CQ_HOLD". Eventually, the low level of DPR(s) finish so that the input for high level of DPR(s) become available, and the high level DPR(s) kick off and "completionState" change from "CQ_HOLD" to "STARTED".

PGE Registration and Test Data Preparation

PGE Registration

The integration of science software with ECS requires that information about the Product Generation Executives (PGEs) be made known to the PDPS in its database. In addition, the PGEs themselves and the test files that they use (both input and output) need to be placed on the Data Server. These steps must be accomplished before the science software can be run and tested within the ECS.

The following procedures describe how to register a new PGE with ECS. This involves updating the PDPS database with information needed to plan, schedule, and run the PGE.

1. The first step in the PGE registration process is to determine which ESDTs are needed for the PGE. You must Verify that an ESDT metadata ODL file exists for each ESDT or generate an ODL file.

2. The next step in the process is to create a PGE metadata ODL file using the delivered PCF.

3. Finally, additional operational information (resource requirements and runtime statistics) must be input into the PDPS database. This is the last step in the PGE registration process. The order in which these procedures are done is important and should be done as indicated.

PGE ODL Preparation

This section describes how to prepare PGE ODL files. It is assumed that the SSIT Manager is running .

1 From the SSIT Manager, click on the **Tools** menu, then choose **PDPS Database** and then **PCF ODL Template**.

An xterm with title "SSIT: Science Metadata ODL Template Creation" will be displayed.

2 At the program prompt **Configuration Filename (enter for default: ../../cfg/EcDpAtCreatODLTemplate.CFG)?**

Press **Enter** for the default configuration file

3 At the program prompt **ECS mode of operations?,** type ***mode***, press **Enter** or just press **Enter** if the default shown is correct.

The *mode* refers to the database used and will typically be OPS or TS1.

4 At the program prompt **Process Control file name (PCF to generate template from)?**, type ***PCFpathname*/*PCFfilename*** and then press the **Enter** key.

The *PCFpathname* is the full path name to the location of the PCF. If not specified, the directory from which the SSIT Manager was run will be assumed.

The *PCFfilename* is the file name of the PCF.

5 At the program prompt **PGE name (max 10 characters)?**, type ***PGEname*** and then press the **Enter** key.

The *PGEname* is the name of the PGE that will be registered.

6 At the program prompt **PGE version (max 10 characters)?**, type ***PGEversion***, press **Enter** or just press **Enter** if the default shown is correct.

The *PGEversion* is the version of the PGE that will be registered.

7 At the prompt **PGE Profile ID (0 for Null, max 999)?,** type 1 or any valid profile ID.

After a brief time, the message "Successfully created ODL template file" should be displayed if the task was successful.

The program will output a file with the filename PGE_*PGEname#PGEversion#ProfileID*.tpl.

For example, if the PGE name was PGE35, and the version and profile ID were both 1 this output file will be named PGE_PGE35#001#01.tpl.

GUI will ask for a full path to place template file.

8 At the program prompt **Directory for output template file (including full path)?**, type **ODLtplPathname,** press **Return.**

- The **ODLtplPathname** is the full path to the directory for the output template file, e.g., /usr/ecs/TS1/CUSTOM/data/DPS.

9 At the program prompt **Hit Enter to run again, 'q <Enter>' to quit:**, press **Enter** to repeat process with another PCF or type **q** and press **Enter** to quit.

The xterm will disappear.

10 At a UNIX prompt on an AIT Sun, type **cd *ODLtplPathname*** and then press the **Enter** key.

The *SSITrunPathname* is the full path to the directory from which the SSIT Manager was run, for example **/usr/ecs/TS1/CUSTOM/bin/DPS**. This will be the directory where the file PGE_*PGEname#PGEversion#ProfileID*.tpl will reside.

11 At a UNIX prompt on the AIT Sun, type **cp PGE_*PGEname#PGEversion#ProfileID*.tpl PGE_*PGEname#PGEversion#ProfileID*.odl** and then press the **Enter** key.

The PGE_*PGEname#PGEversion#ProfileID*.tpl is the file name of the ODL template file created in step 7.

The PGE_*PGEname#PGEversion#ProfileID*.odl is the file name of a copy which can be safely edited. This file name convention must be used.

12 At a UNIX prompt on the AIT Sun, type

mv PGE_*PGEname#PGEversion#ProfileID*.odl
/usr/ecs/<mode>/CUSTOM/data/DPS/ODL

This will place the ODL file in the directory where the executable that populates the PDPS database will read from. PGE_*PGEname#PGEversion#ProfileID*.odl is the file name of the copy created in step 11.

13 At a UNIX prompt on the AIT Sun, change the directory to the one in step above and type **vi PGE_*PGEname#PGEversion#ProfileID*.odl** and then press the Enter key.

The PGE_*PGEname#PGEversion#ProfileID*.odl is the file name of the copy created in step 11.

Any text editor may be used such as *emacs*. For example, emacs PGE_PGE35#001#01.odl and then press the **Enter** key.

14 In the file, add required metadata to the ODL template.

For an explanation of what metadata is required, see file /usr/ecs/<mode>/CUSTOM/data/DPS/PGE_ODL.template.

Note that the ShortNames typed into this file must each have a corresponding PDPS ESDT metadata ODL file .

All objects corresponding to output ESDTs will automatically have the SCIENCE_GROUP and YIELD set during the generation of PGE ODL.

All objects corresponding to output ESDTs will have an attribute “ASSOCIATED_MCF_ID. Place here the Logical Unit Number (LUN) listed in the PCF for the associated MCF listing.

All objects corresponding to static input ESDTs must have the SCIENCE_GROUP set. Objects corresponding to *dynamic* input ESDTs should NOT have the SCIENCE_GROUP set.

See Appendix E for an example of PCF and corresponding PGE ODL files.

15 Save the changes made to the ODL template file and exit the editor.

The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is :wq and then press the **Enter** key.

For other editors, refer to that editor’s documentation.

ESDT ODL Preparation

Assumption:

The PGE ODL file has been created and edited for the required PGE.

Follow the steps below to prepare ESDT ODL files for each ESDT required by the PGE.

1 Determine ShortName for required ESDTs corresponding to a Logical Unit Number (LUN) in the PGE ODL file.

2 At a UNIX prompt on an AIT Sun, type **ls /usr/ecs/<mode>/CUSTOM/data/DPS/ODL/ESDT_*ShortName#Version*.odl** and then press the Enter key.

The ESDT_*ShortName#Version*.odl is the file name of the ESDT ODL file you are looking for where *ShortName* is the ESDT's ShortName and *Version* is the ESDT version. If a file for the desired ESDT is listed, then it has already been prepared and this procedure can be exited now.

For example, if the desired ESDT has the ShortName MOD03 and version 001, type **ls /usr/ecs/TS1/S/CUSTOM/data/DPS/ODL/ESDT_MOD03#001.odl**, press **Enter**.

If the desired file is *not* listed, continue on to step 3.

3 At a UNIX prompt on the AIT Sun, type **cd *WorkingPathname*** and then press the **Enter** key.

The ***WorkingPathname*** is the full path name to a working directory for which the user has write permissions.

For example, **cd /home/jdoe/working/** and then press the **Enter** key.

4 At a UNIX prompt on the AIT Sun, type **cp /usr/ecs/<mode>/CUSTOM/data/DPS/ESDT_ODL.template ESDT_*ShortName#Version*.odl** and then press the **Enter** key.

For **<mode>** enter the mode you are working in, for example **OPS** or **TS1**.

The ESDT_***ShortName#Version*.odl** is the file name of the ESDT ODL file to be created.

This command copies a template ESDT ODL file to the ESDT ODL file to be created. The template is well commented.

For example, type cp /usr/ecs/<mode>/CUSTOM/data/DPS/ESDT_ODL.template ESDT_MOD03#001.odl and then press the Enter key.

The **ESDT_*ShortName#Version*.odl** file naming convention ***must*** be observed.

5 At a UNIX prompt on the AIT Sun, type **vi ESDT_*ShortName#Version*.odl** and then press the **Enter** key.

The **ESDT_*ShortName#Version*.odl** represents the file name of the ESDT ODL template file created in step 4.

Any text editor may be used such as *emacs*. For example, **emacs ESDT_MOD03#001.odl** and then press the **Enter** key.

6 In the file, add required metadata to the ODL template.

Use the internal documentation contained in the ODL file (from the original template) to aid in populating with metadata.

Note that the ShortName specified within the file must match the ShortName of the file name itself.

In addition, the ShortNames used in the PDPS PGE metadata ODL file must match the ShortNames in these files.

7 Save the changes made to the ESDT metadata ODL file and exit the editor.

The specifics depend upon which editor is being used. If using ***vi,*** the command sequence to enter is :**wq** and then press the **Enter** key.

For other editors, refer to that editor's documentation.

8 Next type **mv ESDT_*ShortName#Version.odl* /usr/ecs/<mode>/CUSTOM/data/DPS/ODL.**

This will place the just created ESDT ODL file in the directory where PDPS will read it from.

9 Repeat steps 1 through 8 for each ESDT required by a particular PGE. When all ESDT metadata ODL files have been completed, continue on to next section.

Updating the PDPS Database with Science Metadata

1 The directory used for containing the PDPS ESDT metadata ODL files can be accessed by the following commands:

On workstation **x0ais##,** at the UNIX prompt in a terminal window, type as in step 1 below your **user id** and **password.**

NOTE: The x in the workstation name will be a letter designating your site:

g = GSFC, m = SMC, l = LaRC, e = EDC, n = NSIDC, o = ORNL, a = ASF, j = JPL; the ## will be an identifying two-digit number (e.g.,g0ais01 indicates a Data Processing Subsystem (DPS) workstation at GSFC).

Prior to the rlogin, enter **setenv DISPLAY <local_workstation IP address>:0.0.** The **<ipaddress>** is the ip address of **x0ais##,** and xterm is required when entering this command on a Sun terminal.

2 telnet to (AITTL/DPS) p0ais01 or a machine that matches the SSIT Manager host.

The directory used for containing the PDPS ESDT metadata ODL files. is **/usr/ecs/<mode>/CUSTOM/data/DPS/ODL**

To update the PDPS Database with PGE/ESDT Metadata, execute the procedure steps that follow:

---

1 From the SSIT Manager, click on the **Tools** menu, then choose **PDPS Database** and then **SSIT Science Metadata Update**.

An xterm with title "SSIT: Science Metadata Database Update" will be displayed.

2 At the program prompt **Configuration Filename (enter for default: ../../cfg/EcDpAtRegisterPGE.CFG)?**

Press Enter.

3 At the program prompt **ECS mode of operation?**, type ***mode***, press **Enter** or just press **Enter** if the default shown is correct.

The ***mode*** refers to the database used and will typically be OPS or TS1.

4 At the program prompt **PGE name (max 10 characters)?**, type ***PGEname*** and then press the **Enter** key.

The ***PGEname*** is the name of the PGE that will be registered. This name must match the PGE name specified.

5 At the program prompt **PGE version (max 10 characters1)?**, type ***PGEversion***, press **Enter** or just press **Enter** if the default shown is correct.

The ***PGEversion*** is the version of the PGE that will be registered. This version must match the PGE version specified.

6 At the program prompt **PGE Profile ID (0-999, 0 means null)?** Type in a valid profile ID and press **Enter,** or if already listed just press **Enter.**

The PDPS database will then be updated with the information contained in the file **PGE_*PGEname#PGEversion#ProfileID*.odl**

7 At the program prompt **Hit Enter to run again, q <Enter> to quit:**, press **Enter** to update the PDPS database with another PGE ODL metadata file or type **q** and press **Enter** to quit.

If you make a mistake entering any values, press **Enter** here; your previous entries are restored as defaults and you won't have to retype them.

NOTE: If you make mistakes while editing the PGE and ESDT ODL files, you can run the ODL checker (Tools → PDPS Database → Check ODL) via the SSIT manager to locate any errors.

ODL files must have been created to define the PGE to PDPS. Examples of the ODL files are under the data directory: PGE_ODL.template, ESDT_ODL.template, ORBIT_ODL.template, TILE_ODL.template and PATHMAP_ODL. A tool can be run to generate a template ODL file for the PGE from the SSIT Manager via Tools->PDPS Database->PCF Odl Template script. This then has to be populated with all information that can not be garnered from the PCF. The

CheckOdl tool from the SSIT Manager via Tools->PDPS Database->Check ODL can be used to flag any errors in ODL before trying to put it in the database.

Example of ESDT.odl files being established in ECS

```
home/emcleod/MODIS/STORE/PGE07
p0ais01{emcleod}10: ls
ESDT_MD10L2#0.odl     MOD_PR10          pge_cfg
ESDT_MD35L2#0.odl     MOD_PR10.mk       scf_cfg
ESDT_MOD02H#0.odl     PGE07.mk          script
ESDT_MOD03#0.odl      doc
p0ais01{emcleod}11: cp ESDT_MD10L2#0.odl ESDT_MD35L2#0.odl
ESDT_MOD02H#0.odl ESDT_MOD03#0.odl /usr/ecs/OPS/CUSTOM/data/DPS/ODL/
```

AlternativeTool for SSIT Metadata Update:

Source the buildrc file for the mode in which you are working (*source .buildrc)*. **/usr/ecs/<MODE>/CUSTOM/utilities,** Note that this only has to be done once per login. Then **(*cd* /usr/ecs/<MODE>/CUSTOM/bin/DPS)**

(The tool can also be executed by being in the **/usr/ecs/<MODE>/CUSTOM/bin/DPS** and executing **EcDpAtDefinePGE..)**

Shell script prompts user for information.

1 Enter in the location of the configuration file **(../../cfg/EcDpAtRegisterPGE.CFG).**

2 Filename Enter the MODE of operation (<MODE>).

3 Enter name of PGE (it must match what is in the PGE ODL file).

4 Enter the version of the PGE (it must match what is in the PGE ODL file).

5 Enter the Profile ID (it must match what is in the PGE ODL file). Note that the ODL file for the PGE must have the of: PGE_<PGE NAME>#<PGE VERSION>#<PROFILE ID>.

Each ODL file is displayed as it is processed. A good status message should be displayed as a result. Information about the PGE (inputs and outputs, Production Rules, etc) should be entered in the Database.

Example of a successful PDPS Science Metadata Update:

PDPS/SSIT SCIENCE Metadata Database Update **
Configuration filename? (enter for default: ../../cfg/EcDpAtRegisterPGE.CFG)
ECS Mode of operations? (enter for default: OPS)
OPS
PGE name (max 10 characters)?
PGE07
PGE version (max 10 characters)?
0

```
PGE Profile ID (0-999, 0 means null)? (enter for default: 1)
1
Warning: Could not open message catalog "oodce.cat"
EcDpAtRegisterPGE: Process Framework: ConfigFile
../../cfg/EcDpAtRegisterPGE.CFG  ecs_mode OPS
Processing for PGE name = 'PGE07'  PGE version = '0'  PGE profile id = '1' ...
Do you wish to overwrite the previous PGE PGE07( (y)es or (n)o):
y
FILES PROCESSED
:  PGE SCIENCE ODL file =/usr/ecs//OPS/CUSTOM/data/DPS/ODL/PGE_PGE07#0#001.odl
   ESDT SCIENCE ODL file = /usr/ecs//OPS/CUSTOM/data/DPS/ODL/ESDT_MOD02H#001.odl
   ESDT SCIENCE ODL file = /usr/ecs//OPS/CUSTOM/data/DPS/ODL/ESDT_MD35L2#001.odl
   ESDT SCIENCE ODL file = /usr/ecs//OPS/CUSTOM/data/DPS/ODL/ESDT_MOD03#001.odl
   ESDT SCIENCE ODL file = /usr/ecs//OPS/CUSTOM/data/DPS/ODL/ESDT_MD10L2#001.odl
********** Update of PDPS/SSIT database with  PDPS SCIENCE metadata SUCCESSFUL **********
Hit Enter to run again, 'q <Enter>' to quit:
```

SSIT Operational Metadata Update GUI

The SSIT version of the PDPS database is initialized and updated with SSIT Operational Metadata so that the Planning and Processing Subsystem can schedule and run PGEs. Here, PDPS Operational Metadata refers to PGE information which is supplied to the DAAC/SSIT Operator and may change frequently.

The operator enters this data directly into the SSIT Operational Metadata Update GUI (Figure 41). The program then writes the data directly to the SSIT version of the PDPS database. The SSIT Operational Metadata Update GUI is used to view or update the following operational parameters for a particular PGE:

Performance parameters for the PGEs.

Resource parameters for the PGEs.

PGE user-defined static parameter.

View the PGE science metadata file.

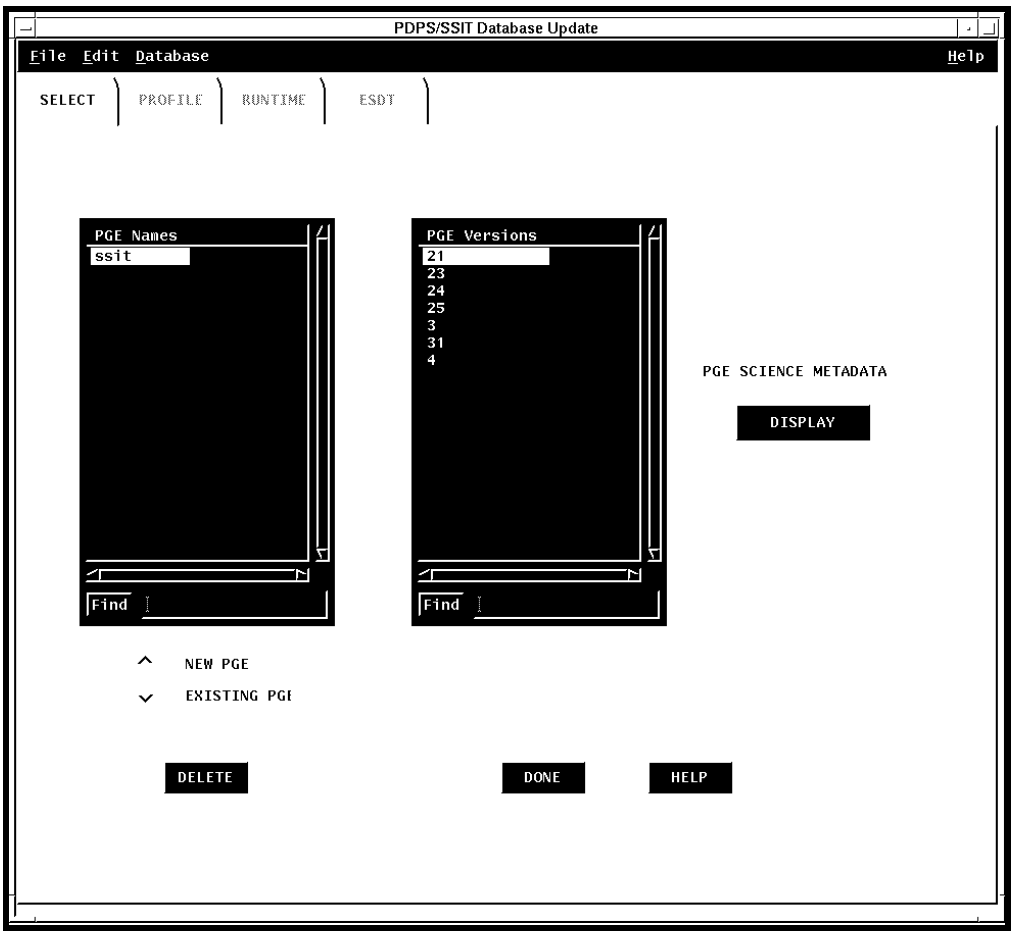


Figure 41. SSIT Database Operational Metadata Update GUI – SELECT view

Operational Metadata

The SSIT version of the PDPS database is initialized and updated with SSIT Operational Metadata so that the Planning and Processing Subsystem can schedule and run PGEs. Here, PDPS Operational Metadata refers to PGE information which is supplied to the DAAC/SSIT Operator and may change frequently.

The operator enters this data directly into the SSIT Operational Metadata Update GUI. The program then writes the data directly to the SSIT version of the PDPS database.

Before running the SSIT Operational Metadata Update from the SSIT Manager, you must first update the PDPS with SSIT Science Metadata. In addition, to get initial PGE Performance data which will be entered into the GUI, you need to run the profiling utility, EcDpPrRusage on the PGE or have the information on profiling provided.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The required UNIX environment variables have been set.
2. The Science metadata has been updated to the PDPS database for this PGE.

To update the SSIT version of the PDPS database with operational metadata, execute the steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **PDPS Database** and then **SSIT Opnl Metadata Update**.

The PDPS/SSIT Database Update GUI **"dpAtOpDbGui"** will be displayed.

2 Click on the radio button labeled **NEW PGE** in the lower left quadrant.

The PGE that you are working on should appear in the subwindow **labeled PGE Names** along with its version number in the subwindow labeled **PGE Versions**.

3 In the subwindow labeled **PGE Names**, click on a PGE name. Then in the subwindow labeled **PGE Versions**, click on the PGE version for that PGE. Then click on the button labeled **EDIT**.

The PGE name and version will be highlighted when you click on them.

The page tabs **PROFILE, RUNTIME,** and **ESDT** will change from gray (indicating disabled) to black (indicating enabled).

To see the contents of PGE Metadata, click on the button labeled **DISPLAY** and then click on the button labeled **DONE**.

If the PGE name and/or version does not appear in the lists, it means that updating of PDPS database with PGE metadata was not successful.

4 Click on the **PROFILE** page tab.

The Profile page will be displayed.

5 In the fields under the label **Performance Statistics**, enter the information specified.

In the field labeled **Wall clock time,** enter the amount of wall clock time it takes for one execution of the PGE, in seconds. The **tab PROFILE** will change from black (indicating enabled) to red (indicating database needs to be updated by APPLY button).

In the field labeled **CPU time (user),** enter the so-called *user* time of the PGE, in seconds. This value should come from profiling the PGE .

In the field labeled **Max memory used**, enter the maximum amount of memory used by the PGE, in megabytes (MB). This value should come from profiling the PGE .

In the fields labeled **Block input ops and Block output ops**, enter the integer number of block inputs and block outputs, respectively. These values should come from profiling the PGE .

In the field labeled **Swaps**, enter the integer number of page swaps from the PGE. This value should come from profiling the PGE .

In the field labeled **Page faults**, enter the integer number of page faults from the PGE. This value should come from profiling the PGE .

6 In the fields under the label **Resource Requirements**, enter the information specified.

In the field labeled **Runtime directory Disk Space**, **enter** the maximum amount of disk used by the PGE during execution, in megabytes (MB). Space should be allowed for the executable(s), input files, output files, ancillary files, static files, MCFs, and the PCF. (This number should also be in the PGE metadata ODL file; yes, there is duplication here.)

Click on the radio button labeled **Processing Strings/Computer** (if not already clicked on).

A list of processing strings should appear in the scrollable window to the left of the two radio buttons **Proc. String and Computer Name**. Nominally, only one item should be listed and should be highlighted.

In the field labeled **Number of CPUs**, the number 1 should appear.

7 In the field labeled **Local filename of top level shell**, type in the appropriate executable file name.

8 In the field labeled **SGI Application binary Interface (ABI)**, click the button to select **Old32** or **64**.

9 Once the fields on the **PROFILE** page have been completed, click on the **APPLY** button.

This will update the PDPS database with the information just entered. The tab **PROFILE** will change from red (indicating database needs to be updated) to black (indicating enabled).

An information box will be displayed; click on **Ok.**

To start over, click on the **RESET** button. This will clear all fields.

10 Click on the **File** menu and select **Exit**.

This will end the session with PDPS/SSIT Database Update and the GUI will disappear.

Test Data Preparation and Insertion of Data Granules

This section describes how to prepare test data for use by registered PGEs. When PGEs are first delivered to the DAAC and registered within the PDPS, they will typically be run in isolation. That is, they will be run without any PGE dependencies. For this testing to be possible, test input data granules required by the PGE need to be pre-Inserted to the Data Server.

Data granules can be *dynamic* or *static*. Dynamic data granules are those whose temporal locality differs for each instance of the granule. Examples of dynamic granules are Level 0, Level 1, and Level 2 data sets. Static data granules are those whose temporal locality is static over long periods of time. Examples of static granules are calibration files which may only change with a new version of a PGE. For any granule to be Inserted to the Data Server, a Target MCF is needed (also known as an ASCII metadata ODL file or a **.**met file).

In the actual production environment, a Target MCF is produced by the PGE during execution. Thus, the data granule can be inserted automatically. In isolation testing of a PGE, however, the inputs needed by this PGE will not have been inserted by a previous PGE in the chain. This Insertion must be done manually. The next two sections describe how to use the Source MCF for a dynamic data granule to create a Target MCF. and then describes how to do the insert. In this way, a dynamic data granule can be inserted to the Data Server as if a PGE had produced it.

Generating a Metadata Configuration File (Source MCF)

Detailed procedures for tasks performed by the SSI&T Operator are provided in the sections that follow.

Assumptions:

1. The SSIT Manager is running.
2. ESDT's are installed onto the **Science Data Server.**

The MCF's for the output files are to be generated only for the purpose of comparison with the delivered MCF's. This way an SSIT operator can notice any metadata mismatch between the two files and notify the instrument team and ECS ESDT group. In actual run of the PGE, the system will create the MCF's for the output files.

To Generate the Metadata Configuration File (Source MCF) for the input and output ESDT's, execute the steps that follow.

1 From the SSIT Manager, click on the **Tools** menu, then choose **Data Server** and then **Get MCF.**

An xterm in which EcDpAtGetMCFis running will be displayed as **"SSIT: Acquire MCF".**

Alternatively, the same tool can be invoked by typing at a UNIX prompt on an AIT Sun **EcDpAtGetMCF.sh** and then press the **Enter** key.

2 At the program prompt **Configuration Filename (default *defaultConfigFile*)?**

Type in ../../cfg/ *defaultConfigFile* and press Enter.

The ***defaultConfigFile*** will be replaced by the full path name and file name of the default configuration file. The file name will be EcDpAtGetMCF.CFG where *daac* will be replaced by one of {GSFC, EDC, LARC, NSIDC}.

3 At the program prompt **ECS mode of operation (enter for default: def*aultMode*)?**, type ***mode***, press **Enter** or just press **Enter** if the default shown is correct.

The ***mode*** refers to the database used and will typically be **TS1**.

4 At the program prompt **ESDT Short Name?**, type ***ESDT ShortName*** and then press the **Enter** key.

The ***ESDTShortName*** is the name of the ESDT that the EcDpAtGetMCF tool will use to generate the MCF.

5 At the program prompt **ESDT Version?**, type ***ESDTversion***, press **Enter** or just press **Enter** if the default shown is correct.

The ***ESDTversion*** is the version of the **ESDT.**

6 At the program prompt **Directory to receive MCF (must be full path)?**, type ***MCFpathname*** and then press the **Enter** key.

The ***MCFpathname*** is the full path name to the location where the source MCF will be placed. For example, **/home/jdoe/ssit**.

7 To the final prompt **Hit Enter to run again, 'q <Enter> to quit:**, press **Enter** to generate another Source MCF or type **q** and press **Enter** to quit.

If you make a mistake entering any values, press **Enter** here; your previous entries are restored as defaults and you won't have to retype them.

Example of a successful installation of a Source MCF:

Configuration filename? (enter for default: ../../cfg/EcDpAtGetMCF.CFG)
ECS Mode of operations?
OPS
ESDT Short Name?
MOD03
ESDT Version?
0
Directory to receive MCF? (must be full path)
/home/emcleod/MCF/
Warning: Could not open message catalog "oodce.cat"
EcDpAtGetMCF: Process Framework: ConfigFile ../../cfg/EcDpAtGetMCF.CFG
ecs_mode
OPS

incomplete group entries in the configfile,using default G1
Request for MCF successful for:
ESDT name = 'MOD03'
ESDT version = '0'
directory = '/home/emcleod/MCF/'
Hit Enter to run again, 'q <Enter>' to quit:

Creating a Target MCF (.met) for a Dynamic/Static Granule

A Target MCF file for a corresponding data granule can be created based on the information provided in the Source MCF file and the involved science software package (PGE).

In standalone or isolation testing of a PGE, the inputs needed by it will not have been Inserted by a previous PGE in the chain. This Insertion must be done manually. A Target MCF file for a corresponding data granule is required to run a standalone PGE. This way a dynamic data granule can be Inserted to the Science Data Server as if a PGE had produced it.

The following steps can be used to obtain .met files for standalone PGE runs.

1 For all dynamic data granules, try to locate a .met file from an output of a previous PGE. Usually the input to a PGE is the output of some previous PGE. Hence, one can use the relevant documents from the instrument team to obtain such information on the required PGE. Once a .met file is available, all you need to do is edit the timestamp for your run of the PGE.

2 Use HDF_EOS-view from the SSIT manager to take a look at the header of the HDF data granules. The header contains information on the .met file.

Creating a Metadata File for a Static Granule

1 At the UNIX prompt on the AIT Sun, type cd WorkingPathname, then press the **Enter** key. Example: **cd /usr/ecs/{MODE}/CUSTOM/data/DPS/ODL/**

The ***WorkingPathname*** *is* the full path name of the working directory containing the template metadata ODL file.

2 At the UNIX prompt on the AIT Sun, type **cp *StaticODLmet.tpl filename*.met**, then press the **Enter** key.

The ***StaticODLmet.tpl*** is the file name of the template Target MCF.

The ***filename*.met** is the file name of the Target MCF for this static file. The file name extension must be **.met.**

This command will copy the template **Target MCF to *filename*.met.** For example, **type cp StaticODLmet.tpl CER11T.mcf.met**, then press the **Enter** key.

3 At a UNIX prompt on the AIT Sun, type **vi *filename*.met**, then press the **Enter** key.

This command invokes the ***vi* editor** and reads in the Target MCF created above.

4 Edit the Target MCF with the specific information for the static data granule to be Inserted. The following guidelines should be followed when editing on the template MCF:

The value for the ShortName object should be filled out with proper instrument name.

The value for the Version ID object should be filled out with the proper version number.

In the **INFORMATIONCONTENTCONTAINER** object enter the following:

The value for the **PARAMETERNAME** object of the class **"1"** should be filled out with the name of static data file.

The value for the **PARAMETERVALUE** object of the class **"2"** should be filled out based on the following guideline:

If the data granule is a coefficient file, a **"C"** followed by a numerical number **n (n=1,2,….)** will be used. Here n stands for the number of the coefficient file.

If the data granule is a MCF file, a **"M"** followed by a numerical number n (n=1,2,….) will be used. Here n stands for the number of the MCF file.

5 Save the changes made to the Target MCF (***filename*.met**) and exit the editor.

The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is :**wq,** then press the **Enter** key.

Inserting Static Data Granules into the Science Data Server

In order for static data files to be used both during the SSI&T and in production, this file must exist in the Data Server and be accessible by the local machine. A program called the Insert Static File can be used for inserting a static data granule into the Data Server.

The following Servers/Services must be up and operational:
Science Data Server, Storage Management.
The following must have occurred between those Servers/Services:
1. The **ESDT** of the static file must have been installed on the Science Data Server.
2. The **Target MCF (.met)** for this data granule has been created for the Insert,

What the user must do before trying SSIT functionality when using a Script to Insert the Static granule: If using the SSI&T Manager, go to the next section.

1. Source the buildrc file for the mode in which you are working **(*source .buildrc)*.** Note that this only has to be done once per login.
2. Then go to the executables directory (***cd* /usr/ecs/<MODE>/CUSTOM/utilities)**
3. Enter (**cd /usr/ecs/<MODE>/CUSTOM/bin/DPS** and execute **EcDpAtInsertStatic**

The Shell script will prompt user for information at each step of the process. See the next section, step 4 to pick up on this process.

What must be done using the SSIT Manager tools:

Assumptions:
The SSI&T Manager is running.

To start the Insert Static File Tool:

1 From the **SSIT Manager** choose **Tools** menu and then **Data Server** submenu. Choose ***Insert Static File*.**

A GUI with title "SSIT: PGE Static Input File Insertion" will be displayed.

2 Enter in the location of the DpAtInsertStaticFile configuration filename**? (enter for default: ../../cfg/EcDpAtInsertStaticFile.CFG).**

3 Enter the MODE of operation **(<*MODE*>).** At the program prompt **mode (default ops)?**, or press **Enter** to take default.

4 Enter the **short name** of the ESDT (for the static file). This value is in the pdps database under the PlDataTypeMaster table and must be in the PGE ODL file.

At the program prompt ESDT name? type ***ESDTShortName*,** then press the **Enter** key. For example type: **MOD02LUT.**

5 Enter the version of the ESDT for the static file. This value is also in the pdps database under the PlDataTypeMaster table and must be in the PGE ODL file.

At the program prompt PGE version (default 1)?, type ***PGEVersion***, then press the **Enter** key.

The ***PGEVersion*** must match exactly the PGE version entered into the PDPS for this PGE.

6 Enter the science group for this static (this will be from the ODL created during Populating the PGE information in the Database).

At the program prompt Science group for Static file**(one of{C,L,D,O}** followed by a 3 digit number)?, type ***ScienceGroupID*,** then press the **Enter** key.

The ***ScienceGroupID*** is an identifier used to define the file type as a coefficient file, a lookup table file, or a MCF. It distinguishes static granules of different types which share the same ESDT. For instance, for a coefficient file, use **C*n***, where number ***n*** could be 0, 1, 2…; this number ***n*** needs to be matched with the number ***n*** in the PGE_PGEName#Version.odl file. For an MCF. For example, type C001 and then press the **Enter** key.

The **Science Group ID** must match what was edited into the PGE metadata ODL file for that PCF entry.

7 At the program prompt **Is there more than one data file for this Static (Y = Yes, N = No)? (enter for default: N).** If there is only one data file, press **Enter** and go to next step. If there are more than one data files, type **Y,** press **Enter** and go to step 10.

8 At the program prompt **Single Static Filename to Insert (including FULL path)?**, type ***pathname*/*GranuleFileName***, press **Enter**

The ***pathname*/*GranuleFileName*** is the full path name and file name of the static data granule to be inserted. For example, type **/home/MODIS/PGE10/MOD_PR28/coeff/emissivity.dat** and then press the **Enter** key.

9 At the program prompt **Associated ASCII Metadata Filename to Insert (including FULL path).** Type ***pathname*/*GranuleFileName.met*)?** and then press the **Enter** key.

The ***pathname/GranuleFileName.met*** is the full path name and file name of the **.met** file for the associated static data granule to be Inserted. For example, **type /home/MODIS/PGE10/MOD_PR28/MOD28LUT.met** press **Enter.**

10 At the program prompt **Directory where all data files and .met file exist (FULL path)?** Type ***pathname*** press **Enter**.

where ***pathname*** is the full path of the directory where all data files and **.met** file exist.

Note for a multifile granule, the data files and .met file should be placed in the same working directory.

11 At the program prompt **Name of MFG file (enter to end list)?** Type in the ***GranuleFileName,*** one at a time and press **Enter.** To end the list press **Enter.**

where ***GranuleFileName*** is the names of the multifile granules.

12 At the program prompt **Associated ASCII Metadata Filename to Insert?** Type ***GranuleFileName.met*** and then press the **Enter** key.

13 Where ***GranuleFileName.met*** is the name of one **.met file** that is used with all data granules in the event of a multifile granule.

14 The static data granule will be inserted to the Data Server. For reference, the Data Server Universal Reference (UR) will be printed on the screen.

15 At the program prompt **Hit Enter to run again, 'q <Enter>' to quit:** type **q** and press **Enter** to quit or just press **Enter** to insert additional dynamic granules.

If continuing, repeat steps 2 through 9.

Inserting Dynamic Data Granules to the Science Data Server

In order for dynamic data files to be used both during the SSI&T and in production, this file must exist in the Data Server and be accessible by the local machine. A program called the Insert Test Dynamic File can be used for inserting a dynamic data granule into the Data Server.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ESDT's have been installed on the Data Server.
2. The Target MCF for this data granule has been created for the Insert.

To Insert a dynamic granule to the Data Server, execute the following steps:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Data Server** and then **Insert Test Dynamic**.

An xterm with title “SSIT: PGE Test Dynamic Input File Insertion” will be displayed.

2 At the program prompt **Configuration filename? (enter for default: ../..cfg/EcDpAtInsertTestFile.CFG)** and then press the **Enter** key.

3 At the program prompt **ECS Mode of operations?**

Type in the **<mode>** you are working in. For example, **TS1** or **OPS**. Press **Enter**.

4 At the program prompt **ESDT short name for the file(s) to insert?** type ***ESDTShortName***, press **Enter**

The ***ESDTShortName*** is the ShortName of the ESDT descriptor file corresponding to this granule to be inserted. For example, type **MOD021KM** press **Enter**.

5 At the program prompt **ESDT Version for the file(s) to insert?** Type in the ESDT version and press **Enter.**

6 At the program prompt **Is there more than one data file to this Dynamic Granule (Y = Yes, N = No)? (enter for default: N)?** If there are no multifiles for this ESDT, press **Enter** and go to step 7. If there are more than one file for this granule go to step 9.

7 At the program prompt **Single Filename to Insert? (including FULL path)** type ***pathname/GranuleFilename***, press

The ***pathname/GranuleFileName*** is the full path name and file name of the data granule to be Inserted. For example, type **/home/MODIS/PGE10/MOD021KM.A1996217.0014.002.hdf** and then press the **Enter** key.

8 At the program prompt **Associated ASCII Metadata Filename to Insert (including FULL path)** , Type ***pathname/GranuleFileName.met*** and press **Enter**.

pathname is full name of the path and ***GranuleFileName.met*** is the name of the associated .met file. For example, **/home/MODIS/PGE10/MOD021KM.met**

The dynamic data granule will be inserted to the Data Server. For reference, the Data Server Universal Reference (UR) will be printed on the screen.

9 At the program prompt **Directory where all data files and .met file exist (FULL path)?** Type ***pathname*** press **Enter**.

where ***pathname*** is the full path of the directory where all data files and **.met file** exist.

Note for a multifile granule, the data files and .met file should be placed in the same working directory.

10 At the program prompt **Name of MFG file (enter to end list)?** Type in the ***GranuleFileName,*** one at a time and press **Enter.** To end the list press **Enter.**

Where ***GranuleFileName*** is the names of the multifile granules.

11 At the program prompt **Associated ASCII Metadata Filename to Insert?** Type ***GranuleFileName.met*** and then press the **Enter** key.

Where ***GranuleFileName.met*** is the name of one .met file that is used with all data granules in the event of a multifile granule.

The dynamic data granule will be Inserted to the Data Server. For reference, the Data Server Universal Reference (UR) will be printed on the screen.

12 At the program prompt **Hit Enter to run again, 'q <Enter>' to quit:** type **q** and press **Enter** to quit or just press **Enter** to insert additional dynamic granules.

If continuing, repeat steps 2 through 8.

Example of a successful insertion of a Dynamic Input Data Granule into the Data Servers:

```
PGE Test Dynamic Input File Insertion **
Configuration filename?  (enter for default:
../../cfg/EcDpAtInsertTestFile.CFG)
ECS Mode of operations?  (enter for default: OPS)
OPS
ESDT name
MOD02H
ESDT Version (enter for default: 1)
0
Staged Filename to Insert? (including FULL path)
/home/emcleod/MCF/MOD02HKM.A1997217.1730.002.hdf
Associated ASCII Metadata Filename to Insert? (including FULL path)
/home/emcleod/MCF/MOD02H.met
Warning: Could not open message catalog "oodce.cat"
EcDpAtInsertTestFile: Process Framework: ConfigFile
../../cfg/EcDpAtInsertTestFile.CFG  ecs_mode OPS
incomplete group entries in the configfile,using default G1
Trying to make a request to [MDC:DSSDSRV]
incomplete group entries in the configfile, using default
Trying to make a request to [MDC:DSSDSRV]
incomplete group entries in the configfile, using default
Insert to Data Server successful:
   ESDT Version = '0'
  staged file = '/home/emcleod/MCF/MOD02HKM.A1997217.1730.002.hdf'
  metadata file = '/home/emcleod/MCF/MOD02H.met'
Inserted at UR:
'UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[MDC:DSSDSRV]:16:SC:M
OD02H:1757'
Hit Enter to run again, 'q <Enter>' to quit:
```

Alternative way to Insert a Granule into the Science Data Server

The Science Data Server subsystem includes a utility that will allow users to manually insert a data granule into the ECS. The tool will prompt the user for key inputs for inserting he granule. The following procedures describe this process.

1 Ensure that the **Science Data Server** subsystem is currently executing on the appropriate ACMHW HWCI server machine. Tested on **p0acs03.**

2 Start the Science Data Server test utility by entering the following at the UNIX prompt on the SDSRV workstation:

setenv MODE TS1

cd /usr/ecs/<mode>/CUSTOM/bin/DSS/

source ../ ../utilities/EcCoEnvCsh

/usr/ecs/<mode>/CUSTOM/bin/DSS/dttest6ConfigFile
/usr/ecs/<mode>/CUSTOM/cfg/EcDsScienceDataServerClient.CFG

ecs_mode <mode>

3 The following selection menu is displayed:

1. **INSERT granule**
2. **INSERT granule with browse file**
3. **ACQUIRE granule with date**
4. **ACQUIRE granule with a UR**
5. **DELETE granule**
6. **Exit**

Please make selection=>

Choose option **1** to perform the search and acquire.

4 The program indicates that the search process will take place first by displaying the message "**Executing insert .....** ". The user is prompted to enter the datatype of the data that will be inserted. For example:

Enter data type=> AST_L1BT

5 The program will then prompt the user for the full path name of the data file. For example:

Enter datafile name (full path)=>/tmp/:SC:AST_L1BT:1391:1.EOSHDF

6 The program will then prompt the user for the full path name of the metadata file.

For example: Enter data metafile name(full path)=>**/tmp/AST_L1BT.MCF**

7 The program will then give status on the success of the insert. The following messages should appear on the successful insert:

Insert science data only…

Trying to make a request to {:DSSDSRV} Success.

8 The user should hit Enter at this prompt and the program will redisplay the first menu that was given in step 3. The user can then choose option 6 to exit.

Alternative way to Acquire a Granule from the Science Data Server

The Science Data Server subsystem includes a utility that will allow users to manually search and retrieve (acquire) a data granule from the ECS. The tool will prompt the user for key inputs for acquiring the granule. The following procedures describe this process.

1 Ensure that the **Science Data Server** subsystem is currently executing on the appropriate ACMHW HWCI server machine. Tested on **p0acs03.**

2 Start the Science Data Server test utility by entering the following at the UNIX prompt on the SDSRV workstation:

setenv MODE TS1

cd /usr/ecs/<mode>/CUSTOM/bin/DSS/

source ../ ../utilities/EcCoEnvCsh

/usr/ecs/<mode>/CUSTOM/bin/DSS/dttest6

ConfigFile /usr/ecs/<mode>/CUSTOM/cfg/EcDsScienceDataServerClient.CFG

ecs_mode <mode>

3 The following selection menu is displayed:

1. **INSERT granule**
2. **INSERT granule with browse file**
3. **ACQUIRE granule with date**
4. **ACQUIRE granule with a UR**
5. **DELETE granule**
6. **Exit**

Please make selection=>

Choose option **3** to perform the search and acquire.

4 The program indicates that the search process will take place first by displaying the message "**Executing search .....** ". The user is prompted to enter a hostname. Enter the hostname of the machine on which the Science Data Server process is executing:

Enter hostname=> **dss2**

5 The user is prompted to enter a data type. Enter the ESDT short name of the type of data that is to be acquired. An example:

Enter data type=> **AST_L1BT**

6 The user is prompted to enter a start and end date. These dates indicate a range over which the user would like to search the database for data of the given type. The start and end dates will narrow the search to those data granules that were collected within that time range. Times are given in the format **mm/dd/yy**. For example:

Enter starting date**(mm/dd/yy)=> 07/04/97**

Enter end date**(mm/dd/yy)=> 07/05/97**

7 After the start and end dates are entered the utility will make a request of the Science Data Server and the following message will be displayed:

Trying to make a request to [:DSSDSRV]

A table of data granules will be displayed to the user if any were found within the given time range. For example, if the user had requested to display all of the data granules for the **ESDT AST_L1BT** within a certain time range, the following table would be displayed:

UR Type Create Date Size

-- ------- ----------- ----

AST_L1BT SC:AST_L1BT:1390

AST_L1BT SC:AST_L1BT:1391

AST_L1BT SC:AST_L1BT:1322

AST_L1BT SC:AST_L1BT:1289

AST_L1BT SC:AST_L1BT:1299

The table displays the data type (AST_L1BT) and UR (i.e., SC:AST_L1BT:1390) for each granule found.

In addition to the table, a list of the granules with a corresponding numerical index will be displayed with a prompt to the user to enter the index of the granule that they wish to acquire. An example of the indexed list follows:

NOW ENTERING ACQUIRE

There is(are)5 in the collection

Index = 0 AST_L1BT SC:AST_L1BT:1390 Index = 1 AST_L1BT SC:AST_L1BT:1391 Index = 2 AST_L1BT SC:AST_L1BT:1322 Index = 3 AST_L1BT

SC:AST_L1BT:1289 Index = 4 AST_L1BT SC:AST_L1BT:1299 Please enter the index of the associated UR

8 At this time the user should enter the numerical index of the granule that they wish to acquire and hit enter.

9 The SDSRV utility will prompt the user for a media type, the type of media on which the data will be retrieved:

Valid Media Types:

FtpPull

FtpPush

8MM

4MM

CDROM

9TRK

Enter media type (case sensitive)=>

The user should enter one of the values of the valid media types. If the user entered **FtpPush,** the user will expect the data to be **ftp'd** to a given directory.

10 After the user has entered the Media type, the utility will prompt the user for the media format. The media format should be entered as **"FILEFORMAT**"

Enter mediaformat=> **FILEFORMAT**

11 After the user has entered the Media format, the utility will prompt the user for the user profile id. This entry can be any alphanumeric character.

Enter userProfID => **a**

12 After the user has entered the user profile id, the utility will prompt the user for a user id and associated password. The user id/password will be used for authorization to perform the ftp of the data file from the archive area to the user-specified directory. The password field will not be echoed to the screen. The following is only an example. The user should use a valid user id/password within the current environment.

Enter username=> **sdsrv**

Enter password=>

13 The next entry that the user must enter is the host id of the machine to which they want the data ftp'd. Enter a valid host name as follows:

Enter host=> **dss2**

14 After the host id, the user must enter the fully qualified destination directory to which the file will be ftp'd:

Enter destination=> **/tmp**

15 After the destination directory has been entered, the utility will give the following message:

Trying to make a request to [:DSSDSRV]

If the acquire operation is successful, the utility will give the following message:

Acquire successful.

Please <CR> to continue.

16 The user should hit Enter at this prompt and the program will redisplay the first menu that was given in step 3. The user can then choose option 5 to exit.

Placing the Science Software Executable (SSEP) onto Science Data Server

In order to be able to run a PGE within the ECS system, the EXE TAR file has to be inserted to the Science Data Server. This tar file consists of all files needed to run a PGE, except for input data files. This includes the executables, any scripts, and the SDP Toolkit message files.

Assembling a Science Software Executable Package

This section describes how to assemble a Science Software executables Package (SSEP) and create a corresponding Target MCF. A SSEP is a UNIX tar file which contains PGE executables and SDP Toolkit message files.

In order to Insert a PGEEXE tar file into the Science Data Server, a corresponding Target MCF (.met) must be generated before insertion. Such an ASCII metadata ODL file can be obtained by editing an existing template ODL file with the information of the specific PGE. The following procedures describe how to assemble a PGEEXE tar file and create an ASCII metadata ODL file.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. PGE executables and message files required by this PGE are available to make a SSEP.

To create an SSEP, execute the steps that follow:

---

1 At the UNIX prompt on an AIT Sun, type **mkdir *SSEPpathname*** and then press the **Enter** key.

The ***SSEPpathname*** is the full path name of a ***new*** directory which will contain all the files to be placed into the SSEP as well as the SSEP itself.

It is recommended that ***SSEPpathame*** be named with a convention that indicates the PGE for which a SSEP will be created. For example, type **mkdir PGE35.ssep** and then press the **Enter** key.

2 At the UNIX prompt on the AIT Sun, type **cd *SSEPpathname*** and then press the **Enter** key.

The ***SSEPpathname*** is the directory name of the new directory created in step 1.

3 At the UNIX prompt on the AIT Sun, type **cp *pathname/file1 pathname/file2 … pathname/filen* .**, press **Enter** (note the "dot" and then space at the end of the command).

The ***pathname/file1, pathname/file2,…pathname/filen*** represents a list of path names and file names (delimited by spaces) to copy into the current directory, ***SSEPpathame*** (the "dot" represents the current directory and must be last in the command).

For example, type **cp /data/MODIS/pge/PGE35.exe /data/MODIS/MOD_13453** ., press **Enter** (note the space and then "dot" at the end of the command).

The files copied into this directory should be the PGE executable, any shell scripts or other executables that are part of the PGE and SDP Toolkit message files.

Files can be individually copied into the ***SSEPpathame*** directory. For example, type **cp /data/MODIS/pge/PGE35.exe** ., press **Enter** (note the space and then "dot" at the end of the command). Repeat for each file needed in the SSEP for this PGE.

4 At the UNIX prompt on the AIT Sun, type **tar cvf *SSEPfilename*.tar *** and then press the Enter key.

The ***SSEPfilename*.tar** is the file name for the **SSEP tar file**. The file name extension .tar is recommended but not required.

The asterisk **(*)** is a file name wildcard that represents all files in the current directory.

This will place all files in the SSEP tar file.

Once created, the contents of the SSEP tar file can be viewed by typing **tar tvf *SSEPfilename*.tar** and then press the **Enter** key.

Do not apply compression (*e.g.* UNIX compress or gzip) to the tar file.

5 At the UNIX prompt on the AIT Sun, type **cp *filename.met.tpl filename.met*** and then press the **Enter** key**.**

The ***filename.met.tpl*** is the file name of the template Target MCF for this SSEP. If a template is not available, see **Appendix D** or use one used for another SSEP.

The ***filename.met*** is the file name of the Target MCF to be tailored for this **SSEP**.

6 At the UNIX prompt on the AIT Sun, type **vi *filename.met*** and then press the **Enter** key.

The ***filename.met*** is the Target MCF for this SSEP.

This command invokes the ***vi*** editor. Edit the ***filename*.met** with the specific information for the **SSEP** to be inserted.

The following guidelines should be followed when editing on the Target MCF **(*filename*.met):**

The value for the **VERSIONID** object should be filled out with the proper **PGE version**. For example**:** "**1"**.

In the INFORMATIONCONTENTCONTAINER object,

The value for the **PARAMETERNAME** object of the **class "1"** should be filled out with the PGE name. For example: "**BTS**".

The value for the **PARAMETERNAME** object of the **class "2"** should be filled out with the PGE Science Software Version. For example: **"1".**

The value for the **PARAMETERNAME** object of the **class "3"** should be filled out with the Platform Name. For example: **"IRIX".**

The value for the **PARAMETERNAME** object of the **class "4"** should be filled out with the Platform Version. For example: **"6.2".**

The value for the **PARAMETERNAME** object of the **class "5"** should be filled out with the date to perform the Insertion. For example: **"970319".**

The value for the **PARAMETERNAME** object of the **class "6"** should be filled out with the time to perform the Insertion. For example: **"14:45:00".**

7 Save the changes made to the SSEP's Target MCF (***filename*.met**) and exit the editor.

The specifics depend upon which editor is being used. If using ***vi,*** the command sequence to enter is **:wq** and then press the **Enter** key.

For other editors, refer to that editor's documentation

Inserting Science Software Executable Package onto Science Data Server

Science software, like any other data that are managed in the ECS, must be placed on the Science Data Server. A program called the Insert EXE TAR Tool can be used for Inserting a Science Software Executable Package into the Data Server.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The ESDT called PGEEXE has been installed on the Science Data Server.
2. A Target MCF (.met) for this PGEEXE tar file has been created for the Insert.
3. The PGEEXE tar file has been created .\
4. The following Servers/Services must be up an operational:

Science Data Server, Storage Management.

To Insert the SSEP to the Science Data Server, execute the steps that follow:

1 From the SSIT Manager, click on the **Tools** menu, then choose **Data Server** and then **Insert EXE TAR**.

An xterm with title "SSIT: PGE Executable Tar File Insertion" will be displayed.

2 At the program prompt **Configuration filename? (enter for default: ../../EcDpAtInsertExeTarFile.CFG)** and then press the **Enter** key.

3 At the program prompt **ECS mode of operations?**, Type <***mode***> press **Enter.**

<***mode***> can either be **OPS** or **TS1**.

4 At the program prompt **Name of PGE?**, type ***PGEName*** and then press the **Enter** key.

The ***PGEName*** is the name of the PGE for which this static granule is being Inserted. For example, type **PGE01** and then press the **Enter** key.

The ***PGEName*** must match exactly the PGE name entered into the PDPS for this PGE.

5 At the program prompt **Science software version of PGE?**, type ***SSWversion*** and then press the **Enter** key.

The ***SSWversion*** is the version of the science software which is being Inserted in this SSEP. Press **Enter** to accept the default or enter in a version and press **Enter.**

6 At the program prompt **Staged filename to insert (including Full path)?**, type ***pathname/SSEPFileName***, press **Enter**

The ***pathname/SSEPFileName*** is the full path name and file name of the SSEP tar file to be Inserted. For example, type **/data/MOD35/ssep/PGE35_1.tar** and then press the **Enter** key.

The SSEP tar file must not be compressed (*e.g.* with UNIX compress or gzip).

7 At the program prompt **Associated ASCII metadata filename to insert (including Full Path)? *pathname/SSEPFileName.met*)?** and then press the **Enter** key.

The default is the file name of the granule to insert with the .met file name extension. If the default is not correct, then the file name of this file must be entered.

8 At the program prompt **Top level shell filename within tar file?**, type ***ExecFileName*** and then press the **Enter** key.

The ***ExecFileName*** is the file name of the top level executable or script within the SSEP tar file. It should be the same as was entered into the PDPS/SSIT Database Update GUI.

The SSEP will be Inserted to the Science Data Server.

9 At the program prompt **Hit Enter to run again, 'q <Enter>' to quit:** type **q** and press **Enter** to quit or just press **Enter** to insert additional dynamic granules.

If continuing, repeat steps 3 through 8.

Example of a successful insertion of a SSEP EXE TAR:

PGE Executable Tar File Insertion Script

Configuration filename? (enter for default:../../cfg/EcDpAtInsertExeTarFile.CFG)

ECS Mode of operations? (enter for default: OPS)

OPS

Name of PGE? (enter for default: PGE07)

PGE07

Science software version of PGE? (enter for default: 0)

0

Staged filename to insert (including FULL path)? (enter for default:

/home/emcleod/SSEP/MODPGE07.tar)

Associated ASCII metadata filename to insert (including FULL path)? (enter for

default: /**home/emcleod/SSEP/MOD_PR10.tar.met**)

Top level shell filename within tar file? (enter for default:

/home/emcleod/SSEP/MOD_PR10.exe)

MOD_PR10.exe (note: this entry is done a second time) Note: If you get **core dump**, execute using "dbx command: type in: **dbx filename .exe**. This will help isolate error message that caused core dump.

Warning: Could not open message catalog "oodce.cat"

EcDpAtInsertExeTarFile: Process Framework: ConfigFile

../../cfg/EcDpAtInsertExeTarFile.CFG ecs_mode OPS

Performing INSERT.....

incomplete group entries in the configfile,using default G1

Trying to make a request to [MDC:DSSDSRV]

incomplete group entries in the configfile, using default

Trying to make a request to [MDC:DSSDSRV]

incomplete group entries in the configfile, using default

Insert to Data Server and PDPS database update successful for:

PGE name = 'PGE07'

Ssw version = '0'

ESDT = 'PGEEXE'

ESDT Version = '0'

staged file = '/home/emcleod/SSEP/MODPGE07.tar'

metadata file = '/home/emcleod/SSEP/MOD_PR10.tar.met'

```
  Top level shell name = 'MOD_PR10.exe'
Inserted at UR:
'UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[MDC:DSSDSRV]:14:LM:PGEEXE:1787'
Hit Enter to run again, 'q <Enter>' to quit:
```

This page intentionally left blank.

PGE Planning Processing and Product Retrieval

Using the Production Request Editor

When standalone tests (Run from the command line) have completed successfully and information about the PGE has been entered into the PDPS Database (through PGE registration), the PGE is ready to be run through the automated ECS PDPS environment.

To process Science data, a Production Request (PR) must be submitted to the ECS system. The Production Request Editor GUI accomplishes this function. Only one PR may be submitted at a time. A single PR is exploded by the PDPS into one or more jobs called Data Processing Requests (DPRs). The number of DPRs that are created for a single PR is determined by the number needed to cover the requested time interval, orbital extent and tile schema. Some PRs may only require one DPR.

********* Please be advised: Only one user per mode can be executing a DPR for a specific PGE at a time.** If more than one occurs the system will cancel the remaining DPR's. Problem discovered when chained PGE's failed to kickoff. *******

Invoking the Production Request Editor

Currently, the Production Request Editor is invoked from a command line script. Once the Production Request Editor is invoked, it brings up a screen with five tabs at the top for selection. The first tab is labeled "Planning". Selection of this tab displays a list of four capabilities available for the PR Editor by selecting the other tabs at the top of the primary GUI screen: PR Edit, PR List, DPR View, and DPR List.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The PGE has been registered in the PDPS Database.
2. The PGE has been successfully compiled and linked with the DAAC version of the SDP Toolkit.
3. The required servers are up and running.

To invoke the Production Request Editor GUI, execute the procedure steps that follow:

1 In any currently available xterm window, spawn a new session: type **xterm &** and then press the **Enter** key. Then **telnet** to the **PLS host.**

It is recommended that this procedure begin within a new command shell on a **PLS Host**.

Set the DISPLAY environment variable. At the UNIX prompt on the **PLS host (e.g. p0pls01**), type **setenv DISPLAY clientname:0.0** and then press the **Enter** key.

2 At the UNIX prompt on the PLS host (e.g**. p0pls01**), under the directory where the scripts are located (e.g. **cd /usr/ecs/TS1/CUSTOM/utilities**), type **EcPlPRE_IFStart *mode application_id* &**, then press **Enter**.

The ***mode*** is the operations mode (e.g**. TS1**).

The ***application_id*** is a numerical number **(**e.g**. 1).**

For example, **type EcPlPRE_IFStart *TS1 1 &*** then press the **Enter** key.

Various messages from the Production Request Editor may appear in this window as it is running. For this reason, avoid using this window for other tasks until the Production Request Editor has terminated.

3 In the Production Request Editor, click on one of the tabs **PR Edit, PR List, DPR View, or DPR** List corresponding to desired task.

To define a new Production Request or edit a Production Request, click **on PR Edit**. Proceed to Section Defining a New Production Request.

To review or list a Production Request, click on **PR List.**

To view or inspect a Data Processing Request, click on **View.**

To review or inspect a Data Processing Request, click on **List.**

4 When tasks are completed in the Production Request Editor GUI, click on the **File** menu, then choose **Exit**.

The Production Request Editor will disappear.

Refer to Section (Troubleshooting and General Investigation) if fail to bring up the Production Request Editor GUI.

Figure 42. Production Request Editor Introductory GUI.

Defining a new Production Request

A Production Request (PR) is a request for data production of granules between a start date/time and an end date/time for the time-based PGE (or a start orbit and an end orbit for the orbit-based PGE). A PR will explode into one or more Data Processing Requests (DPR) depending upon the time interval (for the time-based PGE) or orbit range (for the orbit-based PGE) involved. Each DPR corresponds to the execution of a PGE at one time. Therefore, a PR results in the execution of a PGE one or more times. Only one PGE is involved in a single Production Request.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The user has selected the **PR Edit** tab from the Production Request Editor.
2. The PGE involved in the Production Request has been registered in the PDPS database.

To define a new Production Request, execute the procedure steps that follow:

1 From the Production Request Editor GUI, click on the **PR Edit** tab.

The PR Edit page will be displayed as shown in Figure. 39.

2 In the field labeled **PR Name:**, enter **New** or verify that **New** is already entered as the default.

3 The PGE for the Production Request must be selected from a list. To do this, click on the **PGE…** button.

A GUI labeled PGE Selection will be displayed within which registered PGEs will be listed. The appropriate **PGE** can then be selected by clicking on **it** and then on the **OK** button.

The selected PGE will then be used to populate Satellite Name, Instrument Name, PGE Name, and PGE Version fields of the main GUI.

4 In the field labeled **Priority:**, enter ***priority***.

The ***priority*** is the priority to be assigned to this Production Request in the range **0** through **999** with **0** being the highest priority and **999** the lowest. For example, enter **40**.

5 In the Production Request Editor GUI, a **Duration** option is selected automatically based on the PGE registered into the PDPS. Two options are provided:

UTC Time for time range.

Orbit for orbit number range.

6 In the Production Request Editor GUI, enter ***StartDate*** and ***StartTime*** in fields labeled **Begin**, respectively.

The ***StartDate*** and ***Starttime*** are the start date and time of the Production Request and should be entered in the **mm/dd/yy** and **hh:mm:ss** formats.

7 In the case of **UTC Time Duration**, enter ***EndDate*** and ***EndTime*** in fields labeled **End**, respectively.

The ***Enddate*** and ***EndTime*** are the end date and time of the Production Request and should be entered in the **mm/dd/yy** and **hh:mm:ss** formats.

8 In the case of Orbit duration, enter ***StartOrbit*** and ***EndOrbit*** in fields labeled **From** and **To**, respectively.

The ***StartOrbit*** and ***EndOrbit*** are the orbit range of the Production Request.

9 Optionally, enter ***Comment*** in field labeled **Comment:**.

This comment will be displayed whenever this Production Request is brought up and viewed.

10 When Production Request is complete, click on **File** menu and select **Save As…**.

A GUI labeled File Selection will be displayed.

In the field labeled Selection, enter a user-defined name to be assigned to the Production Request. Then click on the OK button. A message box will be displayed stating "Production Request Explosion into DPRs ok***, n*** DPRs Generated", where ***n*** will be the number of DPRs (e.g. a 2-hr PR time will generate 24 DPRs for the 5-min processing period). Click on the **Ok** button. A second message box stating "Write to Database of Production Request ok"; again click **Ok.**

Note that you will not be allowed to enter a PR name that already exists. PR names that already exist will be displayed in the main window. The Production Request will then be saved under the name specified.

Refer to Section 22 (Troubleshooting and General Investigation) if fail to generate the DPRs.

11 When tasks are completed with the Production Request Editor GUI, click on the **File** menu, then choose **Exit**.

The Production Request Editor GUI will disappear.

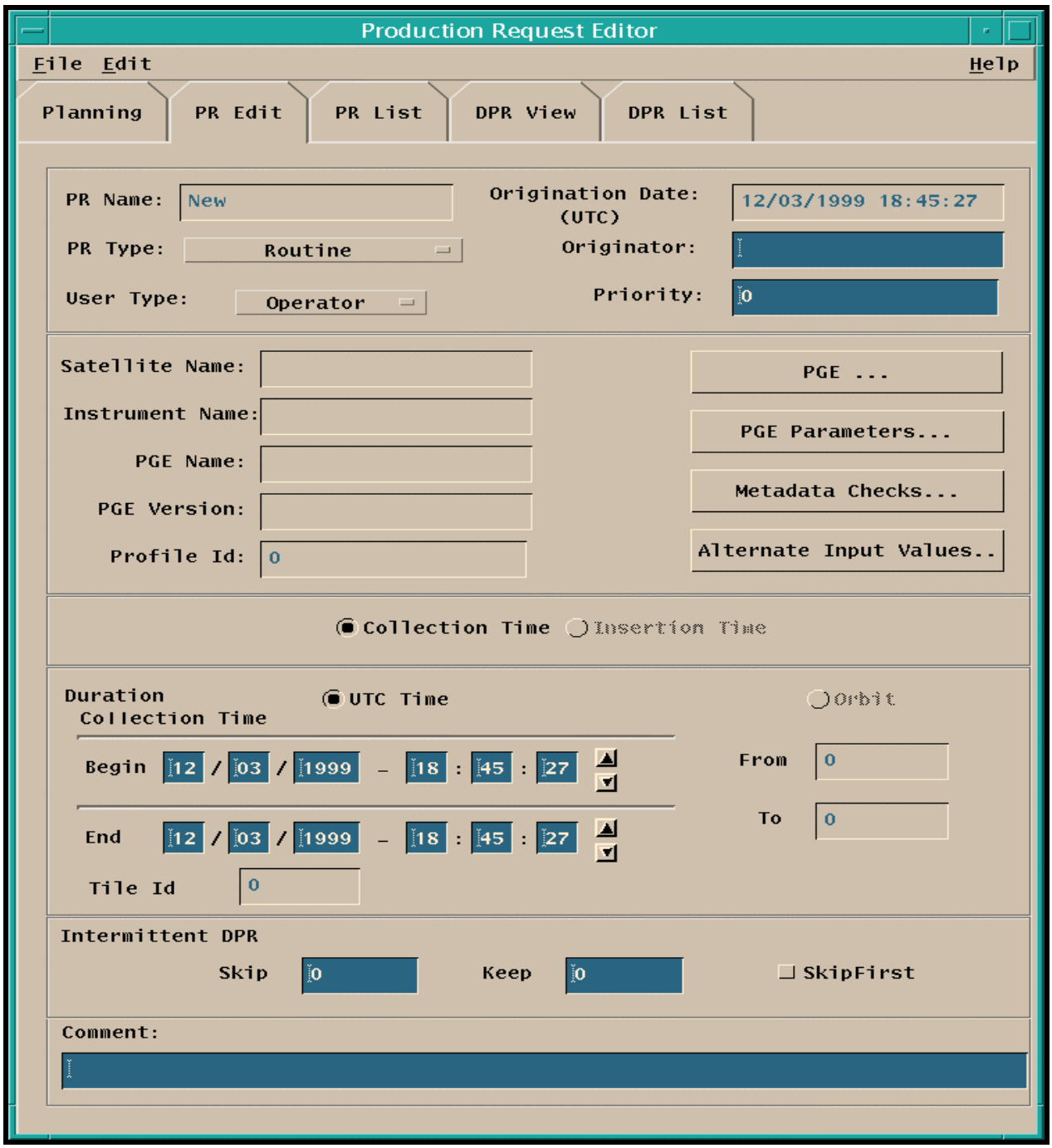


Figure 43. PR Editor GUI (Planning).

Processing

Once a candidate plan has been activated, each of the DPRs will result in subscriptions to the Data Server for the data needed. A request will go to the Data Server asking for notification when the required input data arrives.
Planning knows what data to request from the Data Server because the PDPS database stores this information as determined by the ESDT for each PGE. When the Data Server receives new data, it routinely checks to see if there are any outstanding subscriptions. If there are subscriptions, Planning will be notified. Once the input data required by a DPR becomes available, the DPR can be queued for processing.

Staging - The Data Processing Subsystem requests that the required input data, PGE (binary executables and shell scripts) and SDP Toolkit files be placed on a disk set aside for processing.

Process Control File (PCF) - establishes a linkage between logical Ids that the science software uses and the physical files that exist on the staging disk.

After the PGE has completed, the DPS will deallocate resources.

A Production History file will be created and will contain information concerning the conditions that the data products were generated by the PGE.

Viewing a New Production Request

This procedure describes how to view PRs that have already been defined. It assumes that the **PR List** tab has been selected from the Production Request Editor.

The information listed for each PR is:

PR Name - The name assigned to the Production Request when it was defined.

PGE ID - The name of the PGE involved in the PR.

Priority - The priority (0 - 99) of the PR assigned when it was defined.

Start - The start date and time of the PR.

End - The end date and time of the PR.

Comment - Any comment that was entered when the PR was defined.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The user has selected the **PR List** tab from the Production Request Editor.

To view Production Requests, execute the procedure steps that follow:

1 From the Production Request Editor GUI, click on the **PR List** tab.

The PR List page will be displayed as shown in Figure 44.

2 View the listed PRs. Optionally, find a PR by entering a search string in the field next to the **Find** button and then clicking on the **Find** button.

3 To modify a PR listed, click on the PR in the list and from the **File** menu select **Save As…**.

In the **File Selection GUI**, replace the current PR name shown in the **Selection field** with a new PR name. Then click on the **OK** button.

When modifying an existing PR, it must be saved under a new **PR** name.

Next, click on the **PR Edit** tab. The PR Edit page will be displayed with fields populated from the existing PR name, but having the new **PR** name chosen above.

See Section on using the **PR Edit** page and saving any changes made.

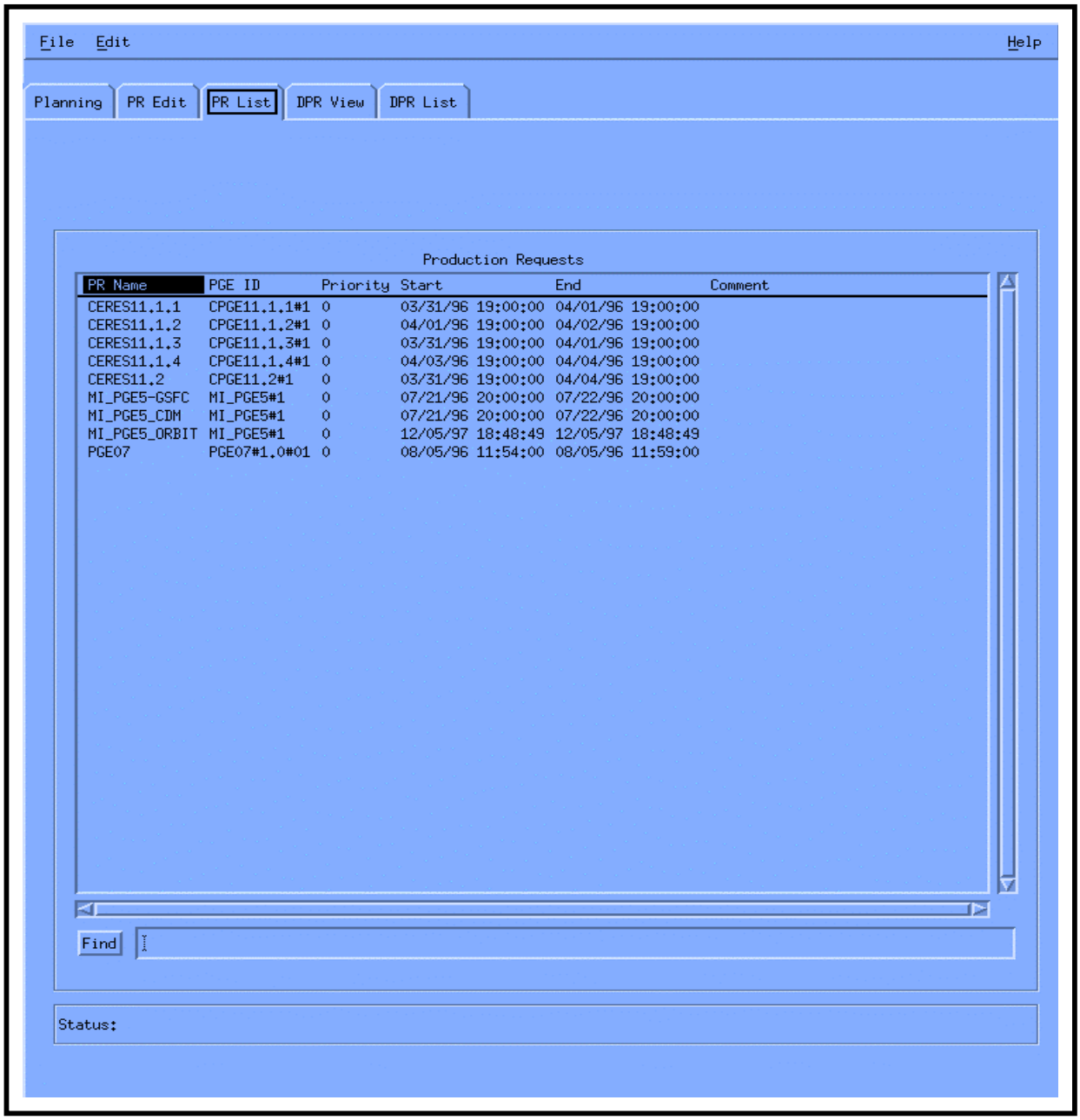


Figure 44. PR List GUI.

Viewing Data Processing Requests

Clicking on the DPR View GUI tab displays a list of all DPRs for all **PR**s entered into the system.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

To invoke the DPR View GUI, execute the procedure steps that follow:

1 From the PR Editor GUI, click on **DPR View** tab. The following information will be displayed:

Data Processing Request Identification.

PGE ID and its parameters.

Request Data and Status.

2 Listing Data Processing Requests

Selection of one PR on the PR List by highlighting it and then clicking on the **DPR List tab**, brings up a detailed display of all DPRs associated with the selected PR. These may be examined in order to develop production plans and schedule jobs.

3 To invoke the DPR List GUI, execute the procedure steps that follow:

From the PR Editor GUI, click on **DPR List** tab**.** The following information is displayed:

DPR Id.

PGE Id.

PR Name.

Tile Id.

Data Start Time (UTC).

Data Stop Time (UTC)

Using the Production Planning Workbench

The Production Planner uses the Production Planning Workbench to create new production plans and display a planning timeline.

Creating and Activating a Production Plan

The Production Planner creates a plan for production data processing at the DAAC by selecting specific PRs whose DPRs are to be run. The planning tool provides a forecast of the start and completion times of the jobs based upon historical experience in running these PGEs. Through the planning tool, when the generated plan is “activated,” the information included in the plan is transferred to the Data Processing subsystem and loaded into the Platinum AutoSys tool where production processing is managed.

The Production Planner creates the plan by selecting PRs from two lists of PRs, i.e., the list of available "Unscheduled" PRs and the list of "Scheduled" PRs. Using arrow buttons, the Production Planner moves the PRs between lists until the "Scheduled" list contains the desired set of PRs that define the new plan. Only one user can use the Planning Work Bench at a time. It is recommended for SSI&T that only one person do the planning for the group.
Before creating a new production plan the Production Planner must have available the following information:

Name of the plan.

Comments (if any).

PRs to be included in the new production plan

The plan is deactivated without activating another plan.

Using the Planning Workbench to Run One PGE

Once a PGE has been fully registered, its test data files have been inserted to the Science Data Server, and a single Data Processing Request (DPR) has been generated, the Planning Workbench can be used to plan for one execution run of a single PGE.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The required servers of the ECS System are up and running.
2. A DPR has been generated successfully.

To use the Planning Workbench to run one PGE, execute the procedure steps that follow:

1 In any currently available xterm window, spawn a new session: type **xterm &** and then press the **Enter** key. Then **telnet** to the **PLS host.**

It is recommended that this procedure begin within a new command shell on a PLS Host.

Set the DISPLAY environment variable. At the UNIX prompt on **the PLS host (e.g. p0pls01**), type **setenv DISPLAY clientname:0.0** and then press the **Enter** key.

2 Set the UNIX environment variable. At the UNIX prompt on the **PLS host**, change to the directory where the scripts are located (e.g. **cd /usr/ecs/TS1/CUSTOM/utilities**), then,

3 At the UNIX prompt on an **PLS Host**, under the directory where the scripts are located (e.g**. cd /usr/ecs/TS1/CUSTOM/utilities)**, type **EcPlAllStart <mode> <application_id>**, press **Enter**.

- The ***mode*** is one of modes used in the ECS system. e.g. **TS1**.
- The ***application_id*** is a numerical number. (e.g. **1**).
- For example: **EcPlAllStart TS1 1, Enter**

A Planning Workbench GUI will be appeared as shown in Figure 45.

A Planning Master Timeline GUI will also be appeared.

4 In the **Planning Workbench GUI**, go to the subwindow labeled **Unscheduled** and click on a Production Request name.

The **Production Request name** is the name under which the **PR** was saved in Section Defining a New Production Request.

The **PR** name entry will be highlighted.

5 In the Planning Workbench GUI, click on the button next to the label **Schedule** (the button has an inverted triangle on it).

The **PR** highlighted in step 3 will appear in the subwindow labeled **Scheduled.**

6 In the Planning Workbench GUI, click on the **Activate** button

A small GUI labeled Plan Activation will be displayed.

Click on the **Save** button.

Click on the **Activate** button again.

7 In the **Plan Activation GUI**, set the time in the time field forward to allow ample time for the PGE to run. Then click on the **Ok** button.

All that is necessary is for there to be sufficient time for the PGE run. There is no penalty for allowing ***too*** much time (e.g. Activate Start: 01/01/1990 00:00:00, Activate Stop: 01/01/1999 00:00:00).

The Production Request thus planned will be submitted to processing and its progress can be monitored with **AutoSys**.

Refer to Section 22 (Troubleshooting and General Investigation) if fail to schedule the DPR jobs.

8 When tasks are completed with the Planning Workbench GUI, click on the **File** menu, then choose **Exit**.

The Planning Workbench GUI will disappear.

9 To terminate the processes of Planning Workbench GUI, at the UNIX prompt on an PLS Host, type **EcPlSlayAll *mode***, then press **Enter**.

Important Note **:** The creation of Production Requests and Plans requires close coordination among all who are using the same mode in SSIT. Otherwise, the submittal of a Plan may prevent a waiting DPR from starting. In particular, when submitting a new Plan in a mode being shared with others, one should:

1. Ask everyone else if they have a DPR awaiting data as part of a chain.
2. Check the PDPS database table PlDataProcessingRequest for any DPRs that are in state CQ_HOLD and include these in the new Production Request and Plan.
3. Also, include any DPRs (except those marked SUCCESS) upon which the given DPR depends in the new Production Request and Plan.

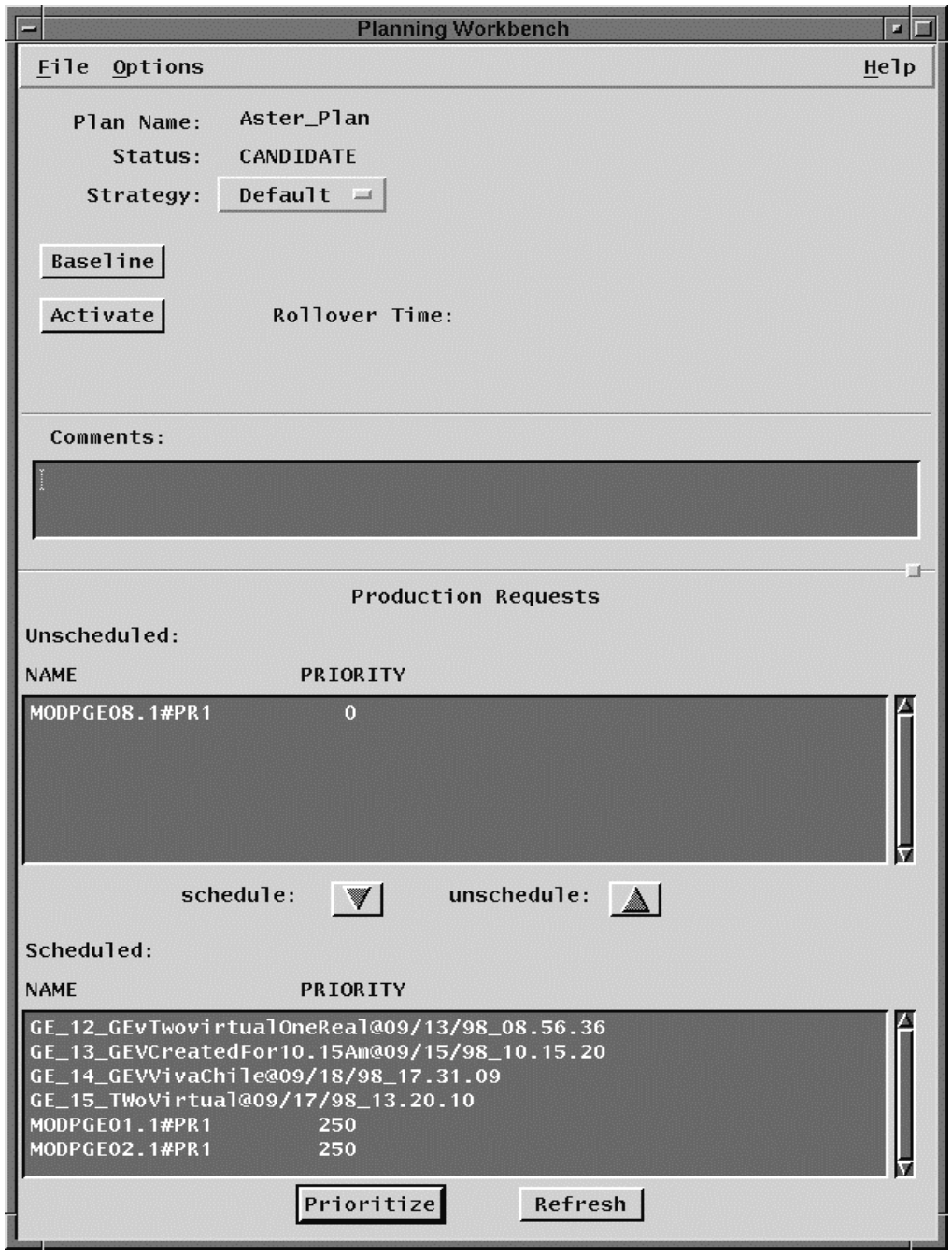


Figure 45. Planning Workbench GUI.

Monitoring Production

The progress of one or more PGEs running within the PDPS may be monitored. The COTS tool used for this purpose is **AutoSys®** by Atria Software. Each Data Processing Request results in seven AutoSys jobs that are boxed together. An AutoSys job name follows the template:

PGEname#Suffix

where ***PGEname*** is replaced by the name of the PGE, and ***Suffix*** is a character indicating the job phase of the **DPR.** Refer to Table 3.

For example, for a scheduled PGE named **MOPITT4**, the AutoSys jobs making up that DPR would be:

MOPITT4#A
MOPITT4#S
MOPITT4#P
MOPITT4#E
MOPITT4#p
MOPITT4#I
MOPITT4#D

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The required servers of the ECS System are up and running.
2. A DPR has been scheduled successfully.

To monitor production, execute the procedure steps that follow:

1 In any currently available xterm window, spawn a new session: type **xterm &** and then press the **Enter** key. Then telnet to the **DPS host**.

It is recommended that this procedure begin within a new command shell on a **DPS Host**.

Set the DISPLAY environment variable. At the UNIX prompt on the **DPS host (e.g. p0pls01),** type **setenv DISPLAY clientname:0.0** and then press the **Enter** key.

2 At the UNIX prompt on the DPS Host, under the directory where the scripts are located (e.g. **cd /usr/ecs/TS1/CUSTOM/utilities**), type **EcDpPrAutosysStart** ***mode application_id*** **&**, then press **Enter.**

- The ***mode*** is one of modes used in the ECS system (e.g**., TS1**).
- The ***application_id*** is a numerical number (e.g., **1**).
- For example: **EcDpPrAutosysStart TS1 1, Enter.**

The ***mode*** is one of modes used in the ECS system (e.g., **TS1**).

For example: EcDpPrAutosysStart TS1, Enter.

A GUI labeled **AutoSys** will be displayed.

This GUI will contain eight buttons for invoking various tools available under AutoSys.

3 In the AutoSys GUI, click on the **Ops Console** button.

A GUI labeled AutoSys Job Activity Console GUI will be displayed.

The main subwindow of this GUI will contain a dynamically updated list of AutoSys jobs (seven jobs make up one DPR) currently scheduled.

To disable dynamic updating of the main subwindow (which may be distracting), click on the View menu, then choose Select Jobs. A GUI labeled Job Selection will be displayed. Under the label Select by Name, click on the square labeled All jobs, then click on the OK button.

DPRs will be listed in the column labeled Job Name and their statuses (e.g. SUCCESS) will be listed in the column labeled Status.

To view job status information for a particular DPR, click on a DPR. Below the main subwindow, available job status information will be displayed.

To view the existing event report, under the label Reports, click on the middle diamond labeled Event. The current event status for the selected DPR will be displayed in the subwindow labeled Event Report. Alternatively, the summary report for the selected DPR will be displayed by clicking on the middle diamond labeled Summary.

To view the job definition, under the label Show, click on the Job Definition tab. A GUI labeled Job Definition will be displayed. The selected DPR is shown in Job Name. To kill the DPR, click on the Delete tab. *(Warning: Be very careful to avoid deleting wrong DPR!)*

Exit the AutoSys Job Activity Console by clicking on the Exit button.

4 In the AutoSys GUI, click on the **JobScape** button.

A GUI labeled JobScape will be displayed.

The main subwindow of this GUI will contain a dynamically updated list of AutoSys jobs (seven jobs make up one DPR) currently scheduled. The colors indicate the job statuses for DPRs and their jobs at present.

There are eleven job statuses: ACTIVATED, STARTING, RUNNING, SUCCESS, FAILURE, TERMINATED, RESTART, QUE_WANT, ON_ICE, OFF_HOLD, and INACTIVE. The color chart is on the left of GUI.

To view job status information for a particular DPR (or job) in detail, click on a **DPR (or job)**, then click on the **Job Console** button. A GUI labeled Job Console will be displayed. The information shown here is similar to that shown in Ops Console as mentioned above.

To change the status of a job for a particular DPR, click on a job under that DPR, then press the **right button** of the mouse. Choose one of the functions in the lower part of the menu. For example, to hold a job, choose **On Hold**, then click the **Yes** button.

Suggestion: For a scheduled DPR, hold all jobs for that DPR except the first one (Allocation). After the first job runs successfully, release the next job (Staging) by choosing **Off Hold**. Repeat

until all jobs are done. Therefore, if a job fails, it can be fixed and rerun without interfering with the others.

This is especially important for the Postprocessing job. If the PGE fails and the Postprocessing job is not **On Hold,** the DPR will have to be deleted and a new one created. This happens because PDPS will have deleted all references to the output granules in the PDPS database.

To exit the **JobScape,** click on the **File** menu, then choose **Exit.**

5 In the AutoSys GUI, click on the TimeScape button.

A GUI labeled TimeScape GUI will be displayed.

The main subwindow labeled Job Name of this GUI will contain a dynamically updated list of **AutoSys jobs** (seven per DPR) currently scheduled.

To disable dynamic updating of AutoSys jobs (which may be distracting), click on the **Freeze Frame** button.

The color of each job indicates its status according to the legend on the left side of the GUI.

The time line is shown on the right side of the GUI with time marked at the top. A red vertical line (dashed) indicates the current time.

Exit the TimeScape GUI by clicking on the File menu and selecting **Exit.**

6 To quit AutoSys, in the AutoSys GUI, click on the **Exit** button.

The AutoSys GUI will disappear.

Refer to Section (Troubleshooting and General Investigation) if any job fails on AutoSys.

7 Use the "**tail -f em.log**" to create a permanent record of a log file if debugging is necessary. Note: the **xx.log file** has to be created first: "**vi em.log**"

8 To look at the Data Base type: **setenv MODE <MODE>, cd utilities**
source EcCoEnvCsh, cd dbr, dbrowser-syb <MODE> 2 &

9 Scrips to restart the servers and subsystems are located in: **/home/cmops/restart/drop4/**

Using the Q/A Monitor

The Q/A Monitor allows the output products produced during a PGE run to be accessed and examined. Input test data granules and Production History files can be retrieved in the same manner. The Q/A Monitor retrieves output products based on the collection name (*i.e.* the ESDT) and time of Insertion to the Science Data Server.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The required UNIX environment variables have been set properly.
2. The desired output products have been successfully Inserted to the Science Data Server.

To use the Q/A Monitor, execute the procedure steps that follow:

1 **telnet to (PDPS) p0sps06** or from the SSIT Manager, click on the **Tools** menu, then choose **Xterm**. Then **telnet** to a **PLN Host.**

Alternatively, in any currently available xterm window, spawn a new session: type **xterm &** and then press the **Enter** key. Then **telnet** to the **PLN Host**.

2 Set the DISPLAY environment variable: **setenv DISPLAY clientname:0.0** and then press the **Enter** key**.**

Type setenv <mode> (e.g..,TS1). clientname

Type **source** environment_setup_file **(**e.g., **EcCoEnvCsh** for C shell users).

login: ID, password:

Change the working directory to the location where the script for starting the Autosys is stored. (e.g. **cd /usr/ecs/<mode>/CUSTOM/utilities).**

3 At the UNIX prompt on the PLN Host, type: **EcDpPrQaMonitorGUIStart <MODE> <Q/A Monitor Instance>, example: TS1 1, &** and then press the **Enter** key.

The ***mode*** is the operations mode.

The **Q/A Monitor Instance** e is a numerical number. (e.g. **1**).

The **Q/A Monitor GUI** will be displayed.

Various messages from the Q/A Monitor will appear in this window as it is running.

4 In the Q/A Monitor subwindow labeled **Data Types,** select an **ESDT** from the list presented and **click** on it.

Use the **scroll bars** if necessary to locate desired **ESDT.**

Optionally, use the Find field and button to locate an ESDT.

5 In the Q/A Monitor, under the label **Data Granule Insert Date (mm/dd/yy)**, set the date range within which the search for granules of the ESDT selected will be conducted.

The date range can be made arbitrarily large to select all granules of a particular collection (**ESDT**).

The dates refer to date of granule Insert to the Science Data Server.

6 In the Q/A Monitor, click on the **Query** button.

The results of the query will be displayed in the bottom window, labeled **Data Granules.**

All granules having the ESDT selected in step 3 and having Insert times within the date range specified in step 4 will be listed in this window.

7 In the Q/A Monitor subwindow labeled **Data Granules**, click on one of the data granules listed to be examined.

The data granule selected will be highlighted.

8 To retrieve the data granule's Production History file, click on the **Retrieve Prod History** button.

The **Production History (PH) tar file** corresponding to the selected data granule will be retrieved from the **Science Data Server** and placed on the local machine (a **PLN Host**) in the directory **/var/tmp.**

The **PH** can then be moved or copied manually from the **/var/tmp** directory to a user working directory for examination.

Only the **PH** file is retrieved with the **Retrieve Prod History** button.

If only the **PH** is desired, exit this procedure. To retrieve the data granule itself, continue on to step 7.

9 To retrieve the data granule, click on the **Retrieve Data Granule** button and note its file name (listed in the entry for the granule; you may have to scroll over to the right to see it).

The data granule selected will be retrieved from the Science Data Server and placed on the local machine (a **PLS Host**) in the directory **/var/tmp**.

A granule of any format (**binary, ASCII, HDF, HDF-EOS)** may be retrieved in this manner. Only **HDF and HDF-EOS** granules, however, may be further visualized using **EOSView** as described in the next steps.

10 To examine the data granule, click on the **Visualize data** tab.

The Visualize data page will be displayed.

11 In the main subwindow on the **Visualize data** page, locate the file name of the granule retrieved in step 8 and **click** on it.

The item selected will be highlighted and will appear in the Selection subwindow below.

12 Click on the **Visualize** button.

This action will invoke **EOSView** with the granule selected.

The granule must be **HDF** or **HDF-EOS** format.

13 When tasks are completed with the **Q/A Monitor GUI**, click on the **File** menu, then choose **Exit**.

Postprocessing and General Investigation

An important part of SSI&T is verifying that the output files produced at the DAAC are identical (within particular tolerances) to the test output files delivered with the DAPs. A successful comparison is a strong indication that the porting of the science software from the development facility at the SCF to the operational facility at the DAAC has not introduced any errors.

A number of file comparison tools are available during SSI&T via the SSIT Manager GUI or they can be invoked from the UNIX command line. Two tools are available for comparing HDF or HDF-EOS files, one tool for comparing ASCII files, and another tool for assisting in comparing binary files.

It is assumed that the Instrument Team has delivered test output files (produced at their SCF) with which to perform the comparison.

Examining PGE Log Files

Three log files are produced by PGEs during runtime: the Status log, User Log, and the Report log. These log files are written by the SDP Toolkit and by the science software using the Toolkit's Status Message Facility (SMF). The location of these log files is specified in the Process Control File (PCF). When the PGE is built and run with the SCF version of the SDP Toolkit, the location and file names of the log files can be set as desired. When the PGE is built with the DAAC version of the SDP Toolkit and run within the PDPS, the location and file names of the log files is set by the system in the instantiated PCF.

The Status log file captures all error and status information. The User log file captures a subset of messages which are more informational. The Report log file captures arbitrary message strings sent by the PGE.

The section aforementioned describes how to examine log files produced by PGEs that have been built with the SCF version of the SDP Toolkit and run from the command line.

The section aforementioned describe how to examine log files (within the Production History) produced by PGEs that have been built with the DAAC version of the SDP Toolkit and run within the PDPS.

Examining PDPS-Related Scripts and Message Files

This section describes how users may access files, in addition to the PGE-produced log files, which are created during the execution of a DPR job and which may hold information useful in tracing processing problems.

Some of these files are written by default to directory paths that can only be accessed on either the SGI processor machine or one of the Sun workstations. More detailed descriptions of these files and the conditions under which they are generated will be supplied in future Green Book versions.

ECS Mechanisms for Capturing Information about a PGE's Execution

Note: This information was provided by ECS expressly for this document. It describes two tar files which are generated by the ECS Planning and Data Processing System (PDPS). One of these file types is generated for each execution of a PGE that is run within the ECS environment. The particular file type that is generated depends on the success or failure of the PGE execution. The ECSDataGranule metadata contains a reference to the Production History.

ECS Production History (PH)

For each successful execution of an instance of a PGE (represented by a data processing request - dpr), the processing history is captured in the production history tar file. The production history tar file is generated and archived by ECS (inserted into the Data Server) upon PGE completion. The PH is a UNIX tar file. The PH contains multiple files. Each PH can be uniquely retrieved from the Data Server. A science user has the option to acquire the PH when ordering a science granule. The name of the tar file will include a UR for the PH.

The tar file can be untarred and contains numerous component files. After untarring the PH, the file names of the components provide the traceability to a particular dpr (data processing request or job) that ran under ECS. Please note that the dprid is a concatenation of an abbreviated form of the PGE name (e.g., for MODIS PGE01, it would be MoPGE01#version) with a date/time stamp which is the start of the processing time for the dpr. The dprid string may have some trailing 0's filled in.

A PHcomponentFile can be examined to identify the components within the PH.

Other component files are:

The PH log file or processing log: Named PGEname#versionMMDDhhmm.Log – contains the DPR ID, the actual command used to run the PGE (.in), resource usage information, and the PGE exit status. It also contains a listing of output products generated, their file paths and file sizes.

The PCF file: Named PGEname#versionMMDDhhmm.Pcf – contains the actual instantiated PCF used when running the instance of the PGE (this dpr). Note that the PCF contains URs for all inputs to this execution of the PGE. There is one UR for each input granule. If a granule is a multi-file granule, the same UR will appear (repeat itself) for each inputs file of the granule.

The Production Log file: Named PGEname#versionMMDDhhmm.ProdLog – Contains the DPR ID, the PGEID, and resource usage information (same as in the .Log file). Resource usage information includes:

CPU time in application
CPU time in system
Physical memory
Max. resident set size
Avg. shared text size
Avg. shared data size
Avg. shared stack size
Page reclaims

Page faults
Swaps
Block input ops
Block output ops
Messages sent
Messages received
Signals received
Voluntary context switches
Involuntary context switches

The Profile file: Named PGEname#versionMMDDhhmm.Profile – Contains the environment variables defined during the execution of the PGE including the contents of the PATH environment variable.
The SDP Toolkit log files:

1) The Report log file: Named PGEname#versionMMDDhhmm.TkReport – This is the same log file that the SDP Toolkit generates when the PGE runs outside of ECS. (See SDP Toolkit documentation.)

2) The Status log file: Named PGEname#versionMMDDhhmm.TkStatus – This is the same status log file that the SDP Toolkit generates when the PGE runs outside of ECS (See SDP Toolkit documentation.)
3) The User log file: Named PGEname#versionMMDDhhmm.TkUser – This is the same user log file that the SDP Toolkit generates when the PGE runs outside of ECS (See SDP Toolkit documentation.)

ECS Failed PGE Tar File

For each unsuccessful execution of an instance of a PGE (represented by a data processing request or dpr), a failed PGE tar file (may also be called the History Log or HL) is created as a diagnostic tool for the science software provider/analyst. The Failed PGE tar file is generated and archived by ECS (inserted into the Data Server) upon the abnormal completion of the PGE execution. Similar to the PH, the Failed PGE tar file is a UNIX tar file. An Instrument Team user would acquire a FailedPGE tar file by interacting with the DAAC operations staff. The name of the tar file will include an UR for the FailedPGE tar file.
The Failed PGE tar file contains numerous component files which can be examined after untarring. After untarring the FailedPGE tar file, the PHcomponentFile can be examined to identify the components within. The component files are the same as those in the PH. In addition, if the PGE's execution ended with a core dump, the core file is included in the tar file. Since the FailedPGE tar file results from a failed execution of a PGE, the contents will reflect the point of the failure.
Operationally, the ECS SDSRV error logs are examined to view the PH or Failed PGE tar files that were generated.
The error log entries referencing a PH tar file are of the format:
:PH.version:dbid:1.BINARY where version is the ESDT version number for the Production History ESDT, and dbid is the database ID, a unique identifier within SDSRV.
The error log entries referencing a FailedPGE tar file are of the format:

:LM: FAILEDPGE.version:dbid:1.BINARY where version is the ESDT version no. for the FailedPGE ESDT, and dbid is the database ID, a unique identifier within SDSRV
Operations staff may use the dbid to access the corresponding tar file itself from the data server. The file may then be untarred.

Examining AutoSys JIL Scripts

JILxxxxxxxxx is the Job Information Language (JIL) script that defines the DPR job to **AutoSys** and which must be submitted to the **AutoSys** Database before a DPR job can be run. The name of the file created is system-generated and begins with the characters 'JIL' followed by nine characters (e.g. JILAAAa0066c).

Sample file content:

```
insert_job: 5251_823122483_1
job_type: command
command:        /usr/ecs/{mode}/CUSTOM/data/bin/sgi/EcDpAtExecutionMain
   5251_823122483_1
machine: spr1sgigsfc
std_out_file: /home/cboettch/mockpge_msfc/out/dpat_std.out
std_err_file: /home/cboettch/mockpge_msfc/out/dpat_std.err
profile: /usr/ecs/<MODE>/CUSTOM/data/bin/sgi/EcDpAtRunProfile.sh
```

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

To examine **JILxxxxxxxx scripts** on the AIT Sun, execute the procedure steps that follow:

1 At the UNIX prompt on an AIT Sun, type **cd *JILscriptPathname*** and then press the **Enter** key.

The ***JILscriptPathname*** is the full path name to the location of the **JILxxxxxxxx** scripts to be examined.

2 At the UNIX prompt on the AIT Sun, type **vi *JILscriptFilename*** and then press the **Enter** key.

The ***JILscriptFilename*** is the file name of the **JILxxxxxxxx script** to be examined.

This brings up the file named ***JILscriptFilename*** in the ***vi* editor.**

Any text editor may be used such as ***emacs.*** For example, emacs ***JILscriptFilename*** and then press the **Enter** key.

Examining Application Log Files (ALOG)

Most of the custom code used during SSI&T routinely produce log files. For example, the SSIT Manager produces a log file named **EcDpAtMgr.log** and the tool used to Insert SSEPs to the Data Server (EcDpAtInsertExeTarFile.sh) produces a log file named **EcDpAtInsertExeTarFile.log.** These files are placed in the directory in which the tool was executed. If the **SSIT Manager** is run from the user's home directory, then the log files for each of the associated tools will be found in the user's home directory. Log files are produced at the first invocation of the tools, even if no messages are written to them. During subsequent use of the tools, the associated log files will be appended.

Log files are generally named according to the convention:

***ApplicationName*.log**

where ***ApplicationName*** is replaced with the name of the tool's executable binary. For tools that are shell scripts **(*e.g.* .sh files),** the shell name is left out of the log file name. For example, the tool EcDpAtInsertStaticFile.sh produces a log file named **EcDpAtInsertStaticFile.log** and not EcDpAtInsertStaticFile.sh.log.

Where an **SSIT Manager** application has been run using login **ID, PW**:**,** the path will be found using: **/usr/ecs/{MODE}/CUSTOM/logs/.**

Server connectivity failures have been encountered when installing **ESDT's , MCF's** and **.met files**. The term bounce the servers has been widely used in conjunction with the effort to re-install or delete files. **Bounce** means to **shut down a server** and then **bring them back up** to rid the servers of unwanted or old bindings.

The nature of what needs to be done is outlined as follows:

1 Install or Delete **ESDT's -** the **SDSRV** and **ADSRV** need to be bounced after installation or removal of **ESDT**'s to allow for a refresh of the **DCE** cell management.

2 For **PGE…..odl, MCF's** and **.met files,** bouncing the servers **SDSRV** and A**DSRV** need to done after installation and reinstallation.

3 This can be done by logging into **ECS Assistant** for each server. The login should be with generic **ID:** and **PW**:**,** and then press **Enter** Key**.**.

This page intentionally left blank.

File Comparison and Data Visualization

File Comparison Overview

An important part of SSI&T is verifying that the output files produced at the DAAC are identical (within particular tolerances) to the test output files delivered with the DAPs. A successful comparison is a strong indication that the porting of the science software from the development facility at the SCF to the operational facility at the DAAC has not introduced any errors.

A number of file comparison tools are available during SSI&T via the SSIT Manager GUI or they can be invoked from the UNIX command line. Two tools are available for comparing HDF or HDF-EOS files, one tool for comparing ASCII files, and another tool for assisting in comparing binary files.

Using the GUI HDF File Comparison GUI

The following is a list of tools, and or assumptions:

The SSIT Manager is running.

Two HDF or HDF-EOS files exist with similar structures.

The Instrument Team has delivered test output files.

If either of the two HDF/HDF-EOS files is in the ClearCase VOB, a ClearCase view was set before the SSIT Manager was started.

Comparing Two HDF or HDF-EOS Files Using the HDF File Comparison GUI

1 From the SSIT Manager, select **Tools→ Product Examination → HDF** from the menu.

The HDF File Comparison GUI window will be displayed.

2 In the HDF File Comparison Tool GUI, click on the **File 1** button.

Read the Systems Description document and the Operations Manual. Both of these or their equivalent should be in the delivery.

Using the hdiff HDF File Comparison Tool

The hdiff File Comparison Tool is a text-oriented tool run from the command line. It allows comparison of two HDF or HDF-EOS files

The following is a list of tools, and or assumptions:

1. The SSIT Manager is running.
2. Two HDF or HDF-EOS files exist with similar structures.
3. The instrument Team has delivered test output files.
4. If either of the two HDF/HDF-EOS files is in the ClearCase VOB, a ClearCase view was set before the SSIT Manager was started.

Comparing two HDF or HDF-EOS Files Using the hdiff File Comparison Tool

1 From the SSIT Manager, select **Tools→ Product Examination → File Comparison → HDF** from the menu.

The **HDF File Comparison Tool** window will be displayed.

An xterm window running ***hdiff*** will be displayed.

2 In the xterm window at the prompt **Options? (-h for help)**, type in any desired options then press the **Enter** key.

To see the list of available options, type -h then press the Enter key. to the prompt.

3 In xterm window at the prompt **1st file to compare?**, type ***filename1***, then press the **Enter** key.

The ***filename1*** is the file name of the first of two **HDF or HDF-EOS** files to be compared.

If ***filename1*** is not in the current directory (the directory from which the SSIT Manage was run), include the full path name with the file name.

4 In xterm window at the prompt **2nd file to compare?**, type ***filename2***, then press the **Enter** key.

The ***filename2*** is the file name of the second of two HDF or HDF-EOS files to be compared. Select another students' file.

If ***filename2*** is not in the current directory (the directory from which the SSIT Manage was run), include the full path name with the file name. The two files will be compared and the output will be displayed in the xterm window.

Using the ASCII File Comparison Tool

Most output files (products) from PGEs run in the DAAC will be in HDF-EOS format. A small minority may be in ASCII (text) format. The ASCII File Comparison Tool is a front-end to *xdiff* UNIX X Window tool for comparing two ASCII files.

Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

The following is a list of tools, and or assumptions:

1. The SSIT Manager is running.
2. Two ASCII files exist and have read permissions.
3. The instrument Team has delivered test output files.
4. If either of the two ASCII files are in the ClearCase VOB, a ClearCase view was set before the SSIT Manager was started.

Comparing Two ASCII Files

1 From the SSIT Manager, select **Tools→ Product Examination → File Comparison → ASCII** from the menu.

An xterm window running ***xdiff*** will be displayed.

2 In xterm window at the prompt **1st file to compare?**, type ***filename1***, then press the **Enter** key. Select a descriptor or mcf file in the directory with the PGE.

The ***filename1*** is the file name of the first of two ASCII files to be compared.

If ***filename1*** is not in the current directory (the directory from which the SSIT Manage was run), include the full path name with the file name.

3 In xterm window at the prompt **2nd file to compare?**, type ***filename2***, then press the **Enter** key.

The ***filename2*** is the file name of the second of two ASCII files to be compared. Select another student's corresponding file.

If ***filename2*** is not in the current directory (the directory from which the SSIT Manager was run), include the full path name with the file name.

A window labeled **xdiff** will be displayed.

4 In the window labeled **xdiff**, view the differences between the two files displayed.

File ***filename1*** will be displayed on the left side of the window. File ***filename2*** will be displayed on the right.

Only sections of file in which there are differences will be displayed. A "**bang"** character (!) at the beginning of a line indicates that a difference was found.

For further help on ***xdiff,*** type **man xdiff**, in an xterm window then press the **Enter** key.

Close the display window by using the pull down menu from the X window in the upper left corner.

5 In the xterm window at the prompt **Hit Enter for another diff, 'q <Enter>' to quit:**, type **q** press **Enter** to quit or just press **Enter** to perform another comparison.

Using the Binary File Difference Assistant

Most output files (products) from PGEs run in the DAAC will be in HDF-EOS format. A small minority may be in some binary format. The Binary File Difference Assistant aids the user in constructing code that allows comparison of binary output files. Since there is an unwieldy number of possibilities for binary file formats, this tool cannot compare two binary files without some custom code written at the DAAC, hence, the "Assistant" in the name. The Binary File Difference Assistant aids the user by generating a makefile, a driver module, and a template comparison module in C, FORTRAN 77 or IDL (Interactive Data Language). The user then edits these templates to read the particular binary format in question according to a SCF-supplied format specification.

The binary file comparison will not be performed during the SSIT training lesson.

The following is a list of tools, and or assumptions:

1. The SSIT Manager is running.
2. Two Binary files exist and have read permissions.
3. The instrument Team has delivered test output files.
4. If either of the two Binary files are in the ClearCase VOB, a ClearCase view was set before the SSIT Manager was started.

Comparing two Binary Files

1 From the SSIT Manager, select **Tools→ Product Examination → File Comparison → Binary** from the menu.

The Binary File Difference Assistant tool GUI will be displayed.

2 In the Binary File Difference Assistant tool GUI, click on one of the languages listed under the **Select Language** label. The choices are **C, FORTRAN, or IDL.**

The choice of language depends largely on preference. It does not necessarily have to be the language that was used to create the files being compared.

3 Optionally, click on either the **Image** button or the **Structure** button located under the label **Compare Function**.

Clicking on the Image button will display a code example for comparing binary files containing images.

Clicking on the Structure button will display a code example for comparing binary files containing structures or records.

The displayed listing well documented and should be read.

The language of the code will depend on the language selection made in step 2.

4 Optionally, click on either the **Image** button or the **Structure** button located under the label **Driver**.

Clicking on the Image button will display a code example for a driver invoking the compare function for binary files containing images.

Clicking on the Structure button will display a code example for a driver invoking the compare function for binary files containing structures or records.

The displayed listing well documented and should be read.

The language of the code will depend on the language selection made in step 2.

5 Optionally, click on either the **Help** button.

A Help window will be displayed.

To end help, click on the **Dismiss** button.

The Help window may remain displayed while using the **Binary File Difference Assistant.**

6 Once familiar with the code examples (steps 3 and 4), click on the **Copy** button.

A window labeled **Enter Unique ID** will be displayed.

In the field labeled **Enter unique file identifier:**, **type *fileID***, click on the **OK** button.

The ***fileID*** will be used in the file names of the files copied over. These files will be:

C:

DaacBinDiff_***fileID***.c	Compare function
DaacBinDiff_***fileID***_driver.c	Driver
DaacBinDiff_***fileID***.mak	Makefile

FORTRAN:

DaacBinDiff_***fileID***.f	Compare function
DaacBinDiff_***fileID***_driver.f	Driver
DaacBinDiff_***fileID***.mak	Makefile

IDL:

DaacBinDiff_***fileID***.pro	Compare function

DaacBinDiff_***fileID***_driver.pro	Driver
DaacBinDiff_***fileID***.sh	Shell script with here document

The files will be copied into the directory from which the SSIT Manager is being run.

7 Using any desired text editor, customize the files for the job at hand. Then build the executable using the customized makefile provided (for C and FORTRAN). Then run the program to perform the binary file comparison.

Data Visualization

In order to view the success of science software in producing scientifically valid data sets, the data needs to be displayed in forms that convey the most information. Data visualization enables this to be done.

There are two visualization tools provided to the DAAC: EOSView and Interactive Data Language (IDL). These tools are both accessible via the SSIT Manager. EOSView is user friendly GUI for creating two-dimensional displays from HDF-EOS objects(Grid, Swath) as well as the standard HDF objects (SDS, Vdata, Image, Text). It has additional features such as thumbnail panning, colorization, zooming, plotting, and animation. Only some aspects of data visualization will be addressed in this training material. For further information, see the related references.

IDL is a COTS display and analysis tool widely applied in the scientific community, It is used to create two-dimensional, three-dimensional (volumetric), and surface/terrain displays from binary, ASCII, and many other formats in addition to HDF.

Only a limited number of file types will be available during SSIT training.

Viewing Product Metadata with the EOSView Tool

This procedure describes how to use the EOSView tool to inspect the metadata in the HDF-EOS output file from a PGE. To view product metadata with the EOSView tool, execute the procedure steps that follow:

Viewing Product Metadata

In the event that the SSIT Manager does not support Data Visualization . A backup example is as follows for EOSView:

On workstation **x0ais##**, at the UNIX prompt in a terminal window, type as in step 1 below your **user id** and **password**.

NOTE: The **x** in the workstation name will be a letter designating your site:

g = GSFC, m = SMC, l = LaRC, e = EDC, n = NSIDC, o = ORNL, a = ASF, j = JPL; p=PVC, the ## will be an identifying two-digit number (e.g.,**g0ais01** indicates a Data Processing Subsystem (DPS) workstation at GSFC).

. Prior to the **rlogin**, enter **setenv DISPLAY <local_workstation IP address>:0.0.** The **<ipaddress>** is the ip address of **x0ais##,** and xterm is required when entering this command on a Sun terminal.

1 Log into an Algorithm and Test Tools (AITTL) environment using a machine so configured. At the **PVC** this machine is **p0ais01**

2 Telnet into **p0ais01.**

3 logon using your own **ID** and **Password**

4 **cd /usr/ecs/TS1/CUSTOM/eosview.**

5 Select **EOSView,** The EOSView GUI will be displayed**.**

6 **Use the select buttons to guide you toward the view desired**

The following is a list of tools, and or assumptions:

1. The SSIT Manager is running.
2. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools.

Viewing Product Metadata

1 From the SSIT Manager, select **Tools→ Product Examination → EOSView** from the menu.

The EOSView GUI will be displayed.

2 In the GUI labeled **EOSView - EOSView Main Window**, click on the **File** menu and select **Open.**

The Filter GUI will be displayed.

In the subwindow labeled Filter, select the appropriate directory and file to open.

3 A GUI labeled **EOSView - *MyOutputFile.hdf*** will be displayed where ***MyOutputFile.hdf*** is the file name of the file chosen in step 2. Once displayed, a list of **HDF** objects will appear in the main window. If nothing is listed, it means that no **HDF** objects were found within the file.

4 In the GUI labeled **EOSView - *MyOutputFile.hdf***, click on an **object listed** for which metadata is to be inspected. The object selected will be highlighted.

Do **not** double click on object since this will cause a **Dimension GUI** to be displayed instead.

5 The global metadata associated with the object selected will be displayed in a scrollable field by clicking on the **Attributes** menu and selecting **Global** in the GUI labeled **EOSView - *MyOutputFile.hdf***.

If instead, the message "Contains no Global Attributes" appears, then the selected object contains no global metadata.

6 Repeat steps 4 and 5 for each **HDF** object within the selected **HDF-EOS** file for which metadata is to be examined.

7 In the GUI labeled **EOSView - *MyOutputFile.hdf***, click on the **File** menu and select **Close** to close the **EOSView - *MyOutputFile.hdf*** GUI.

8 In the GUI labeled **EOSView - EOSView Main Window**, click on the **File** menu and select **Exit** to exit **EOSView - EOSView Main Window** GUI.

Viewing HDF Image Objects

This procedure describes how to use the **EOSView tool** to view science Images in the HDF-EOS output file from a PGE.

The following is a list of tools, and or assumptions:

1. The SSIT Manager is running.
2. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools.
3. At least one object is an HDF image **(RIS8, RIS24, *i.e.* Browse data)**.

Viewing HDF Image Objects

1 From the SSIT Manager, select **Tools→ Product Examination → EOSView** from the menu.

The EOSView GUI will be displayed.

2 In the GUI labeled **EOSView - *MyOutputFile.hdf***, double click on an **Image object** listed for which data is to be inspected.

A GUI labeled EOSView - Image Display Window - ***MyImageObject*** will be displayed where ***MyImageObject*** is the name of the object selected.

3 Optional colorization. In the GUI labeled **EOSView - Image Display Window - *MyImageObject***, click on the **Palette** menu, then select one of the palettes listed: **Default**, **Greyscale**, **Antarctica**, **Rainbow**, or **World Colors**.

This selection may be repeated until the desired palette is chosen.

4 Optional zooming. In the GUI labeled **EOSView - Image Display Window - *MyImageObject***, click on the **Zooming** menu, then select and then select one of the resampling methods listed: **Bilinear Interpolation** or **Nearest Neighbor**. Then click on the **Zoom In** or **Zoom Out** buttons to apply the method.

5 Optional panning while zooming. In the GUI labeled **EOSView - Image Display Window - *MyImageObject***, click on the **Options** menu, then select **Pan Window**, a thumbnail representation of the entire Image will be displayed in the subwindow labeled **Pan Window**. The portion of the zoomed Image shown in the main window will be the portion indicated by the hollow rectangle on the thumbnail image. Use the mouse left button to click and drag the rectangle to a new location on the thumbnail image.

The panning option may be repeated as desired.

6 To end the session with colorization, zooming, or panning, in the GUI labeled **EOSView - Image Display Window - *MyImageObject***, click on the **File** menu and select **Close**.

7 Optional animation. In the GUI labeled **EOSView - *MyOutputFile.hdf***, click on the **Options** menu, then select **Animated images**.

A GUI labeled EOSView - Image Animation Window - ***MyOutputFile.hdf*** will be displayed.

Optionally, click on the **Options menu select Mode** to select how the animation is to be run. Choose **Stop at end, Continuous run, or Bounce**.

To end animation session, click on the File → **Close**.

Viewing HDF-EOS Grid Objects

This procedure describes how to use the EOSView tool to view science data in the HDF-EOS output file that are in HDF-EOS Grid format. These are generally the science data and not browse images.

The following is a list of tools, and or assumptions:

1. The SSIT Manager is running.
2. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools.
3. At least one object is an HDF-EOS Grid.

Viewing HDF-EOS Grid Objects

1 From the SSIT Manager, select **Tools→ Product Examination → EOSView** from the menu.

The EOSView GUI will be displayed.

2 In the GUI labeled **EOSView - *MyOutputFile.hdf***, double click on an Grid object listed for which data is to be inspected. A GUI labeled **EOSView - Grid Select** will be displayed.

Information on Grid Information, Projection Information, Dimensions, Attributes for the selected object can be displayed by clicking on the appropriate checkboxes.

3 In the GUI labeled **EOSView - Grid Select**, click on the **Data Fields** checkbox and then click on the **OK** button. Then double click on one of the data fields listed.

A GUI labeled EOSView - Grid - ***GridObjectName* - Start/Stride/Edge** will be displayed where ***GridObjectName*** will be replaced by the name of the Grid object selected in step 1.

4 To display the data in the form of a table of values, in the GUI labeled **EOSView - Grid - *GridObjectName* - Start/Stride/Edge**, click on the checkboxes for both **YDim** and **XDim** and then click on the **OK** button.

A GUI labeled ***MyDataField*** will be displayed where ***MyDataField*** will be replaced by the name of the data field selected in step 2.

5 To display the data field in image form, in the GUI labeled ***MyDataField***, click on the **File** menu and then select **Make Image**. A GUI labeled **EOSView - Swath/Grid Image** will appear,

6 **Optional colorization, zooming, panning while zooming** can be used to obtain your desired output.

7 To end the session with displaying Grid object, in the GUI labeled **EOSView - Swath/Grid**, click on the **File** menu and select **Close**. The **EOSView - Swath/Grid** GUI will disappear.

Viewing HDF-EOS Swath Objects

This procedure describes how to use the EOSView tool to view science data in the HDF-EOS output file that are in HDF-EOS Swath format. These are generally the science data and not browse images.

The following is a list of tools, and or assumptions:

1. The SSIT Manager is running.
2. The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools.
3. At least one object is an HDF-EOS Swath.

Viewing HDF-EOS Swath Objects

---

1 From the SSIT Manager, select **Tools→ Product Examination → EOSView** from the menu.

The EOSView GUI will be displayed.

2 In the GUI labeled **EOSView - *MyOutputFile.hdf***, double click on a Swath object listed for which data is to be inspected.

A GUI labeled **EOSView - Swath Select** will be displayed.

Information on Dimensions, Geolocation Mappings, Indexed Mappings, Geolocation Fields, Attributes for the selected Swath Object can be displayed by clicking on the corresponding checkboxes.

3 In the GUI labeled **EOSView - Swath Select**, click on the **Data Fields** checkbox and then click on the **OK** button. Then double click on one of the data fields listed.

A GUI labeled EOSView - Swath - ***SwathObjectName*** **- Start/Stride/Edge** will be displayed where ***SwathObjectName*** will be replaced by the name of the Swath object selected in step 1.

4 To display the data in the form of a table of values, in the GUI labeled **EOSView - Swath -** ***SwathObjectName*** **- Start/Stride/Edge**, click on the checkboxes for both **ScanLineTra** and **PixelsXtrac** and then click on the **OK** button.

5 To display the data field in image form, in the GUI labeled ***MyDataField***, click on the **File** menu and then select **Make Image**.

A GUI labeled **EOSView - Swath/Grid Image** will appear.

6 **Optional colorization, zooming, panning** while zooming features can be used in the GUI labeled **EOSView - Swath/Grid Image** to obtain your desired image.

7 To end the session with displaying Swath object, in the GUI labeled **EOSView - Swath/Grid**, click on the **File → Close**.

Viewing HDF SDS Objects

This procedure describes how to use the EOSView tool to view science data in the HDF-EOS output file that are in HDF SDS (standard HDF science data set) format. To view an HDF SDS object with the EOSView tool, execute the procedure steps that follow:

The following is a list of tools, and or assumptions:

The SSIT Manager is running.

The output file is HDF-EOS and has been created and populated with metadata using the SDP Toolkit metadata tools.

At least one object is an HDF-SDS.

Viewing HDF SDS Objects

1 From the SSIT Manager, select **Tools→ Product Examination → EOSView** from the menu.

The EOSView GUI will be displayed.

2 In the GUI labeled **EOSView - *MyOutputFile.hdf***, double click on a **SDS** object listed for which data is to be inspected.

A GUI labeled **EOSView - Multi-Dimension SDS** will be displayed.

A number of checkboxes will be displayed, one for each of the dimensions in the selected SDS (there will be at least two, an **X** and a **Y**).

3 In the GUI labeled **EOSView - Multi-Dimension SDS**, click on two of the dimension checkboxes and then click on the **Table** button. Then double click on one of the data fields listed.

A GUI labeled ***MySDS*** will be displayed where ***MySDS*** will be replaced by the name of the **SDS** object selected in step 1.

4 To display the data field in image form, in the GUI labeled ***MySDS***, click on the **File** menu and then select **Make Image**.

A GUI labeled EOSView - Image Display Window - *MySDS* will appear,

5 **Optional colorization, zooming, panning** while zooming can be used to obtain your desired output.

6 To end the session with displaying Swath object, in the GUI labeled **EOSView - Image Display Window - *MySDS***, select **File → Close** from the menu.

The EOSView - **Image Display Window - *MySDS*** GUI will disappear.

Viewing Product Data with the IDL Tool

The following procedures describe how to use the IDL (Interactive Data Language) COTS tool to inspect the data in the output file from a PGE. These procedures are geared toward binary and

ASCII formats, but can be extended to other formats supported by IDL including HDF, NetCDF, and PGE. Consult the IDL references for details on these other formats.

The major activities addressed here include creating an image display, saving an image display, creating a plot display, and saving a plot display.

The following is a list of tools, and or assumptions:

The SSIT Manager is running.

The output file is binary, ASCII, or one of the other IDL supported data formats.

IDL has been properly installed and is accessible to the user.

Viewing Product Data with the IDL Tool

1 From the SSIT Manager, select **Tools→ Product Examination → IDL** from the menu.

An xterm **(on the AIT Sun)** will be displayed within which the IDL command interpreter will be run.

2 Select the procedure depending upon the activity to perform.

3 To end the IDL session, close any display windows remaining, then at the IDL prompt type **quit**, then press the **Enter** key.

The IDL session will be closed.

Creating an Image Display Using IDL

The following procedure describes how to use the IDL Tool to create an image display.

The following is a list of tools, and or assumptions:

The SSIT Manager is running.

The PGE output file to be examined is of an IDL-supported type/format (if in doubt, consult the IDL Reference Guide

IDL has been properly installed and is accessible to the user.

For binary files, data is assumed to be 8-bit characters

Creating an Image Display Using the IDL Tool - Binary Data

1 From the SSIT Manager, select **Tools→ Product Examination → IDL** from the menu.

An xterm (on the AIT Sun) will be displayed within which the IDL command interpreter will be run.

2 At the IDL prompt, type ***OPENR,1,'MyBinaryFilename'***, press the **Enter** key.

The ***MyBinaryFilename*** is the full path name and file name of the binary data file of known dimensions to read in.

The single quotes **(‘)** must be included around the **path/file name**.

The 1 is the logical unit number.

3 At the IDL prompt, type ***MyImage=BYTARR(dim1, dim2)***, press the **Enter** key.

The ***MyImage*** is the name to be given to the image once created.

The ***dim1*** and ***dim2*** are the dimensions of the input data.

4 At the IDL prompt, type ***READU,1,MyImage***, press the **Enter** key.

5 At the IDL prompt, type ***TV,MyImage***, press the **Enter** key.

The image, ***MyImage,*** should then be displayed.

6 At the IDL prompt, type ***LOADCT,3***, press the **Enter** key.

This command loads color table number 3. Other color tables are available

7 At the IDL prompt, type ***CLOSE,1***, press the **Enter** key.

This closes logical unit 1.

Always close logical units or an error will result the next time an access is attempted.

Creating an Image Display Using the IDL Tool - ASCII Data

---

1 From the SSIT Manager, select **Tools→ Product Examination → IDL** from the menu.

An xterm (on the AIT Sun) will be displayed within which the IDL command interpreter will be run.

2 At the IDL prompt, type ***OPENR,1,(‘MyASCIIfilename’)***, then press the **Enter** key.

The **MyASCIIfilename** is the full path name and file name of the ASCII data file of known dimensions to read in.

The single quotes (‘) must be included around the **path/file name**.

The 1 is the logical unit number.

3 At the IDL prompt, type ***MyImage=BYTARR(dim1,dim2)***, then press the **Enter** key.

The ***MyImage*** is the name to be given to the image once created.

The ***dim1*** and ***dim2*** are the dimensions of the input data.

4 At the IDL prompt, type ***READF,1,MyImage***, then press the **Enter** key.

5 At the IDL prompt, type ***TV,MyImage***, then press the **Enter** key.

The image, MyImage, is displayed.

6 At the IDL prompt, type ***LOADCT,3***, then press the **Enter** key.

This command loads color table number 3. Other color tables are available; refer to the IDL Reference Guide for more details.

7 At the IDL prompt, type ***CLOSE,1***, then press the **Enter** key.

This closes logical unit 1.

Always close logical units or an error will result the next time an access is attempted.

Creating an Image Display Using the IDL Tool - PGM Data:

---

1 From the SSIT Manager, select **Tools→ Product Examination → IDL** from the menu.

An xterm (on the AIT Sun) will be displayed within which the IDL command interpreter will be run.

2 At the IDL prompt, type ***READ_PPM,"MyPGMfilename",MyImage,r,g,b***, then press the **Enter** key.

The **MyPGMfilename** is the full path name and file name of the PGM formatted data file.

The double quotes **("**) must be included around the **path/file name**.

The **MyImage** is the name to be given to the image created.

3 At the IDL prompt, type ***TVLCT,r,g,b***, then press the **Enter** key.

Note that **r,g,b color table syntax** is used for most formatted file types in **IDL.**

4 At the IDL prompt, type ***TV,MyImage***, then press the **Enter** key.

The image, **MyImage**, should then be displayed.

Saving an Image Display Using IDL

The next procedure describes how to save an image display (once created) to either a data file or a graphic file.

The following is a list of tools, and or assumptions:

The SSIT Manager is running.

IDL is running

The PGE output file to be examined is of an IDL-supported type/format (if in doubt, consult the IDL Reference Guide

For binary files, data is assumed to be 8-bit characters

The image display is to be saved in a binary (8-bit) or ASCII (comma-delimited characters) format.

Save an image display using IDL - Binary Data

1 From the SSIT Manager, select **Tools→ Product Examination → IDL** from the menu.

An xterm (on the AIT Sun) will be displayed within which the IDL command interpreter will be run.

2 At the IDL prompt, type ***OPENW,1,('MyBinaryFilename.bin'***), then press the **Enter** key.

The ***MyBinaryFilename.bin*** is the full path name and file name of the binary data file to write out.

The single quotes **('**) must be included around the **path/file name.**

The **1** is the logical unit number.

3 At the IDL prompt, type ***WRITEU,1,MyImage***, then press the **Enter** key.

The ***MyImage*** is the name of the image to save.

4 At the IDL prompt, type ***CLOSE,1***, then press the **Enter** key.

This closes logical unit 1.

Always close logical units or an error will result the next time an access is attempted.

Save an image display using IDL - ASCII Data

1 From the SSIT Manager, select **Tools→ Product Examination → IDL** from the menu.

An xterm (on the AIT Sun) will be displayed within which the IDL command interpreter will be run.

2 At the IDL prompt, type ***OPENW,1,('MyASCIIfilename.asc')***, then press the **Enter** key.

The ***MyASCIIfilename.asc*** is the full path name and file name of the binary data file to write out.

The single quotes **('**) must be included around the **path/file name.**

The **1** is the logical unit number.

3 At the IDL prompt, type ***PRINTF,1,MyImage***, then press the **Enter** key.

The **MyImage** is the name of the image to save.

4 At the IDL prompt, type **CLOSE,1**, then press the **Enter** key.

This closes logical unit 1.

Always close logical units or an error will result the next time an access is attempted.

Save an image display using JPEG Data

1 At the IDL prompt, type **WRITE_JPEG,”*MyJPEGfilename.jpg*”,*MyImage*** and then press the **Enter** key.

The ***MyJPEGfilename.jpg*** is the full path name and file name of the **JPEG** data file to write out.

Creating a Plot Display Using IDL

The procedures for creating a plot display are clearly described in the **IDL** manuals; some exceptions are clarified below.

Setting axis limits for a plot:

1 At the IDL prompt, type **SURFACE,MyPlot,AX=70,AZ=70,xrange=[0,20],yrange=[0,20]zrange=[0,30]**, and press **Enter**.

The ***MyPlot*** is the IDL session variable (to which you have assigned some math function, program output, image, etc.).

AX sets the displayed rotation about the **X axis.**

AZ sets the displayed rotation about the **Z axis**.

The values of ***xrange*** set the displayed portion of the **X axis.**

The values of ***yrange*** set the displayed portion of the **Y axis.**

The values of ***zrange*** set the displayed portion of the **Z axis.**

The plot will then be displayed to the screen.

Setting axis titles for a plot:

1 At the IDL prompt, type **SURFACE,MyPlot,AX=70,AZ=70,xtitle=’this is X’, ytitle=’this is Y’,ztitle=’this is Z’**, and press **Enter**.

The ***MyPlot*** is the **IDL** session variable (to which you have assigned some math function, program output, image, etc.).

The value of ***xtitle*** sets the displayed title of the **X axis.**

The value of ***ytitle*** sets the displayed title of the **Y axis.**

The value of ***ztitle*** sets the displayed title of the **Z axis.**

The plot will then be displayed to the screen.

Saving a Plot Display Using IDL

Saving a displayed plot to a permanent file:

1 At the IDL prompt, type **MyPlotDisplay=SURFACE,MyPlot,AX=80,AZ=20**, and press **Enter**.

The ***MyPlotDisplay*** is session name for the displayed plot of ***MyPlot.***

The ***MyPlot*** is the **IDL** session variable (to which you have assigned some math function, program output, image, etc.).

2 At the IDL prompt, type **SAVE,MyPlotDisplay,4,'MyPlotOutput.ps'** and then press the **Enter** key.

The ***MyPlotDisplay*** is the session name of the plot display .

The ***MyPlotOutput.ps*** is the desired name for the saved file.

The SAVE option number 4 sets the output file type to PostScript (ps). There are other options, of course (consult the IDL manuals).

Raster Mapping Fundamentals

This procedure describes how to use the IDL Tool to perform basic raster mapping functions. These are spatial functions involving map projections, but do not include surface modeling (also called "2.5D") or two-dimensional spectral functions.

The following is a list of tools, and or assumptions:

The SSIT Manager is running.

IDL is running

Raster Mapping - Global Data Set Image

1 From the SSIT Manager, select **Tools→ Product Examination → IDL** from the menu.

An xterm (on the AIT Sun) will be displayed within which the IDL command interpreter will be run.

2 At the IDL prompt, type ***TV,MyImage***, then press the **Enter** key.

The ***MyImage*** is the image name of the global image data set.

The image, ***MyImage,*** should then be displayed.

3 At the IDL prompt, type ***MAP_SET,/ORTHOGRAPHIC***, then press the **Enter** key.

IDL also supports other map projections. Refer to IDL Reference Guide.

4 At the IDL prompt, type ***MyNewImage=MAP_IMAGE(MyImage,startx,starty,/BILIN)***, then press the **Enter** key.

The ***MyNewImage*** is the name to assign to the resulting image.

The ***MyImage*** is the name of the original global image data set.

5 At the IDL prompt, type ***TV,MyNewImage,startx,starty***, then press the **Enter** key.

The image ***MyNewImage*** should then be displayed.

6 Optional overlay Lat/Long. At the IDL prompt, type ***MAP_GRID***, then press the **Enter** key.

This overlays Lat/Long graticule onto ***MyNewImage*.**

7 Optional overlay world coastlines. At the IDL prompt, type ***MAP_CONTINENTS***, press the **Enter** key.

This overlays world coastlines onto MyNewImage.

For a sub-global data set image, one having geocentric-LLR coordinates defined for subintervals of longitude and latitude (e.g. from -88 to -77 degrees East Longitude and 23 to 32 degrees North Latitude.

The following is a list of tools, and or assumptions:

1. The SSIT Manager is running.
2. IDL is running

Raster Mapping - Sub-Global Data Set Image

---

1 From the SSIT Manager, select **Tools→ Product Examination → IDL** from the menu.

An xterm (on the AIT Sun) will be displayed within which the IDL command interpreter will be run.

2 At the IDL prompt, type ***TV,MyImage***, then press the **Enter** key.

The ***MyImage*** is the image name of the sub-global image data set.

The image, ***MyImage***, should then be displayed.

3 At the IDL prompt, type ***MAP_SET,/MERCATOR,LIMIT=[lat1,lon1,lat2,lon2]***, then press the **Enter** key.

The ***lat1, lon1, lat2***, and ***lon2*** specify the latitude and longitude intervals of the sub-global image data set.

4 At the IDL prompt, type ***MyNewImage=MAP_IMAGE(MyImage,startx,starty,/BILIN.LATMIN=lat1,LATMAX=lat2 ,LONMIN=lon1,LONMAX=lon2)***, then press the **Enter** key.

The ***MyNewImage*** is the name to assign to the resulting image.

The ***MyImage*** is the name of the original global image data set.

The ***lat1, lon1, lat2,*** and ***lon2*** specify the latitude and longitude intervals of the sub-global image data set.

5 At the IDL prompt, type ***TV,MyNewImage,startx,starty***, then press the **Enter** key.

The image ***MyNewImage*** should then be displayed.

6 Optional overlay Lat/Long. At the IDL prompt, type ***MAP_GRID***, then press the **Enter** key.

This overlays Lat/Long graticule onto ***MyNewImage*.**

7 Optional overlay world coastlines. At the IDL prompt, type ***MAP_CONTINENTS***, then press the **Enter** key.

This overlays world coastlines onto ***MyNewImage*.**

Miscellaneous

Setting Up Environment for Direct PDPS Database Access

The PDPS database contains many tables containing information about science software running in the production system. The population of some of this information is part of standard SSI&T procedures and no special environment set up is required for these procedures. It may be necessary, however, to gain direct access to the PDPS database from time to time and this procedure describes how to set up the required environment. Detailed procedures for tasks performed by the SSI&T operator are provided in the sections that follow.

Assumptions:

1. The user has been given read access (at a minimum) for the PDPS database.
2. The C shell or a derivative (*e.g.* T shell) is the current user shell.

To set up an environment for PDPS database access, execute the procedure steps that follow:

1 At a UNIX prompt on an AIT Sun or on the SPR SGI, type **vi $HOME/.cshrc**, press **Enter**

This brings up the file named **.cshrc** in the **vi editor.**

Any text editor may be used such as emacs. For example, **emacs $HOME/.cshrc** and then press the **Enter** key.

2 In the editor add the following lines if not already there:

setenv SYBASE /vendor/sybase

setenv SYBASE /vendor/sybase

setenv SYBROOT $SYBASE/sybooks

setenv EBTRC $SYBROOT/sun5m/.ebtrc

setenv DSQUERY *computer-server*

set path = ($path $SYBASE $SYBASE/bin $SYBROOT/sun5m/bin)

The ***computer-server*** is the name of the server at the local DAAC and should be known. If not known, ask your SA or DBA.

The above lines should be entered into the **.cshrc file**, one per line as shown.

3 Save the changes made to the **.cshrc file** and **exit** the editor.

The specifics depend upon which editor is being used. If using *vi*, the command sequence to enter is **:wq** and then press the **Enter** key.

For other editors, refer to that editor's documentation.

4 At the UNIX prompt on the AIT Sun, type **source $HOME/.cshrc** and then press the Enter key.

This will reinitialize the setting in the **.cshrc file.**

The environment set up for access to the PDPS database should now be complete.

Examine Application Log Files

Assumptions:

1. The particular application makes use of and produces log files.
2. The location of the application log file is known.
3. The name of the application producing the log file is known.

To examine the application log files, execute the procedure steps that follow:

---

1 At the UNIX prompt on an AIT Sun or on the SPR SGI, type **cd *LogFilesPathname*** and then press the **Enter** key.

The ***LogFilesPathname*** is the full path name to the location of a particular application log file. This is typically the directory from which the SSIT Manager or other tool is run.

For example, type **cd $HOME/ceres** and then press the **Enter** key.

2 At a UNIX prompt on an AIT Sun or on the SPR SGI, type **vi *ApplicationName*.log**, press **Enter**

This *ApplicationName*.log is the file name of the log file to examine.

Any text editor may be used such as ***emacs.*** For example, **emacs QAMonitor.log** and then press the **Enter** key.

3 When finished, exit the editor.

The specifics depend upon which editor is being used. If using ***vi***, the command sequence to enter is :**wq** and then press the **Enter** key.

For other editors, refer to that editor's documentation.

Examples of PGE and ESDT ODL Files for Each Instrument Team

This section is covered in the latest **Green Book 162-TD-001-005** and deals, in part, with the use of ODL files in SSI&T activities. Useful examples of ODL files are given. ODL Template files, from which additional examples were created are depicted. Then, examples of specific ODL files will be listed by instrument (ASTER, MISR or MODIS).

Template ODL Files

There are five Template ODL files listed therein. The specific or tailored ODL files listed were derived from these templates by appropriate editing and filling-in of values. The three ODL Template files listed reside, on the AIT Sun host, at **/usr/ecs/<mode>/CUSTOM/data/DPS** . They are

PGE_ODL.template

ESDT_ODL.template

ORBIT_ODL.template

PATHMAP_ODL.template

TILE_ODL.template

This page intentionally left blank.

Practical Exercise

Introduction

This exercise is designed to practice the key features of the SSIT lesson.

Equipment and Materials

One ECS planning workstation.

Sample Science Software.

Science Software Integration and Test -

Situation: A new PGE and associated files has been developed by the Instrument Team. This exercise will require the CM Administrator and the Science Data Coordinator to prepare the PGE for installation and operation on the Production Server.

1 Acquire the science software as a Delivered Algorithm Package (DAP) tar file and unpack. For this training exercise, copy the tar file from the instructor provided directory.

2 Import the source and code make file into ClearCase. This will require creation of a view and a VOB to place these files.

3 Edit the ESDT descriptor files, MCF, makefile, and all ".met" files.

4 Perform the initial setup of the SSIT Manager and verify that the SSIT Manager is prepared to use the SDP Toolkit, PCF files, and that all required environmental variables point to locations in the SDP Toolkit directory.

5 Verify that the code is in compliance with the ESDIS Standards. For this training exercise, compile the PGE code by using the compiler's default of ANSI Standard.

6 Invoke the Prohibited Function Checker on the source files.

7 Use the PCF checker to verify that the PCFs are syntactically correct, and contain all necessary information for the PGEs to run within the ECS DAAC production environment (if not, update the PCF with correct data and rerun the PCF Checker).

8 Compile the PGE and Link the with the SDP Toolkit.

9 Run the PGE in an SCF type environment and generate performance statistics.

10 Examine the metadata in output HDF files using EOSView.

11 Examine the PGE LogStatus, Log User, and LogReport.

12 Update the PDPS Database by performing the following:

13 Copy the edited ESDT ODL files into the configured PDPS area.

14 Run the PCF ODL Update program from the SSIT Manager and modify the output file.

15 Copy the PCF ODL files into the configured PDPS area.

16 Run the Science Metadata Update Program from the SSIT Manager.

17 Run the PGE Operational Metadata Program from the SSIT Manager.

18 Update the Data Server by performing the following:

19 Run the Insert Static Program from the SSIT Manager to insert the MCF file.

20 Run the Insert Test Dynamic Program from the SSIT Manager to insert the binary data granule for input into the PGE.

21 Make the tar file for the Science Software Executable Package.

22 Run the EXE TAR program inform the SSIT Manager to insert the executable package in the data server.

23 Submit a Production Request for the PGE.

24 Invoke the Planning Workbench to plan, schedule and activate the Production Requests.

25 Monitor production by using Autosys.

26 Invoke the QA Monitor to view the products under EOSView and Production History File.

27 Use DX Browser by invoking command db to look at the PDPS databases.

If there is a problem, activate DDTS and record the problem.

Slide Presentation

Slide Presentation Description

The following slide presentation represents the slides used by the instructor during the conduct of this lesson. This presentation is in two parts, SSI&T and ESDT training.

This page intentionally left blank.

Appendix A. Troubleshooting and General Investigation

This section is primarily meant to serve as a reference to SSIT personnel for use in diagnosing problems encountered during testing of PGEs in the ECS system. It is divided into subsections by category of failure. Some types of failures are common to many steps in the SSIT process, so their investigation follows a common path. Thus, the SSIT user should refer to the following sections on the basis of the underlying process (e.g., file insertion to the Science Data Server) rather than by the distinct stage in the SSIT procedure.

Description

ESDT versioning has been continued in this release, which means a DLL can be shared by more than one ESDT descriptor files.. In this release, the DLL file that will be used during installation is specified in the ESDT descriptor file. Therefore during ESDT installation, a user only need to provide ESDT descriptor file name instead of ESDT descriptor file and DLL file.

An Alternative Way to Install and Remove an ESDT

There is an alternative way to install an ESDT. Detailed procedures for tasks performed by the SSI&T operator are provided in the section follow.

Assumptions:

1. All the required Servers are up running properly.
2. Environment has been properly set up.

To install an ESDT, execute the steps that follow:

1 Open an **xterm window** and then **telnet** to **Science Data Server host**.

2 It is recommended that this procedure begin within a **new command shell on the Sun platform.**

3 At the UNIX prompt, type **cd /usr/Esc/<mode>/CUSTOM/utilities**, then press **Enter**.

4 At the UNIX prompt, type **setenv MODE *mode*** and then press the **Enter** key. Then type, **source EcCoEnvCsh** and then press the **Enter** key**.**

5 type **EcDsSdSrvGuiStart <mode>** and then press the **Enter** key.

6 The script will launch a MOTIF GUI for a user to provide ESDT information.

7 Select the **Add** button.

8 Another window will be pooped up.

9 Type the ESDT descriptor file name include full pathname under Descriptor Filename tab.

10 /usr/ecs/<mode>CUSTOM/data/ESS/DsESDTxxShortName.version.desc.

11 Click **OK** button to perform the execution.

12 To remove an ESDT, execute the steps that follow:

Open an xterm window and then telnet to Science Data Server host.

It is recommended that this procedure begin within a new command shell on the Sun platform.

13 At the UNIX prompt, type **cd /usr/Esc/<mode>/CUSTOM/utilities**, then press **Enter**.

At the UNIX prompt, type **setenv MODE *mode*** and then press the **Enter** key. Then type, **source EcCoEnvKsh** and then press the **Enter** key.

15 Type **EcDsSrRmesdt <mode> DescriptorFileName1, DescriptorFileName2**…

16 The script can be used to remove more than one ESDT in Science Data Server.

17 Open an xterm window and then telnet to **Advertising host.**

It is recommended that this procedure begin within a new command shell on the Sun platform.

18 At the UNIX prompt, type **cd /usr/Esc/<mode>/CUSTOM/utilities**, then press **Enter**.

At the UNIX prompt, type **setenv MODE *mode*** and then press the **Enter** key. Then type,

19 **source EcCoEnvCsh** and then press the **Enter** key.

20 **cd /usr/ecs/<mode>/CUSTOM/bin/IOS**

21 Type **ContributionDriver <mode>**

22 **<ShortName>**

23 **y**

24 Success is indicated by return of **"<"** prompt.

25 Open an xterm window and then telnet to **Advertising host**.

26 At the UNIX prompt, type **cd /usr/Esc/<mode>/CUSTOM/utilities**, then press **Enter**.

At the UNIX prompt, type **setenv MODE *mode*** and then press the **Enter** key. Then type,

27 **source EcCoEnvCsh** and then press the **Enter** key.

28 **cd /usr/ecs/<mode>/CUSTOM/dbms/DMS**

29 Type **DmDbCleanCollection.csh <ShortName> 1**

30 Then answer the prompts with:

31 Enter Server Name (DSQUERY): <**host name>_srvr**

32 Enter Database Name: **EcDmDictService_<mode>**

33 Enter User Name: **dms_role**

34 Enter Password: **welcome**

Handling an ESDT Installation Failure

During ESDT installation, there are two types of failure. One is general failure, which means no ESDT can be installed. This case is usually caused by improper operations, which means the Servers may not be brought up properly, or a login is expired etc. The other is particular ESDT installation failure, which means certain type of ESDT can not be installed, but other ESDTs may be able to installed. This case is relatively hard for a user to trace the problem. The problem is usually caused by particular attributes in the ESDT descriptor file, which means the ESDT

descriptor file is incompatible with Science Data Server Database table definitions and also some attributes in the ESDT descriptor file may violate the validation rule defined in the Data Dictionary. Therefore as a general rule, a user needs to identify the failure type first. The following sections are general guidance for a user to handling an ESDT installation failure.

Check the ESDT descriptor file and the DLL file: Inside Science Data Server configuration file, which is EcDsScienceDataServer.CFG (at /usr/ecs/<mode>/CUSTOM/cfg),
there are entries that indicate where to find ESDT descriptor file and DLL. Those entries are DSSDESCINPUTDIR and DSSDLLDIR. In the ESDT descriptor file, there is an attribute name that is DLLName. The DLL file should be resided at DSSDLLDIR. If the DLL file name indicated in ESDT descriptor file can not be found under the DSSDLLDIR, a failure of installation will occur.
Check the Science Data Server ALOG File: During ESDT installation the Science Data Server writes status messages to two log files, EcDsScienceDataServer.ALOG and EcDsScienceDataServerDebug.log. Entries to the ALOG file should include the ShortName of the ESDT. If the ShortName does not appear with a time stamp which reflects the time of the attempted installation, then the request of installation was not communicated to SDSRV. This might be the case if the SDSRV subsystem is not running, These log files can be inspected further for other errors, if the ESDT ShortName does appear.
Check the IOS Server ALOG file: During ESDT installation the Science Data Server notifies the Advertising Subsystem that the ESDT needs to be advertised for later use. The ALOG file, which is EcIoAdServer.ALOG, should include the ShortName of the ESDT. If the ShortName does not appear with a time stamp which reflects the time of the attempted installation, then Science Data Server ALOG file should indicate failure to advertise the ESDT. This could be caused by IOS Server is not up running or has a server connectivity problem.

Insert File Failure

Description
Files may be inserted into the Science Data Server in a variety of ways, depending on the type of file. In general, files must be accompanied by a descriptive metadata (.met) file in order for the Science Data Server to process them successfully. The types of files to be inserted include DAP (Delivered Algorithm Package) files, input and output science data granules, and PGE executable tar files (PGEEXE.tar). The Science Data Server provides a test driver to insert a file to SDSRV. The SSIT Manager GUI also provides various tools to insert a file, depending on its type. Also, one phase (De-staging) of the execution of a PGE by the system involves the insertion of output data products to the Science Data Server.
For inserting files using the SDSRV test driver or the SSIT Manager, the user first needs to prepare a metadata file to go with the file to insert into Science Data Server. Output file insertions during de-staging use metadata files generated by the PGE.
Files that are to be inserted to the Science Data Server using the SSIT Manager, in general, must be made known to the PDPS database in advance. This includes dynamic and static data files, and PGE executable tar file. This prerequisite is fulfilled by registering a PGE with the PDPS database. Once a data type has been made known to the PDPS database via PGE registration, files of that type may be inserted to the Science Data Server regardless of the PGE for which

they will serve as input or output. The PGEEXE.tar file can only successfully insert following registration of the particular PGE.

Handling an Insert Failure

If the Science Data Server returns a message indicating that the insertion has failed, or in the case of a De-staging failure, a number of things can be done to diagnose the problem causing the failure.

Check for ESDT in SDSRV Database: Check the Science Data Server for the ESDT corresponding to the file type to be inserted. This is done in either of two ways. The first is to enter a few SQL commands on the UNIX command line. The second utilizes a database viewer GUI. With either method, the aim is to verify that the ESDT required is listed in the Science Data Server data base in a table called DsMdCollections.

The table below lists the steps required to view the SDSRV data base using SQL commands.

The table below list the steps required to view the SDSRV data base using a database viewer.

For both methods, the aim is to locate the ESDT corresponding to the file to be installed. In the DsMdCollections table, the ESDT is located by the shortname. If the shortname is not listed, then the ESDT must be inserted to the Science Data Server before the file insertion could succeed..

Check for ESDT in the Advertising Database: When an ESDT is installed into the Science Data Server data base, the system also makes entries in the Advertising (IOS) database. The number and types of entries depends on the contents of the ESDT descriptor file. File insertion failures may also be caused by missing or incomplete IOS database entries for the ESDT. Therefore, it is useful to check IOS to make sure the ESDT corresponding to the file type to be inserted has been properly advertised. This is done by checking the advertising database, IoAdAdvService_*mode*, in a table called IoAdAdvMaster, for the shortname in question. This table, for each ESDT shortname, should show several entries, the number depending on the descriptor file contents. An example of such a listing is given below.

Sample Listing of ESDT Entries in Advertising Database

IoAdAdvMaster
MOD03.001
MOD03.001:ACQUIRE
MOD03.001:INSERT
MOD03.001:UPDATEMETADATA
MOD03.001:BROWSE
MOD03.001:GETQUERYABLEPARAMETERS
MOD03.001:INSPECT
MOD03.001:INSPECTCL
MOD03.001:DELETE
Subscribable Event:ID:##: MOD03.001:DELETE
Subscribable Event:ID:##: MOD03.001:INSERT
Subscribable Event:ID:##: MOD03.001:UPDATEMETADATA

The table below lists the steps required to view the Advertising data base using SQL commands.

Viewing the Advertising Database Using SQL Commands - Quick-Step Procedures

The table below lists the steps required to view the Advertising data base using a database viewer.
For both methods, the aim is to list all entries corresponding to the ESDT of the file to be installed. In the IoAdAdvMaster table, the ESDT is located by the title, which includes the shortname. If the shortname is not listed, then the ESDT must be inserted to the Science Data Server before the file insertion could succeed). If the listing does not include all required entries, then the ESDT must be removed from the Advertising , Science Data Server, and Data Dictionary databases describes ESDT removal from Science Data Server, Advertising and Data Dictionary while reinstallation is covered in Adding ESDTs to the System.

Check SDSRV ALOG File: During any operation involving the Science Data Server, useful information reflecting SDSRV activities is written to two log files. These are EcDsScienceDataServer.ALOG and EcDsScienceDataServerDebug.log. Entries to the ALOG file should include the ShortName of the file data type. Timestamps, which appear throughout the logs files, should be checked to make sure any entries found for a shortname correspond to the time of the attempted insertion. If the ShortName does not appear, then the file insertion request was not communicated to SDSRV. This might be the case if the SDSRV subsystem is not running.

Viewing the EcDsScienceDataServer.ALOG file - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	login to AIT computer which supports the SDSRV subsystem using a generic SSIT account.	
2	cd /usr/ecs/mode/CUSTOM/logs	press Enter
3	vi EcDsScienceDataServer.ALOG	press Enter

If the shortname does appear in the ALOG file, then the ALOG may be investigated further to determine whether the metadata has been successfully validated. Successful metadata validation is indicated when “End Metadata Validation. (Metadata is valid).” appears in the ALOG file. If the metadata is not valid, then the metadata validation section of the ALOG can be scanned to find what metadata errors have been identified by SDSRV.

Verify that the Servers are Running: File insertion requires not only that SDSRV be running, but also the IOS and Archive servers. This may be done in either of two ways. In the first, each subsystem host machine is checked for the status of its resident subsystems. In the second, the ECSAssist GUI is used to view the operation of all subsystems.

The short procedure below outlines how to check that a subsystem is running, using a command-line technique. This method requires that the user know the names of the subsystem components.

Checking That Subsystems are Running Using a Command Line Approach - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	login to AIT computer which supports the subsystem to check.	
2	/usr/bin/ps -ef | grep *mode*	press Enter

Acquire Failure

Description

Files are acquired from the Science Data Server either through the SSIT Manager, DAP acquire tool (it can be used for any type of file acquire), or using a test driver from SDSRV. Additionally, files are acquired by the system during the setup and execution of a PGE. Failure of an acquire is similar to insertion failure, and the methods to diagnose and resolve the failure also resemble those for insertions.

Handling an Acquire failure:

Diagnosing an acquire failure involves inspecting various system log files and checking in directories involved with the process.

Check Science Data Server Log Files: The EcDsScienceDataServer.ALOG file should contain entries regarding the acquire activity and identify the file to be acquired by its ShortName. If the ShortName does not appear in the ALOG file, with a timestamp corresponding to the time of the attempted acquire, then SDSRV may not be running, or may not be communicating with other servers. If the ALOG file does contain entries for that ShortName, and indicates that two files (the file and its associated metadata file) are being distributed, the message in the ALOG file looks like following:

Msg: File 1 to be distributed: :SC:MOD03.001:1369:1.HDF-EOS
Priority: 0 Time : 07/29/98 12:35:42
PID : 24279:MsgLink :1684108385 meaningfulname :DsSrWorkingCollectionDistributeO
neDistributFile
Msg: File 2 to be distributed: SCMOD03.0011369.met

then SDSRV has completed its role in the acquire. Therefore acquire script usually will indicate success.

Viewing the EcDsScienceDataServer.ALOG file - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	login to AIT computer which supports the SDSRV subsystem using a generic SSIT account.	
2	cd /usr/ecs/mode/CUSTOM/logs	press Enter
3	vi EcDsScienceDataServer.ALOG	press Enter

If the ALOG contains the ShortName, and also contains an error showing that the data file time stamp does not match the time stamp required by the acquire, then the data file needs to be removed from the Science Data Server and reinserted. This is usually done using a script called DsDbCleanGranules, however, in Drop 4.p.1, this script does not work. Instead, it is necessary to remove the ESDT from Science Data Server, then reinstall.

Check the Archive Server ALOG File: Acquire success from the Science Data Server is only part of the acquire process. Since any file entered into SDSRV is stored in the archive, the Archive Server must be involved during an acquire. Thus, it may be useful to inspect the Archive Server ALOG file (EcDsStArchiveServer.ALOG) to check for error messages associated with the ShortName of the file type.

Viewing the EcDsStArchiveServer.ALOG file - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	login to AIT computer which supports the Archive Server subsystem using a generic SSIT account.	
2	cd /usr/ecs/*mode*/CUSTOM/logs	press Enter
3	vi EcDsStArchiveServer.ALOG	press Enter

Check Staging Disk: During an acquire, files are copied to a staging area as an intermediate step before distributing them to their destination. As part of diagnosing an acquire failure it is useful to check the staging area to ascertain whether the files have completed part of their journey. Both the file and a subdirectory containing metadata information should be written to the staging area.

Viewing the Staging Area - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	login to AIT computer which supports the Archive Server subsystem using a generic SSIT account.	
2	cd /usr/ecs/*mode*/CUSTOM/drp/*archivehost*/data/staging/user#	press Enter
3	ls -lrt	press Enter

Check Staging Server ALOG: If the failure occurs in copying the files to the staging area, then the Staging log files (EcDsStStagingDiskServer.ALOG or EcDsStStagingMonitorServer.ALOG) may reveal the cause.

Viewing the EcDsStagingServer.ALOG file - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	login to AIT computer which supports the Archive Server subsystem using a generic SSIT account.	
2	cd /usr/ecs/*mode*/CUSTOM/logs	press Enter
3	vi EcDsStStagingDiskServer.ALOG or EcDsStStagingMonitorServer.ALOG	press Enter

Check the Space Available in the Staging Area: Failure can also be caused by a lack of space in the staging area.

Checking the Space Available in the Staging Area - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	login to AIT computer which supports the Archive Server subsystem using a generic SSIT account.	
2	cd /usr/ecs/*mode*/CUSTOM/drp/*archivehost*/data/staging/user#	press Enter
3	df -k **.**	press Enter

Failure During DPR Generation

Description

The creation of a Data Processing Request is an essential part of the SSIT process. There are many reasons for DPR creation failure to occur. During DPR generation, the DPR generation executable will turn subscriptionFlag from zero to non-zero, which needs subscription Server up running and the executable will also query Science Data Server for input granules. The cause of failure to generate a DPR could come from following errors:

1) Incorrect information in PGE_ODL and ESDT ODL files, which generally references to the PGE registration incorrectly.
2) Not all the required Servers up running properly, which generally reference to the System problem.
3) ESDTs required for the PGE were not properly installed.
4) Database queries failure, which generally reference to Database errors.

Handling DPR Generation Failures:

To find out why a DPR generation fails, a user needs to look for the Production Request Editor ALOG file, which is EcPlPREditor.ALOG resided at /usr/ecs/<mode>/CUSTOM/logs of PLS host. The ALOG file needs to be inspected for evidence of the source of a failure. In addition, that ALOG may indicate that the ALOG files for other subsystems, such as SDSRV or IOS, may contain entries describing the errors. Another useful resource for troubleshooting the failure is to look for PDPS database, all the PGE registration information are kept in difference tables. Inside the table PlDataTypeMaster, there is a column called subscriptionFlag, during a DPR generation, the DPR executable will turn the subscriptionFlag for all the ESDTs needed for the PGE from zero to non-zero, if the Flag for the dataTypeId (corresponding to ESDT) did not turn to non-zero, that indicates subscription trouble. On the other hand, inside the table PlDataTypeMaster, there is a column called dataServUrString, during a DPR generation, the DPR executable will turn the dataServUrString for all the ESDTs needed for the PGE from NULL to UR value for Science Data Server. If the PGE contains static files, the UR for the static files have to been included in PlDataGranuleShort table before a DPR can be successfully generated. For dynamic granules, the UR values will become available in the PlDataGranuleShort table if the granules are available during the DPR generation, if the dynamic granules are not available during a DPR generation, it will not cause a failure to generate a DPR.

Failure Scheduling a DPR

Description

Problems scheduling a PGE for execution in the system occur when a DPR, scheduled through the Planning Workbench, does not get passed to Autosys.

Handling a DPR Scheduling failure

There is ALOG file called EcPlWb.ALOG resided at /usr/ecs/<mode>/CUSTOM/utitilies of PLS host, which should be inspected first.

Check the PDPS Data Base: PlDataProcessingRequest: During scheduling, the PDPS data base is updated to reflect a change in the state of the DPR. In the PDPS data base the PlDataProcessingRequest table will show a value of "NULL" in the completionState field if the DPR did not get passed to Autosys.
The table below list the steps required to view the PDPS data base using a database viewer.

Viewing the PDPS Data Base Using the Data Base Viewer GUI - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	login to AIT computer which supports PDPS using a generic SSIT account.	
2	(if necessary) setenv DISPLAY *machinename*:0.0	press Enter
3	/home/opscm/dbr/dbrowser-syb	press Enter
4	enter into GUI login*: PDPSservername, PDPSusername, PDPSpassword*	
5	select pdps_*mode*	
6	select "sample data" under "view" menu	
7	select PlDataProcessingRequest	

The table below lists the steps required to view the PDPS data base using SQL commands.

Viewing the PDPS Database Using SQL Commands - Quick-Step Procedures

Step	What to Enter or Select	Action to Take
1	login to AIT computer which supports PDPS using a generic SSIT account.	
2	isql -Updpsusername -Ppassword -Spdpsserver	press Enter
3	use pdps_*mode*	press Enter
4	go	press Enter
5	select * from PlDataProcessingRequest where dprId = "*dprID*"	press Enter
6	go	press Enter

Check the PDPS Database: PlDataGranuleShort: Input data granules which are ready for use by a PGE running in the system will have entries with full URLs in the PDPS database table PlDataGranuleShort, under universalReference. Standalone PGE, those not running as part of a PGE chain, need to have the full URL entered. If the PGE is running as part of a PGE chain, then the input granules are products of preceding PGEs in the chain. If, in the ESDT odl files prepared for those input granules, the dynamic flag is set to "external" instead of "internal", then the DPR will go into Autosys as soon as the input granules become available. If the dynamic

flag is set as “internal” then the DPR should go into Autosys regardless of the availability of the input granules. PGE execution will commence, and the color of the display in JobScape within Autosys will change, as soon as the input granules are available.
Inspection of the PDPS database follows the procedures outlined in a above, with PlDataGranuleShort as the table, and universalReference as the field. The individual granules can be identified by their data type (dataTypeId).

Failures During Execution

Description
PGEs scheduled for execution in the system follow seven stages of processing. Each has its own types and causes of failure.

Handling a Failures During Execution

Resource Allocation: The first stage of PGE processing in Autosys is Resource Allocation. If this fails, the ALOG file of the Data Processing host can be checked to see whether the PGEEXE.tar file was successfully acquired. If there is an acquire failure, further evaluation proceeds as outlined in Acquire Failure, above.
To inspect the Data Processing ALOG file, execute the following steps:
1 Login to the AIT computer which supports Data Processing.
2 At the UNIX prompt, **type cd /usr/ecs/*mode*/CUSTOM/logs**, then press **Enter**.
3 At the UNIX prompt, type **vi *DPR#. ALOG or DPR#A.ALOG***, then press **Enter.**
Staging: The Staging step in processing involves acquiring files from SDSRV. Thus, it should be handled as in Acquire Failure, above.
To inspect the Data Processing ALOG file, execute the following steps:
1 Login to the AIT computer which supports Data Processing.
2 At the UNIX prompt, **type cd /usr/ecs/*mode*/CUSTOM/logs**, then press **Enter**.
At the UNIX prompt, type **vi *DPR#. ALOG or DPR#S.ALOG***, then press **Enter.**
Preprocessing: Preprocessing rarely completely fails. It may not generate the system PCF file correctly. Placing a “hold” on the PGE execution stage (through Autosys, in the Job Scape GUI, the Job Console button brings up the Job Console - PGE processing stages can be put on hold using the “On Hold” button.). While execution is on hold, the system PCF can be inspected to see whether it matches expectations.
To inspect the Data Processing ALOG file, execute the following steps:
1 Login to the AIT computer which supports Data Processing.
2 At the UNIX prompt, **type cd /usr/ecs/*mode*/CUSTOM/logs**, then press **Enter**.
At the UNIX prompt, type **vi *DPR#. ALOG or DPR#P.ALOG***, then press **Enter.**
To view the system PCF, execute the following steps:
1 Login to the AIT computer which supports Data Processing.
2 At the UNIX prompt, **type cd /usr/ecs/*mode*/CUSTOM/pdps/*hostname*/data/DpPrRm/*hostname*_disk/*pgeId*/*dprId_hostname***, then press **Enter**.
3 At the UNIX prompt, type **vi pgeId.Pcf**, then press **Enter.**
PGE Execution: Failures during PGE execution can be investigated using the Toolkit LogStatus file, the system PCF file, and by running the PGE from the command line using the

environment set in the PGE profile file and PGS_PC_INFO_FILE points to the system *.Pcf. All of these are found in the runtime directory.
To inspect these files, execute the following steps:

1 Login to the AIT computer which supports Data Processing.
2 At the UNIX prompt, **type cd /usr/ecs/*mode*/CUSTOM/pdps/*hostname*/data/DpPrRm/*hostname*_disk/*pgeId*/*dprId_hostname***, then press **Enter**.
3 At the UNIX prompt, type **vi pgeId.Pcf** or **pgeId.TkStatus**, then press **Enter.**

Postprocessing: Postprocessing does not often fail, but it may show as a failure in Autosys if the Execution stage has failed.

To inspect the Data Processing ALOG file, execute the following steps:

1 Login to the AIT computer which supports Data Processing.
2 At the UNIX prompt, **type cd /usr/ecs/*mode*/CUSTOM/logs**, then press **Enter**.
3 At the UNIX prompt, type **vi *DPR#. ALOG or DPR#p.ALOG***, then press **Enter.**

Destaging: Destaging involves the insertion of output data granules into SDSRV. Thus, section, Insertions of Files to the Science Data Server, should be consulted for this case.

To inspect the Data Processing ALOG file, execute the following steps:

1 Login to the AIT computer which supports Data Processing.
2 At the UNIX prompt, **type cd /usr/ecs/*mode*/CUSTOM/logs**, then press **Enter**.

At the UNIX prompt, type **vi *DPR#. ALOG or DPR#I.ALOG***, then press **Enter.**

Deallocation: Rarely are there cases of failure in Deallocation.

To inspect the Data Processing ALOG file, execute the following steps:

1 Login to the AIT computer which supports Data Processing.
2 At the UNIX prompt, **type cd /usr/ecs/*mode*/CUSTOM/logs**, then press **Enter**.

At the UNIX prompt, type **vi *DPR#. ALOG or DPR#D.ALOG***, then press **Enter.**

PDPS Troubleshooting - The PGE Job has Failed

- The PGE Job has failed, but the DPR has not gone into "Failed-PGE" processing
- The Post-Execute Job has failed
- The PGE Job and Post-Execute Job have both failed, but the DPR has not gone into "Failed-PGE" processing
- The PGE Job has failed and the DPR has gone into "Failed-PGE" processing

The PGE Job has failed

This condition is indicated when the PGE job only is red in **AutoSys**. This is hard to do, because the **AutoSys** job definition for this job says to allow any exit code to indicate success. This is because we want the next job, the post-execute job, to continue
even if this job fails. This job will "succeed" even if the PGE Wrapper job, **EcDpPrRunPGE,** doesn't exist. This job can fail if **AutoSys** cannot see the machine.

The Post-Execute Job has failed

This condition is indicated when the Post-Execute Job only is red in **AutoSys**. This happens when the PGE job never ran or if for some other reason (such as a mount point problem) the Execution Manager job cannot read the log file created by **EcDpPrRunPGE.**
Check that **/usr/ecs/B302TS1/CUSTOM/bin/DPS/EcDpPrRunPGE** and **EcDpPrRusage** exist on the science processor and that they are not links.
Check that **/usr/ecs/DEV04/CUSTOM/data/DPS** on the science processor is mounted or linked to **/usr/ecs/DEV04/CUSTOM/data/DPS** on the queuing server machine.

The PGE Job and Post-Execute Job have both failed

This condition is indicated when both the PGE and Post-Execute Jobs are red in **AutoSys**, but no other jobs are red. This indicates that the Post-Execute job has read the log file created by **EcDpPrRunPGE** in the runtime directory and has found an exit status not equal to 0. However, it failed to de-stage the failed **pge tar file**.

The PGE Job has failed and the DPR has gone into "Failed-PGE"

processing

This condition is indicated when the entire job box has turned red along with post-execute, de-staging and de-allocation jobs. A Failed PGE Tar File has been created and archived.
A PGE may fail for many reasons. Some of the possible causes are documented here:

- The PGE is the wrong architecture.

This happens when the PGE was miss-defined as **New32, Old32** or **64** from the SSIT Operational Metadata GUI. The PGE will core dump because of this problem. To fix this you need to go back to **the SSIT Operational Metadata GUI** and enter the correct architecture, then delete any **DPRs** created for that **PGE** and recreate them.

- One of the expected inputs for the PGE is missing

The first reason for this is that an expected input of the PGE is NOT defined in the **PGE ODL**. Check the error messages for something about a missing Logical Id and then check the **PGE ODL** for the expected **Logical Id.**
This can also happen when a miscommunication causes Subscription Manager to release a PGE despite it missing one (or more inputs). To find out if this is the case, verify that all inputs to the DPR have their availability flag set to 1 in PlDataGranuleShort and the corresponding entries in **PlDprData** have the accepted field set to 1.

- The leap seconds file is incorrect.

This page intentionally left blank.

PDPS Troubleshooting - A single DPS job has failed or is hanging

The entire Job Box is hung

A DPS Allocation job is hanging

A DPS Allocation job has failed

A DPS Staging job is hanging
A DPS Staging job has failed
A DPS Preprocessing job is hanging
A DPS Preprocessing job has failed
A DPS PGE job is hanging
A DPS PGE job has failed
A DPS Post-processing job has failed
A DPS De-staging job is hanging
A DPS De-staging job has failed
A DPS De-allocation job has failed
All known problems are corrected and the re-start of the job fails again

Non-PDPS servers are down

Back to the PDPS Troubleshooting home page

The entire Job Box is Hung

This condition is determined by noting that the entire Job Box (all 8 job steps) are the same color, and it is either the one indicated for "Inactive" jobs or the one for "On Hold" jobs. Check the legend to the left on the Jobscape display for the job box color meanings.

- The AutoSys Event server or one of the AutoSys clients could be down. See
- Attempt to re-start the Job Box by selecting the top of the Job Box.

A DPS Allocation job is hanging

This condition is determined by noting that the Allocation job has turned green to indicate that it is running, but that it never turns red (failed) or blue (success). If you "tail" the DPR .err file (eg: "tail -f **/usr/ecs/OPS/CUSTOM/logs/MODPGE08#s28035000OPS.err"**) you see that nothing is happening, or that the job is in a retry loop.

The Science Data Server (SDSRV) may be waiting for a request to Data Distribution (DDIST) to distribute the PGE tar file, but it can't because Storage Management (STMGT) is down. Go to where the DDIST GUI is running. Refresh the GUI. Check to see if the requestor source is EcDpPrEM and that the state is "Suspended with Errors". If this is the case, then you will have to bounce STMGT. After this is done, select the request on the DDIST GUI and click on "resume".

- The Science Data Server (SDSRV) may be waiting for a request to Data Distribution (DDIST) to distribute the PGE tar file, but it can't because Storage Management can't FTP the file to the data directory on the science processor disk. Go to where the DDIST GUI is running. Refresh the GUI. Check to see if the requestor source is EcDpPrEM and that the state is "Suspended with Errors". Does the target directory exist? Can you FTP a file to the directory on the science processor? If the answer to these questions is no, then fix the problem, and resume the request.

- If you observe that the Allocation Job is in a retry loop, then the SDSRV may be down. See Non-PDPS servers are down. Note that the first retry is designed to fail, because the software is retrieving server-side information to refresh the client-side at this point.

- The request may be waiting on the archive to stage the file. If there are several other requests in progress, the PGE acquire may have to wait until one or more of them completes. Check the state in the DDIST GUI - if it is in "staging" state, then the request should eventually complete.

A DPS Allocation job has failed

This condition is determined by noting that the Allocation job has turned red.

- Look at the .ALOG file (in **/usr/ecs/{MODE}/CUSTOM/logs**). If it is there, then look for the following:
  A message of "Error: unable to update Machine in Autosys" means that DPS is unable to access the AutoSys database. The auto.profile **in /usr/ecs/MODE/CUSTOM/bin/DPS** has the wrong settings for AUTOSYS and AUTOUSER parameters. Although they may differ from DAAC to DAAC, the expected values are:

  AUTOSYS = /usr/ecs/MODE/COTS/autotreeb/autosys

  AUTOUSER = /usr/ecs/MODE/COTS/autotreeb/autouser

To fix the problem, you either need to run the AutoSys Mkcfg again or go into the auto.profile file and change the values by hand.

- A message of "Unable to determine type of UR" means that the PGE tar file has not been inserted. To verify this is the problem check the PlResourceRequirement table in the PDPS database. There should be a non-null entry for the field exeTarUR. If that field is null, you need to go back to the SSIT procedure and insert the EXE Tar File. Then you should be able to re-start the job and watch it complete successfully.

If the .ALOG file is NOT present. Then do the following:

Bring up the Autosys Ops Console and select the Allocation Job that has failed.

- Check the return code. A value of 122 means that owner of the job DOES NOT HAVE WRITE PERMISSION to the log files directory. You need to find out the user account that was used to bring up Autosys and verify that it is correct and should have write permission to the logs directory.

A DPS Staging job is hanging

See A DPS Allocation job is hanging

A DPS Staging job has failed

This condition is determined by noting that the Staging job has turned red. Look at the .ALOG file (**in /usr/ecs/{MODE}/CUSTOM/logs**) for the DPRID of the job that has failed.

A message of "ESDT Acquire Failed for UR...." means that SDSRV had trouble processing one of the acquire requests. In this case re-starting the job should allow the acquire to succeed.

A DPS PreProcess job has failed

This condition is determined by noting that the Pre-Process job has turned red.
Look at the .ALOG file (in **/usr/ecs/{MODE}/CUSTOM/logs**) for the DPRID of the job that has failed. If it is there, then look for the following:

- A message of "NOFREECPUS" means that all of the Science Processor CPUs are busy and the PreProcess job went through its maximum number of retries to find an available CPU. You can just start the job again and it will work its way through its retries until a CPU is available.

Possible reasons for a job to run out of CPU resources:

- PGEs are taking longer to run than expected. DPS plans for execution times specified during SSIT, and if those times are exceeded by a large margin (by an executing PGE) it is possible that a PGE that is "ready to run" will be CPU starved.

A DPS PGE job is hung

This condition is determined by noting that the Execution job has turned orange or oscillates between orange and green.

- The AutoSys client is most likely down. See Checking the Status of AutoSys for how to verify AutoSys is up and happy.

A DPS PGE job has failed

This condition is determined by noting that the Execution job has turned red or the entire job box has turned red (failedPGE scenario).

See Troubleshooting - The PGE Job has Failed.

A DPS De-staging job has failed

The destaging job icon on the JobScape GUI will have turned red. Look at the .err log file.

- Typically, you will see a message such as "Error archiving metadata into catalog". You may also see some warning messages in the returned GlParameter list. You can disregard the warnings. If the problem occurred for an existing ESDT which has previously worked within the past day or two, then most likely STMGT is the culprit. Have someone from STMGT look at their log files, paying particular attention to changes/defects in their stored procedures.
- If you see the "Error archiving metadata into catalog" message and the ESDT is knew or has recently been installed, then look at the .MCF file in the runtime directory. Get somebody from SDSRV to help you compare the values of the mandatory parameters in the metadata file with "valids" from the SDSRV database.
- "Error archiving metadata into catalog" may also be associated with a SDSRV temporary directory getting filled up.

A message that indicates "Error archiving files" means that SDSRV is having trouble getting Storage Management to place the file(s) in the archive. Contact a Storage Management person:

to Verify that the Archive (AMASS) is up and functional.

Check through their logs to see why the request for archiving failed.

It is possible that the mount point between the science processor and the Storage Management machine has been lost. Check to make sure that the file that is being de-staged can be seen on the Storage Management machine. The typical path for the mount point is: Name**}**
/usr/ecs/{MODE}/CUSTOM/pdps/{science processor name}/data/DpPrRm/{Disk

- A message that indicates "Error modifying file usage" means that the numberOfUsage column in DpPrFile for a particular file is at 0 and the software is trying to decrement it. This column is an increment/decrement counter and is not normally decremented more times than it is incremented when under software control. However, if someone manually changes the database then the value may get out of sync and need to be manually reset to 1.
- If you see science data files in the disk partition, but no metadata files, then DDIST/STMGT is okay and SDSRV is not okay. Otherwise, suspect STMGT.

When the problem is corrected, re-start the job from AutoSys.

All known problems are corrected and the re-start fails again

The retry information in DpPrRpcID may now be out of sync between one or more servers. Find the appropriate entry in the table by inspecting the readableTag column and remove the entry before trying to re-start the job again.

Non-PDPS servers are down

Always verify that the Science Data Server, Storage Management Servers and Data Distribution Servers are up.

1. Bring up **Monitor GUI WHAZZUP**
2. Select the correct mode
3. Click on the "monitor" button
4. Observe that the status for the following servers is "Listening":
   EcDmDictServer
   EcDpPrDeletion
   EcDpPrJobMgmt
   EcDsDistributionServer
   EcDsScienceDataServer (all instances)
   EcDsStArchiveServer
   EcDsStFtpDisServer
   EcDsStPullMonitorServer
   EcDsStStagingDiskServer
   EcDsStStagingMontitorServer
   EcIoAdServer
   EcPlSubMgr
   EcSbEventServer
   EcSbSubServer
5. If any of these servers are down, contact the MSS operator to bring them up.

PDPS Troubleshooting - Job Activation Fails from the

Planning Workbench

Error reported is "DPR Validation Failed"

1. Check to make sure that Performance data has been entered for the PGE
   1. Use the database browser or isql to get access to the PDPS database.
   2. Look at the entries for PlPerformance.
   3. For the PGE(s) that are schedule, verify there is a non-zero value for the entries in this table.
   4. If entries are 0, then run the SSIT Operational Metadata GUI to enter correct performance values.

5.Delete the DPRs and then re-create them. Activation will succeed on the next attempt.

PDPS Troubleshooting - Input Data Problems

General description of the problem:

The Production Request fails due to too many granules

The Staging Job has failed due to too many granules

A Failure Due to Too Many Granules

A failure of this sort is caused by too many granules meeting the criteria for input granules for a particular DPR. At PGE registration, the number of granules we expect for each input ESDT is defined. We define the minimum number and the maximum number of granules we expect. If the number of granules found is not between the minimum and maximum number, the request fails. This will fail either during production request time, or when the PCF File is generated (Autosys PreProcessing step).

The Production Request fails due to too many granules

The Production Request fails. The ALOG displays an error message as follows:
Msg: PlPge::GetInputForDpr - Extr input to process DPR MoPGE01#2007081600OPS, for data type id MOD000#001, with logical id 599001. PlDataTypeReq has a scienceGroup of for this datatype. Expected 2 max inputs, but got 3. Priority : 2
Time : 07/09/99 17:10:52

The problem in this case was that the Production Request Editor queried the PDPS database for granules that would satisfy the data needs for this DPR and found 3 granules instead of 2 like it expected. When we inspected the PDPS database (PlDataGranuleShort) we found only two entries that satisfied the data needs for this DPR. After some investigation, the true problem was discovered.

The DPR was for 16:00:00 - 18:00:00. This particular DPR takes in the current input granule (16:00:00-18:00:00) and the previous input granule (14:00:00- 16:00:00). When it inspects the PDPS Database for data granules, it "pads" the timeframe with a (5A) configurable percentage or (4PY) a hardcoded percentage - 50%. This was 4PY, so the PRE added 50% of the granule's expected time to each side of the time-range of the granules. This means that the PRE queried the PDPS database for granules that had start and stop times within 15:00:00 - 19:00:00. In this case, there was an invalid data granule with a start time of 15:30:00 - 15:59:59. This granule was found during the query and caused the PRE to find one too many granules and fail the PR.

The Workaround:
Delete the offending granule and recreate your production request.

The Fix:

For 5A, the padding that is added to each side of the range for the query is configurable. If the padding is decreased, these kinds of granules will not be found. In addition, there is a minimum size that a granule must be before the Planning system will recognize it as a valid granule. This is a percentage, and is hardcoded in 4PY to 20% (or so), but this is configurable in 5A. So in 5A we can set this value to 50% which will filter out all granules less than 1 hour long, and we can set the padding to 50% which will filter out granules that do not start before 15:00:00. Since our granules must be an hour long, and we can assume they do not overlap, we would not get any invalid granules such as this one.

The PreProcessing Job has Failed due to Too Many Granules
This condition is indicated when the Staging Job is red in AutoSys. This happens due to the same condition as noted above with the PRE, but the data granules came in after the Production Request was generated. The error message appears in the PGE log files and complains about not being able to generate the PCF file due to too many granules for a particular logical id.
The Workaround:
Delete the offending granule and the production request and recreate the production request.
The Fix:
See fix for the previous scenario.

PDPS Troubleshooting - Jobs are activated, but do not get started in AutoSys

The Job Management Server is down
The DPR is waiting in the AutoSys queue (never got released)
Subscription Server Problems
The DPR was released but failed due to a JIL failure
The DPR was released but failed due to a AutoSys ID failure
The DPR was released but failed to be received by Job Management Server

AutoSys is not functional

AutoSys is full

The Job Management Server is down

1. Bring up ECS ASSIST
2. Select the correct mode
3. Select DPS for the subsystem
4. Click on the "monitor" button
5. Observe that the status for the EcDpPrJobMgmt server is "UP"
6. If it is "down", then do steps 7 - 8
7. Click on the "start" button
8. Repeat steps 1-5

The DPR is waiting in the AutoSys queue (never got released)

The Job Management server may have never received a ReleaseDprJob command from the PLS Subscription Manager

1. Check the database table, DpPrCreationQueue, to see if the job is still waiting to into AutoSys. If it's in this table then it probably never got a ReleaseDprJob command get from the PLS Subscription Manager, unless AutoSys is full)
2. Check the Job Management error log file to see if the ReleaseDprJob command was sent
3. If it was, then there may have been a JIL (AutoSys Job Information Language) processor problem -- a JIL FAILURE
4. If you can't find any evidence that the command was sent to Job Management, then the PLS Subscription Manager didn't send the ReleaseDprJob command. It won't send this command if it doesn't think all of the DPR's required inputs have been received. Verify this for yourself as follows:

   a. For regular DPRs (ie. one without optional inputs), check to see if all of the required inputs are present. First look at the PlDprData table and find all of the granule Ids with an ioFlag of 0 (an input granule) for this DPR. Then look at the UR column for each granule Id in PlDataGranule. If all of the input granules have URs (as opposed to granule Ids), then the Subscription Manager should have sent a ReleaseDprJob command to Job Management.

   Look at the Subscription Manager log file to verify that it never, in fact, sent the ReleaseDprJob command.

   b. While you're looking at the Subscription Manager log file, check to see if it got a subscription notification from the

   Subscription Server for any dynamic data that the DPR needs. If you believe that all of the necessary input files for
   the DPR have been inserted by another DPR, then there may be Subscription Server Problems

   c. If there are no Subscription Server Problems, all of the input granules for the DPR have URs and/or Subscription Manager received notification for all dynamic granules, then something may be wrong with the Subscription Manager. Check with somebody in the PLS subsystem.

The DPR was released but failed due to a JIL failure

A "JIL Failure" means that the Job Management Server had some problem placing the DPR in AutoSys. The Job Interface Language processor rejected the create job command sent to it by the Job Management Server. If you look at the completionState column for the DPR in the PDPS PlDataProcessingRequest table, you will see "JIL_FAILUR". There are 2 main reasons for this:

1. There is already a job with an identical name in AutoSys. Check this by going to the Ops Console, select View->Select Jobs and type a portion of the job name in the "Job Name" box, bracketed by the "*" or "%" wildcard character. If the job is already in AutoSys, it must be removed by using the Production Request Editor or by using the Job Management Server

Client tool (selectable from the Ops Console). Never delete a job from AutoSys using the job definition GUI. This will corrupt the PDPS database.

2. The event processor is down - AutoSys is not functional.
3. The job had a problem when it was loaded into AutoSys and a malformed or mutant job box is the result. This is a job box which will stay dark blue (meaning that it was not activated) and will be missing one of the seven job steps. To correct this problem you must do the following:

- Delete the job from AutoSys by hand. To do this select the job from JobScape and right click. Select the Job Definition and then select Delete from the pop-up window. In general, it is bad practice to delete a job from Autosys using the Job Definition GUI. This can cause corruption in the PDPS database. But for this problem there is no other solution.
- Update the completionStatus of the DPR in the PDPS database for which the mutant job box was created. You must do this via isql and set the completionStatus = NULL (using the isql update command).
- Delete the DPR that maps to the job via the Production Request Editor. Note that you do not want to delete the entire Production Request, only the DPR that had the mutant Job Box. Any DPRs that depend on this DPR will also have to be deleted.
- Re-create this DPR and any subsequent DPRs via the Production Request Editor.

The DPR was released but failed due to a AutoSys ID failure

An "AutoSys ID" failure is indicated if the following messages appear in the Job Management ALOG file:

```
PID : 7668:MsgLink :0 meaningfulname :DpPrAutosysMapList::GetAutosysIDByDpr
 Msg: unable to find autosys id for dpr: ACT#syn1#004130123DEV02 Priority: 2 Tim
e : 03/09/99 11:33:51
PID : 7668:MsgLink :9 meaningfulname :CantFindAutoSysId
 Msg: Unable to find autosys id Priority: 2 Time : 03/09/99 11:33:51
PID : 7668:MsgLink :10 meaningfulname :DpPrSchedulerDObjSmainCreateFailed
 Msg: RqFailed=CreateDpr DprID=ACT#syn1#004130123DEV02 Priority: 2 Time : 03/09/99
11:33:51
```

An "AutoSys ID" Failure means that the Job Management server could not associate the AutoSys ID with the DPR that was activated. When the Job Management server is started it reads various tables in the PDPS database that provide the linkage between processing resources and AutoSys instance. If data is missing from these tables, or was added after the Job Management server was started, then the error shown above can occur when any jobs are activated by the Planning Workbench.

The following actions should be taken when an "AutoSys ID" failure error is reported:

1. Verify that the **PlResource** table in the PDPS database has at least 1 entry for a processing string and at least one entry for an AutoSys Instance. If either of these are missing, then you need to re-do Resource Planning and add them via the Resource Editor GUI.
2. Verify that the **PlRscString** table in the PDPS database has at least 1 entry and that autosysIdKey matches the entry in the PlResource table. Again, if information is missing or wrong, you need to re-do Resource Planning.
3. Verify that the **DpPrAutosysMapList** table in the PDPS database has at least 1 entry and that **resourceString and autosysIdKey** matches the entry in the **PlRscString** table. Yet again, if information is missing or wrong, you need to re-do Resource Planning.
4. If Resource Planning has been done after the Job Management server was brought up, then bounce the server. Since the server reads this information at start up, any changes since it was brought up will NOT have taken affect.

The DPR was released but failed to be received by Job Management Server

In this case, the Planning Workbench thinks it successfully activated the DPR(s) but the Job Management Server had trouble receiving the notification.

Look for the following in the **EcDpPrJobMgmtDebug.log:**

AutoSys is not functional

1. Set the AutoSys environment variables by sourcing something that looks like /data/autotreeb/autouser/AS1.autosys.csh,

   where AS1 is an autosys instance name.
2. type chk_auto_up and verify that you see the message: "Primary Event Processor is RUNNING on machine:
   machine-name"
3. If you don't see that a Primary Event Processor is running, check with your AutoSys administrator.

Note that if Autosys will not stay up **(Autosys administrator brings it up and it goes down right away)** the following could be occurring:

- It may be possible that too many events were queued up to AutoSys while it was down. If Autosys detects a certain number of events in a short time period, it brings itself down. The only way to handle this is to keep bringing Autosys back up. Each time it will work through a few of the events before it detects "too many" and shuts down. Eventually the events will be cleared out and Autosys will stay up.
- It may be the sql server is not up for AutoSys (this may be a different server than the one needed for the PDPS database). Look for the following error messages in the AutoSys log or when you attempt to bring up JobScape:

  Couldn't create DBPROCES

Unable to get encoded and plaintext passwords for l0sps03_srvr:FMR

AutoSys is full

This is an unlikely problem, and would only occur under the following conditions:

- The number of job boxes in the AutoSys instance > DpPrAutoSysMaxJobs/8. Look in EcDpPrJobMgmt.CFG to get this number.
- The DPR completionState in PlDataProcessingRequest is CQ_RELEASE.

What this means is that the Job Management Server got the command from Subscription Manager to release the job, but that no more jobs can fit into AutoSys at present. Wait for a DPR to finish, so that the next waiting one can be put into AutoSys.

PDPS Troubleshooting - SDSRV Troubleshooting

How to examine the SDSRV Database

1. Log into the Science Data Server (SDSRV) machine
2. Bring up ECS Assist
3. Select "Subsystem Manager"
4. Select "DB Viewer"
5. Click on Login
6. A list of data types will appear. Click on the data type and information about all granules in the archive will be displayed.

PDPS Troubleshooting - Quick Tips

Job Management fails with a "JIL FAILURE" in the .ALOG file when trying to cancel a job in AutoSys

Use AutoSys Job Definition to see who owns the job.

Then select "Adv Features" to see if the user who is trying to delete the job is in the same group for "Edit Dfn" as the user who owns the job.

A quick fix is to give the job world "Edit Dfn", but generally, whoever starts the Job Management Server should be in the same group as the person using the tool to cancel the DPR.

This page intentionally left blank.

Using IQ Software to Create Reports

Creating Reports Using IQ Software

ECS no longer plans to offer a Report Generator GUI. Consequently, DAAC operations personnel must use other means to generate various types of reports.

IQ (Intelligent Query) software is a set of commercial off-the-shelf (COTS) products that provides flexible access to the PDPS database from which data for reports can be retrieved. The cost of that flexibility is a somewhat complicated process for initially setting up reports. However, once a particular type of report has been set up, reports can be generated fairly quickly.

The procedure for creating reports using IQ software starts with the assumption that the Production Planner has logged in to the system.

Creating Reports Using IQ Software

NOTE: If using an X-Terminal, it may be necessary to add the following line to the **.Xdefaults** file in the home directory before performing the task for the first time:

```
iqx*background:                    grey
```

NOTE: Commands in Steps 1 through 8 are typed at a UNIX system prompt.

1 At the UNIX command line prompt **rlogin** ***hostname*** then press the **Return/Enter** key on the keyboard.

hostname refers to the host (e.g., e0mss21, g0mss21, l0mss21, or n0mss21) on which GUIs are to be launched during the current operating session. Multiple hostnames can be specified on the same line.

2 Type **setenv DISPLAY** ***clientname*****:0.0** then press the **Return/Enter** key.

Use either the X terminal/workstation IP address or the machine-name for the *clientname*.

When using secure shell, the DISPLAY variable is set just once, before logging in to remote hosts. If it were to be reset after logging in to a remote host, the security features would be compromised.

3 Open another UNIX (terminal) window.

4 Start the log-in to the Applications Server host by typing **/tools/bin/ssh *hostname*** (e.g., **e0mss21**, **g0mss21**, **l0mss21**, or **n0mss21**) in the new window then press the **Return/Enter** key.

If you receive the message, Host key not found from the list of known hosts. Are you sure you want to continue connecting (yes/no)? type yes ("y" alone will not work).

If you have previously set up a secure shell passphrase and executed sshremote, a prompt to Enter passphrase for RSA key '<*user@localhost*>' appears; continue with Step 5.

If you have not previously set up a secure shell passphrase; go to Step 6.

5 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** then press the **Return/Enter** key.

Go to Step 7.

6 At the **<*user@remotehost*>'s password:** prompt type your ***Password*** then press the **Return/Enter** key.

7 Type **setenv ECS_HOME /usr/ecs/** then press the **Return/Enter** key.

When logging in as a system user (e.g., cmshared), the ECS_HOME variable may be set automatically so it may not be necessary to perform this step.

8 Type **cd /usr/ecs/*MODE*/COTS/iqx5** then press **Return/Enter**.

Change directory to the directory containing the IQ software (directory path may vary from site to site).

The *MODE* will most likely be one of the following operating modes:

— OPS (for normal operation).

— TS1 (for Science Software Integration and Test (SSI&T)).

— TS2 (new version checkout).

Note that the separate subdirectories under /usr/ecs apply to (describe) different operating modes.

9 Type **iqx &** then press **Return/Enter**.

If the GUIs are not displayed when the command iqx is given, try using ./iqx instead.

The iqx IQView GUI (Figure 46) and either the iqx Sybase Database Logon GUI (Figure 47) or iqx Open IQView GUI (Figure 48) are displayed.

– If IQViews have been defined previously, they are listed on the **iqx Open IQView** GUI (Figure 48); otherwise, a list of database tables is displayed.

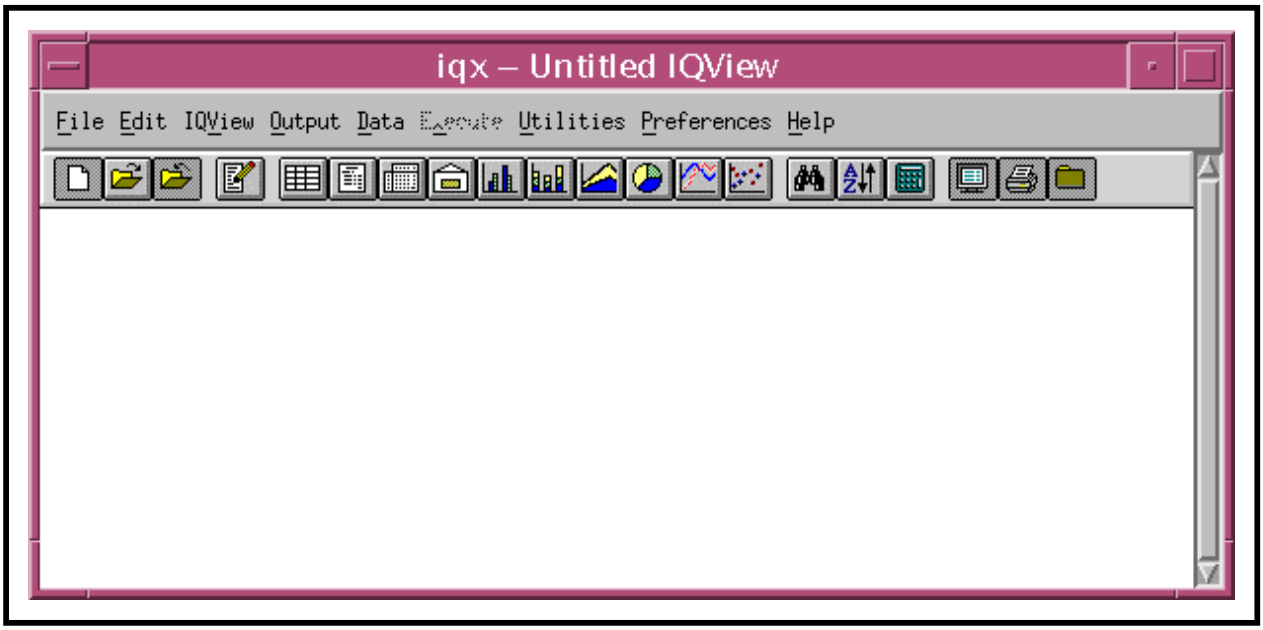


Figure 46. iqx IQView GUI

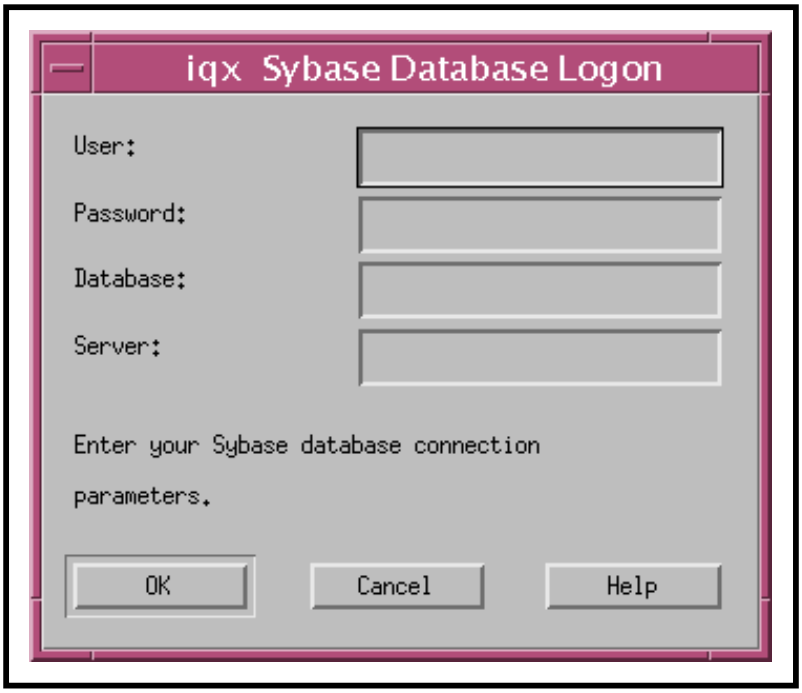


Figure 47. iqx Sybase Database Logon GUI

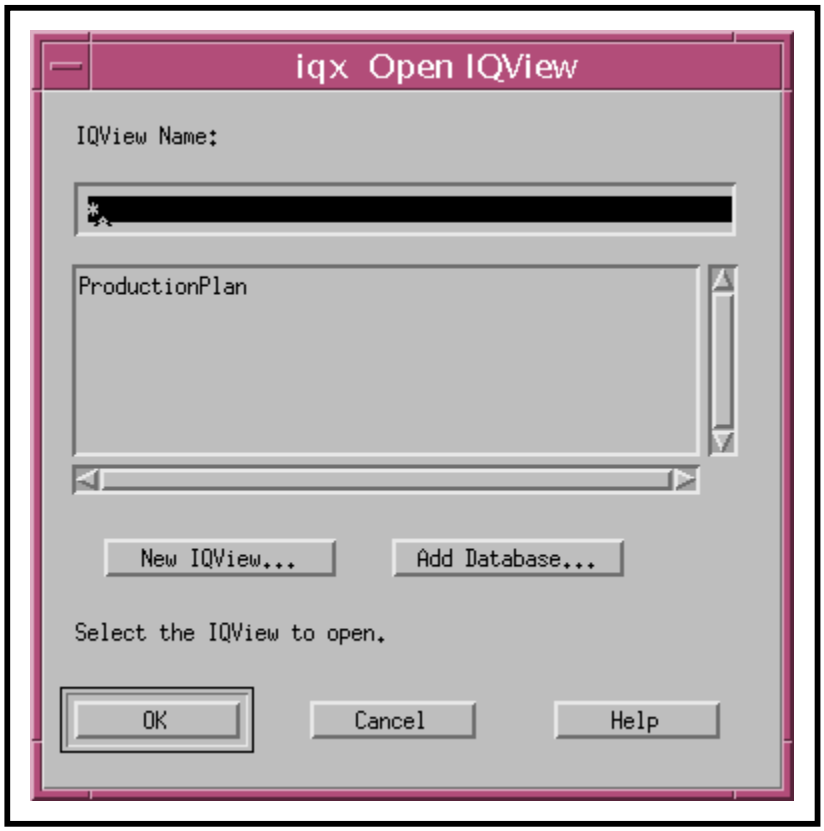


Figure 48. iqx Open IQView GUI

If the iqx License dialogue box (Figure 49) is displayed, click on the Cancel button.

– The **iqx IQView** GUI (Figure 46) and either the **iqx Sybase Database Logon** GUI (Figure 47) or **iqx Open IQView** GUI (Figure 48) are displayed.

10 If the **iqx Sybase Database Logon** GUI (Figure 47) is displayed, go to Step 16.

11 If the **iqx Open IQView** GUI (Figure 48) is displayed and the desired IQView has been defined previously, perform Steps 12 through 14; otherwise, go to Step 15.

If IQViews have been defined previously, they are listed on the iqx Open IQView GUI (Figure 48); otherwise, a list of database tables is displayed.

12 If the desired IQView has been defined previously, highlight the IQView to be opened by clicking on its entry in the list of IQViews.

13 Click on the **OK** button.

– The **iqx Sybase Database Logon** GUI (Figure 47) is displayed.

14 Go to Step 16.

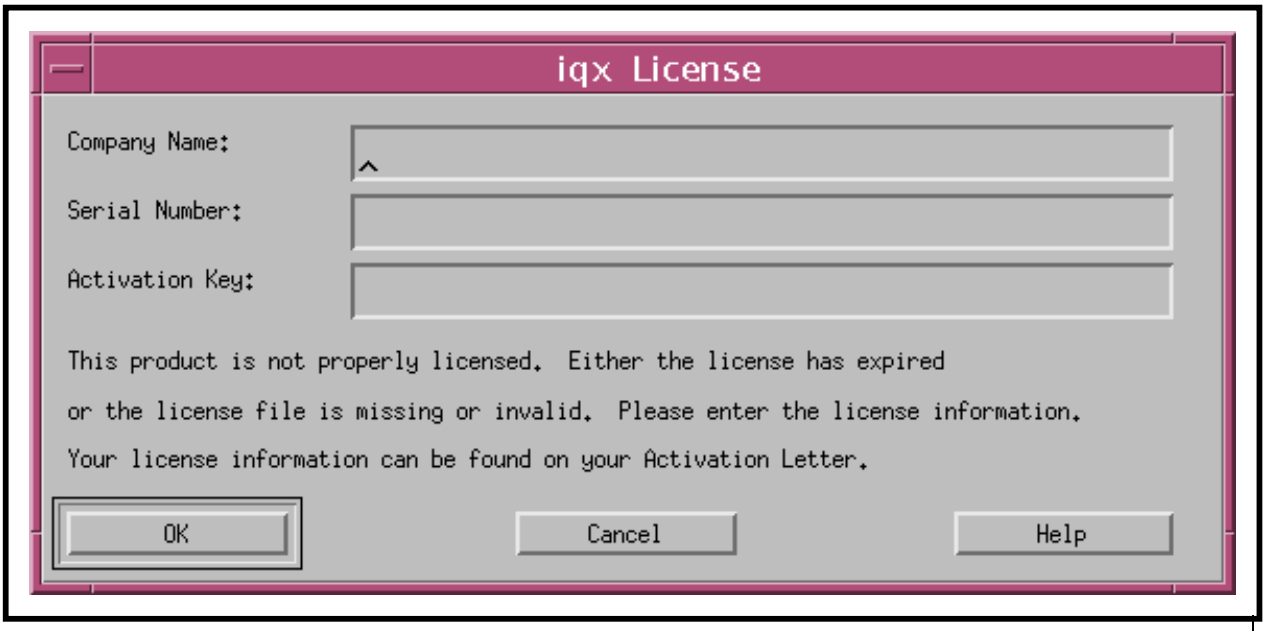


Figure 49. iqx License Dialogue Box

15 If the **iqx Open IQView** GUI (Figure 48) is displayed and the desired IQView has not been defined previously, click on the **Add Database…** button.

– The **iqx Sybase Database Logon** GUI (Figure 47) is displayed.

16 When the **iqx Sybase Database Logon** GUI (Figure 47) is displayed, type the appropriate entries in the following fields:

User:

– For example: **pdpsUsr**

– The DAAC Database Administrator can provide the actual values to be entered.

Password:

– For example: **dbpa$$wd**

Database:

– For example: **pdps_TS1**

Server:

– For example: **x0pls02_srvr**

17 Click on the **OK** button.

Either the iqx Open IQView GUI (Figure 46) or the iqx IQView GUI (Figure 47) is displayed.

18 If the **iqx Open IQView** GUI (Figure 48) is displayed, continue with Step 19; if the **iqx IQView** GUI (Figure 46) is displayed, go to Step 22.

19 Click on the **New IQView…** button.

The iqx Table Selection GUI (Figure 44) is displayed.

20 Move database table names between the **Available Tables:** and **Selected Tables:** lists as necessary by selecting (highlighting) the name of the table to be moved, then clicking on either the **Select** or **Remove** button (as applicable) to move the table name to the other list.

Database tables and the columns within each table are described in the 311-series documents (e.g., 311-CD-503-001, Release 5A Planning and Data Processing Subsystem Database Design and Schema Specifications for the ECS Project). The documents are available on the ECS Data Handling System (i.e., at http://edhs1.gsfc.nasa.gov).

21 When the desired table(s) has/have been moved to the **Selected Tables:** list, click on the **OK** button.

The iqx IQView GUI (Figure 46) is displayed.

22 Select **Output → Columnar** from the pull-down menu.

The iqx Columnar Output GUI (Figure 52) is displayed.

23 Move database table column names between the **Available columns:** and **Selected columns:** lists as necessary by selecting (highlighting) the column to be moved, then clicking on either the **Select** or **Remove** button (as applicable) to move the column name to the other list.

The order in which columns are listed in the Selected columns: list is the order in which the columns will be listed in the eventual report.

Database tables and the columns within each table are described in the 311-series documents (e.g., 311-CD-503-001, Release 5A Planning and Data Processing Subsystem Database Design and Schema Specifications for the ECS Project). The documents are available on the ECS Data Handling System (i.e., at http://edhs1.gsfc.nasa.gov).

24 If changing the order in which columns are listed in the **Selected columns:** list, select (highlight) the column to be moved, then click on the **Up** or **Down** button as necessary to reposition the selected column.

Highlighted column changes position in the Selected columns: list.

25 When the desired columns have been moved to the **Selected columns:** list, click on the **OK** button.

The iqx IQView GUI (Figure 53) is displayed.

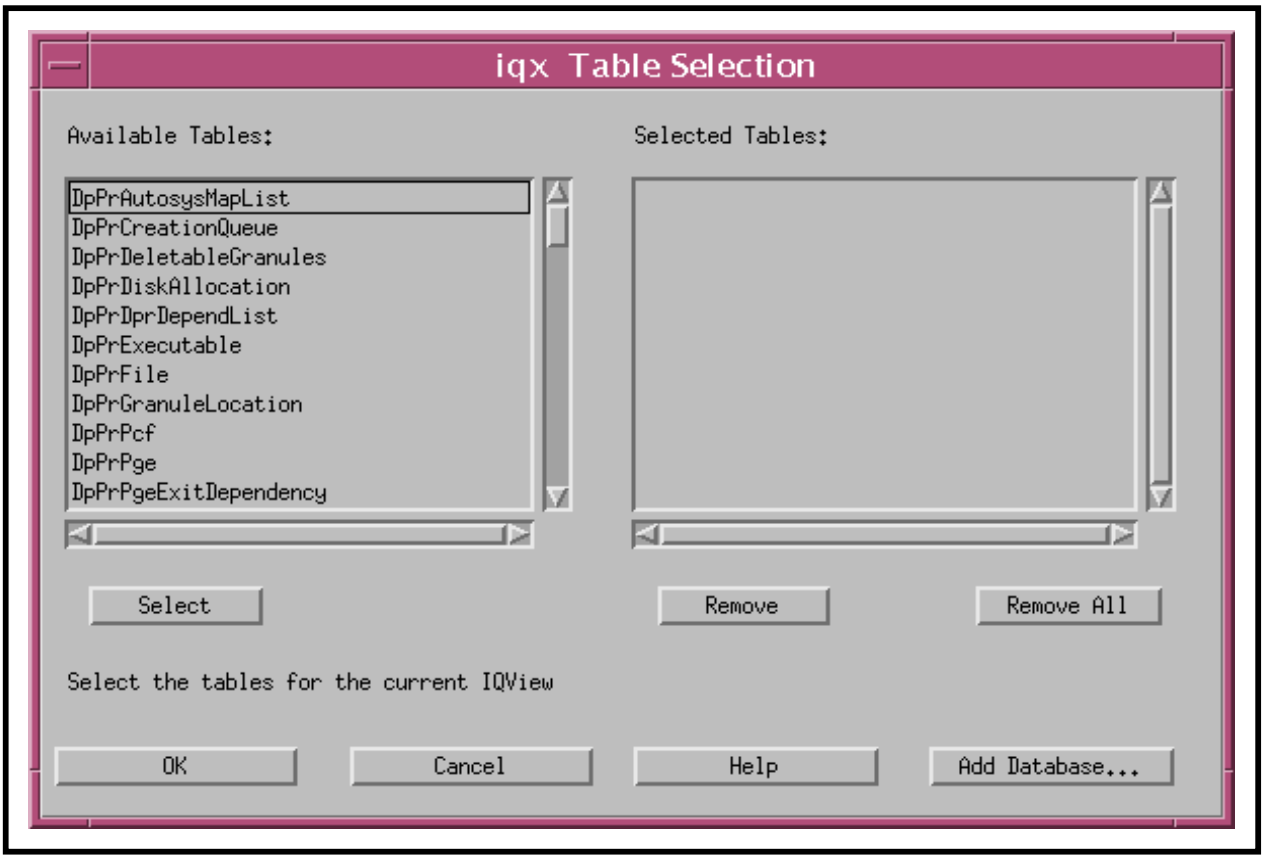


Figure 50. iqx Table Selection GUI

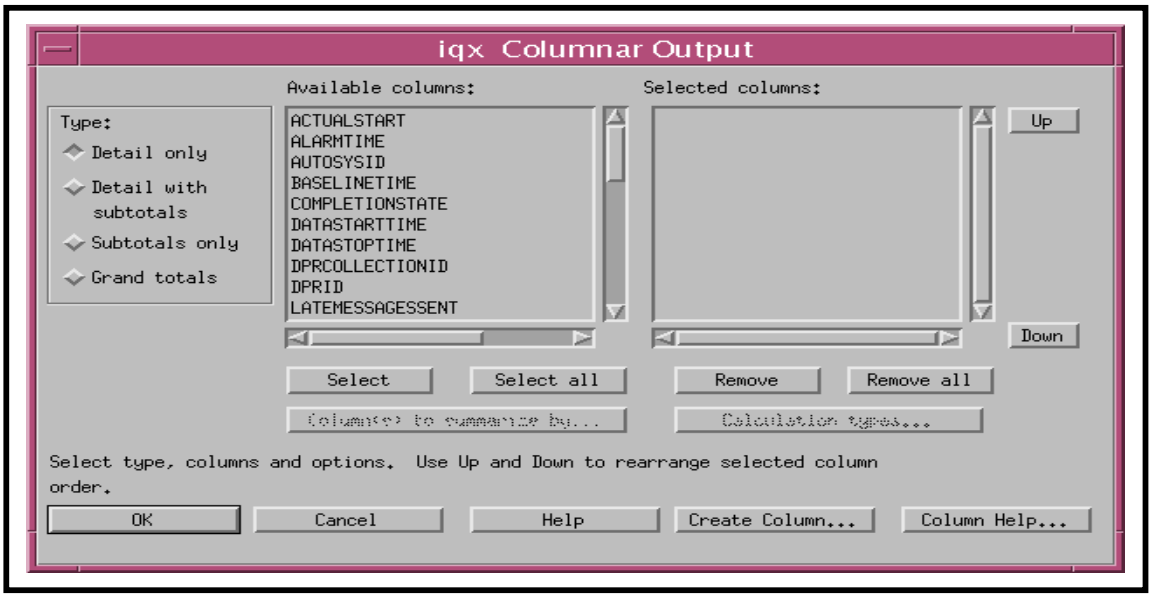


Figure 51. iqx Columnar Output GUI

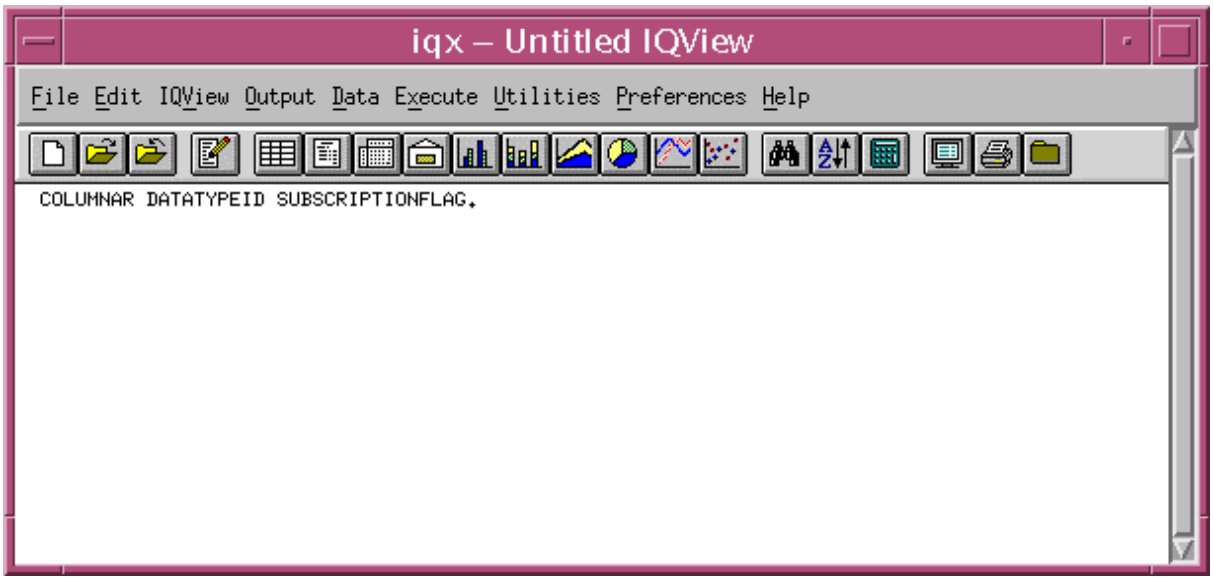


Figure 52. iqx IQView GUI

– Where **/home/cmshared/** represents the path and **reportfile** is the file name.

26 Click on the **OK** button

The columnar selections are listed on the iqx IQView GUI as shown in Figure 53.

27 To generate a report make one of the following selections from the pull-down menu:

Execute → to Display – to display the report on the terminal screen.

– The **iqx IQ Output** GUI (Figure 53) is displayed.

– Go to Step 35 after viewing the report.

Execute → to Printer – to print the report.

– The **iqx Execute to Printer** GUI (Figure 54) is displayed.

– Go to Step 29.

Execute → to File – to save the report in a file.

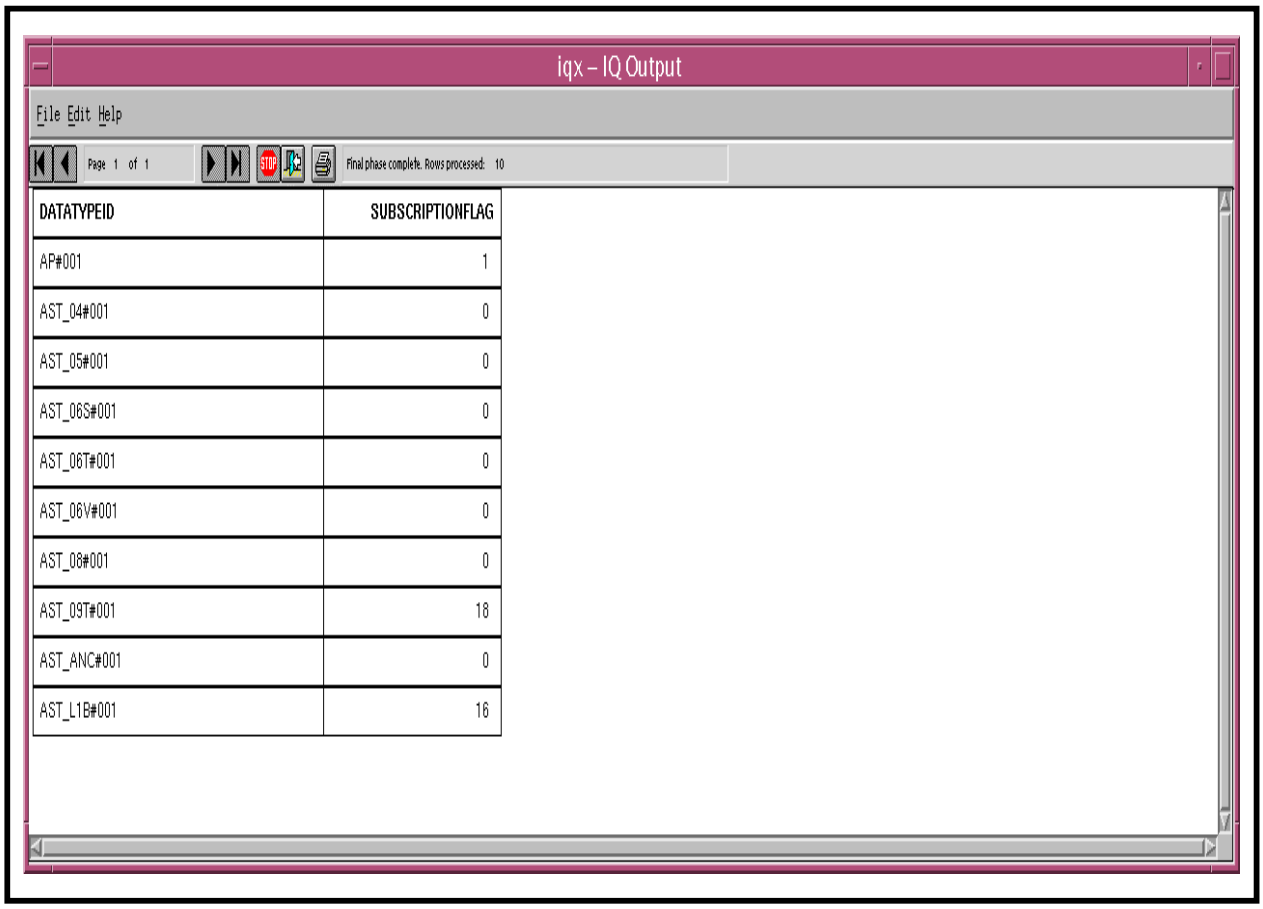
– The **iqx Execute to File** GUI (Figure 55) is displayed.

– Continue with Step 27.

28 Type a valid ***path/filename*** in the **Name:** field of the **iqx Execute to File** GUI (Figure 55).

For example: /home/cmshared/reportfile

Go to Step 35.



DATATYPEID	SUBSCRIPTIONFLAG
AP#001	1
AST_04#001	0
AST_05#001	0
AST_06S#001	0
AST_06T#001	0
AST_06V#001	0
AST_08#001	0
AST_09T#001	18
AST_ANC#001	0
AST_L1B#001	16

Figure 53. iqx IQ Output GUI

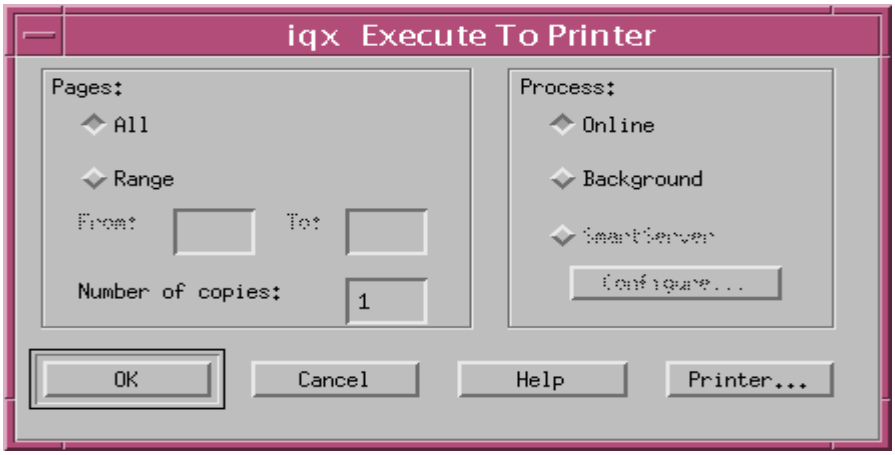


Figure 54. iqx Execute to Printer GUI

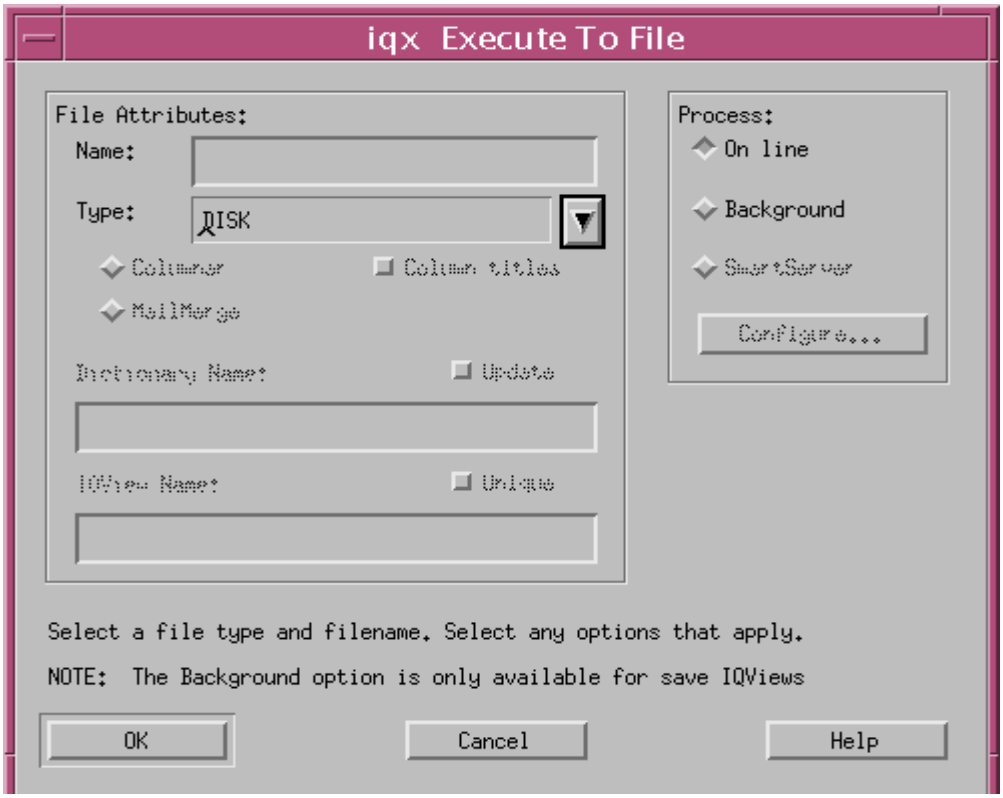


Figure 55. iqx Execute to File GUI

29 Click on the **Printer…** button on the **iqx Execute to Printer** GUI (Figure 54).

The iqx Print Setup GUI (Figure 56) is displayed.

30 To list the available printers, first click on the option button associated with the **Printer** field.

An option menu of printers is displayed.

31 Highlight the desired printer in the option menu.

The desired printer is shown in the Printer field.

For example: Postscript printer one.

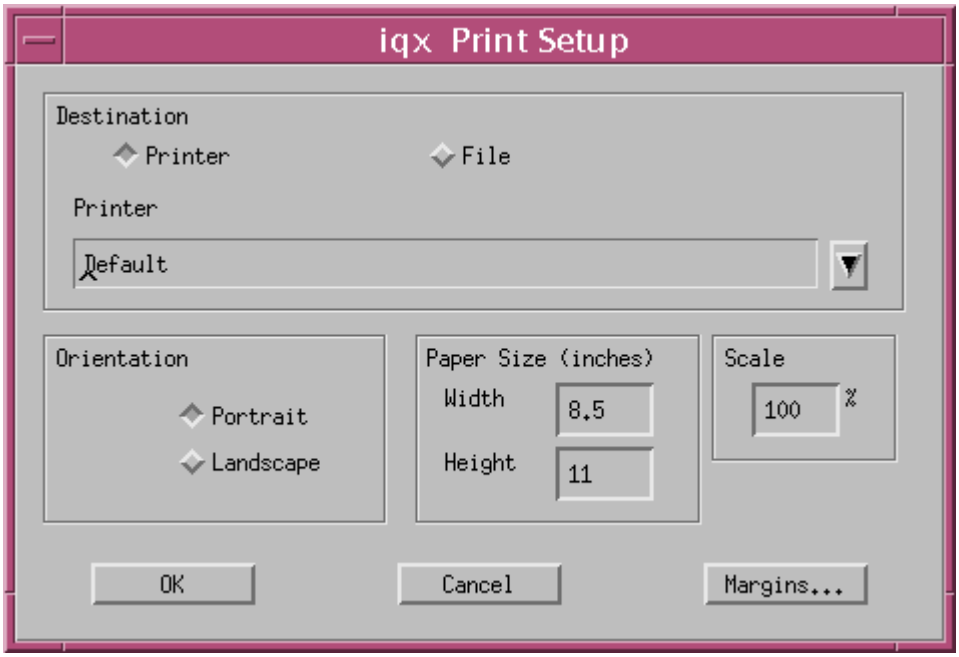


Figure 56. iqx Print Setup GUI

32 If a report in landscape format is desired, click on the **Landscape** button.

33 Click on the **OK** button.

The iqx Print Setup GUI (Figure 56) is dismissed.

The iqx Execute to Printer GUI (Figure 54) is displayed.

34 Click on the **OK** button.

35 To save the procedure/IQView, continue with Step 36; otherwise go to Step 43.

36 Select **File → Save Procedure As…** from the pull-down menu.

The iqx Save IQView GUI (Figure 57) is displayed.

37 Type a file name for the IQView in the name field.

38 Click on one of the following buttons if applicable:

Public.

Public read only.

Private.

39 Click on the **OK** button.

The iqx Save Procedure GUI (Figure 58) is displayed.

40 Type a file name for the procedure in the name field.

41 Click on one of the following buttons if applicable:

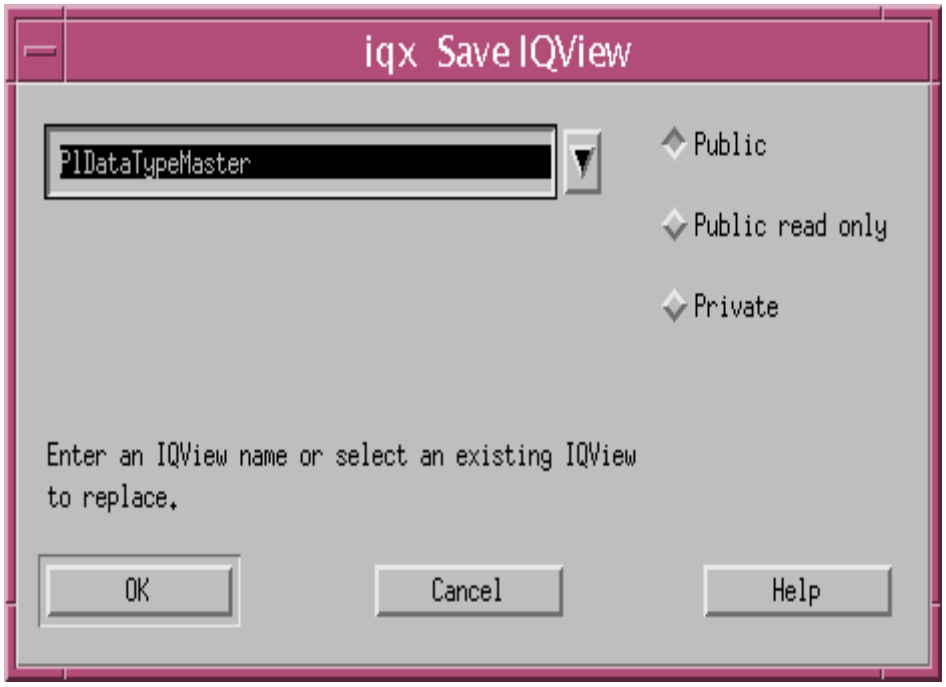


Figure 57. iqx Save IQView GUI

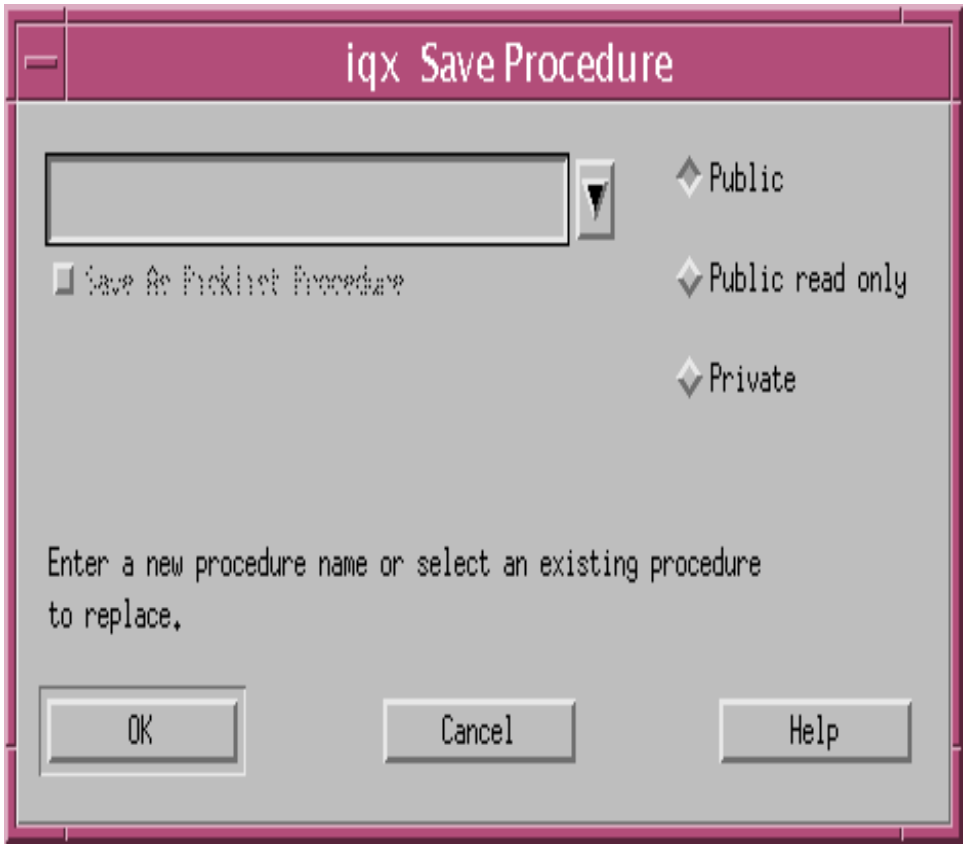


Figure 58. iqx Save Procedure GUI

Public.

Public read only.

Private.

42 Click on the **OK** button.

43 Select **File → Exit** from the pull-down menu to exit from the **iqx IQView** GUI (Figure 46).

Formatting IQ Software Reports

[TBS]

Deleting Granules

As of Release 6A, the system provides a **Granule Deletion** tool, complementing the automatic, scheduled deletion capability that permits operators to delete products produced and archived by the Planning and Data Processing subsystems on a scheduled basis (e.g., deletion at a certain time (configurable by the operator) after product creation.

The **Granule Deletion** tool allows operators to delete products on demand, see Figure 59, STMGT GUI screen for granule deletion from archive . There are a variety of circumstances that may require deletion on demand, such as:

New PGE versions have been created and are used to reprocess large amounts of past data, creating new ESDT versions. As reprocessing progresses, operations deletes the granules for the old ESDT versions from the archive and inventory.

It is determined that certain lower-level (e.g., Level 2) products are of little or no interest to the science or public user community. In concert with the science teams, DAAC operations personnel decide to remove these products from the inventory. Since the products are still referenced by higher-level products as inputs, the DAAC decides to keep the inventory records for production history purposes.

One or more granules were found defective and were reprocessed on an individual basis. When the reprocessing is complete, the operator wishes to delete the old, defective granule(s) from the inventory.

A DAAC has extended ECS with subsetting services. The subsetted products are produced outside ECS, but are then inserted into the ECS archive to take advantage of the ECS distribution capability. The DAAC writes a script to delete the subsetted products on a regular basis.

Deletion Capability and Features

The Science Data Server has provided an application programming interface (API) for deleting granules from the archive, or from both the archive and inventory since earlier releases, but the Granule Deletion tool adds a front-end command-line utility that provides several ways for selecting granules for deletion. Confirmation is generally required so that granules are not inadvertently deleted. However, the confirmation may be suppressed so that operators can run regularly scheduled deletion scripts using background execution. This suppression possibility presents an opportunity for inadvertent loss of data and so must be used with care and only after thorough testing of any deletion script.

The Science Data Server captures deletions and related errors in the application log. Operators may also specify a separate and independent delete log for immediate analysis of the success or failure of a delete operation.

Deletion Sequence

The deletion of granules from the archive involves three elements, and therefore actually occurs in stages. Two of the elements are scripts that address the Science Data Server (SDSRV), and the third is a part of the Storage Management (STMGT) Graphical User Interface (GUI).

For the first stage, a delete script applies deletion checks to the selected granules, "logically" deleting from the inventory those granules that satisfy the checks. These granules are flagged as 'deleted' and can no longer be accessed, but their inventory entries are not yet removed. The deletion flag consists of a time stamp recording the logical deletion time.

The second stage is actual deletion from the inventory, which occurs when the operations staff runs the physical deletion script. The script removes all inventory rows for granules that were flagged as 'deleted,' and produces the list of the granule files that are now eligible for deletion from the archive. That list is transferred to the STMGT database. The operations staff controls the lag time between logical deletion and physical deletion. That lag time is entered into the physical deletion script, which deletes only inventory entries for granules that have been logically deleted prior to that time period.

STMGT provides a GUI screen, as illustrated in **Fiugre 59**, that allows the operator to initiate the removal from the archive of the files listed its deletion table (populated by SDSRV). STMGT creates requests to the archive servers to delete files. The STMGT GUI can be used to look at the state of the deletion requests. Files that are successfully deleted have their associated rows removed from the STMGT database table.

Periodically, as sufficient data removal from the archive makes it appropriate, operations may elect to reclaim the tape space and recycle archive tapes. The AMASS software commands (*volcomp*, *volclean*, *volformat*, *volstat*) are used for that purpose.

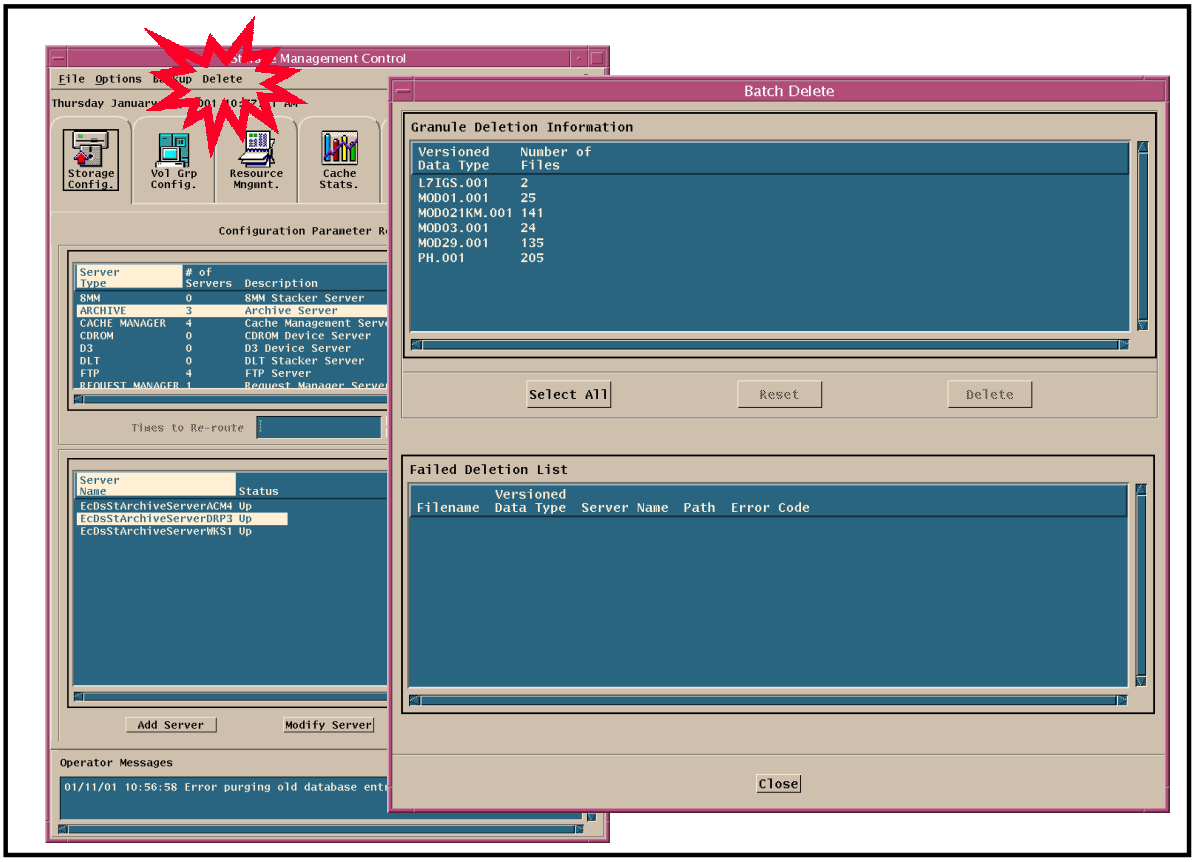


Figure 59. STMGT GUI screen for granule deletion from archive

Granule Deletion Procedures

This process of Granule Deletion is for operators to delete granules from the inventory/archive on demand. Deletion of Granules is a three-part process.

- **Granule Deletion Client**
- **Deletion Cleanup Utility**
- **STMGT GUI**

1. First, a command line SDSRV utility, called the **Granule Deletion Client**, is available for selecting granules to be marked for deletion from the archive (DFA), with an option to also mark for delete from the inventory (PHYSICAL DELETE). A confirmation is generally available, but can be suppressed to allow for background execution. The Science Data Server logs will be inspected for messages related to deletions, and a separate log is available for immediate inspection of only deletion messages.
2. The second, a SDSRV script, called the **Deletion Cleanup Utility**, is run to send those granules that have been marked for deletion to the STMGT database so they can be

physically deleted from the archive. In addition, if granules were marked for Physical Delete in SDSRV, they will also be deleted from the SDSRV database. A lag time is used so the operator can request that not all the granules marked for deletion are immediately deleted. Granules will be deleted when the Deletion Cleanup Utility is executed when they have been marked for deletion before <today - lagtime>. This lag time can be overridden by entering a lag time of 0, in which case all granules marked for deletion will be immediately deleted.

3. The third and last part of the process is the **STMGT GUI** can then be used to complete deletion of data granules from the archive (AMASS). Granules are selected for deletion from the STMGT GUI via the datatype name (ESDT ShortName) and Version ID.
4. Using the Granule Deletion Client, granules can be selected for deletion several different ways:
   - ESDT ShortName, Version and granule time coverage
   - ESDT ShortName, Version and granule insert time range
   - Separate Input file containing SDSRV Granule IDs
   - Separate Input file containing ShortName, Version, and Local Granule ID (Logical Granule ID noted in the ticket is referred to in this test as Local Granule ID, which is the name of the parameter as it resides in the database)
5. The number of granules returned can be verified by making the same query in the database using SQL commands. Granules can be deleted from both the inventory and the archive, from the archive only, in the foreground, and via background (cron) jobs. It can be verified that all occurrences of a file in the archive are deleted when requested, such as from a primary and a backup archive location. It can be verified that Browse, QA and PH granules associated with physically deleted granules are also deleted if not referenced by any other granules.

Error conditions include:

- Attempts to delete granules that are still being referenced by other granules
- Unauthorized users are indeed halted from deleting data
- Recovery from Archive Server, Science Data Server and Science Data Server DBMS server faults occurring during deletion requests

6. Using a user ID authorized for granule deletions, the operator has a number of interfaces to consider for the deletion of granules, including the mechanisms for selecting the granules to be deleted and the confirmation of the deletion.

   a. Granules can be selected by ESDT short name, ESDT version, and granule time coverage

   b. Granules can be selected by ESDT short name, ESDT version, and granule insert time range

   c. Granules can be specified in a separate input file containing either SDRV Granule Ids or Logical Granule Ids

   d. The input file can list granules belonging to collections belonging to different logical volume groups.

   e. The operator can optionally list the geoID and logical granule ID of each of the granules selected for deletion.

   f. The number of granules selected for deletion is displayed to the operator and the operator is asked to confirm the deletion

   g. The operator can suppress the confirmation prompt via a command line argument

h. Granules are tagged for deletion from inventory and archive or from archive only, depending on operator choice
i. By default, BROWSE, QA, and PH granules associated with physically deleted granules are deleted if no longer referenced otherwise
j. The operator can suppress deletion of BROWSE, QA and PH granules
k. The files associated with the deleted granule are deleted from all archive locations (including back-up locations)
l. The deletions are logged as required to the SDSRV application log file and the operator specified granule deletion log file
m. Granule deletions can be performed during normal SDSRV processing, i.e., while other requests such as insert, acquire, and searching are in progress

Granule Deletion Client Steps

Workspaces are created and each process is carried out for Granule Deletion

1. Create the following workspaces on the CDE window manager (xterms created below):

- **SDSRV 5 xterms**
- **STMGT 7 xterms**
- **GUIs 4 xterms**
- **Database 2 xterms**

2. In each workspace, open up the number of xterm windows as is indicated for each workspace above:

- Invoke a terminal window if necessary
- Log on to the appropriate machine for each workspace and create xterms for each workspace (see following commands as an example)
- **setenv DISPLAY <terminalId>:0.0**
- **cd /usr/ecs/<MODE>/CUSTOM/utilities**
- Invoke an xterm: **xterm -sb -sl 5000 –fg white –bg blue -n <workspace.1> &**
  - **Invoke another xterm: xterm -sb -sl 5000 –fg white –bg blue -n <workspace.2>&**
  - **Invoke database xterms as follows: xterm –sb –sl 5000 –fg black –bg green –n nnn_db&**

3.Verify that all Servers are up.

4. The user is logged into the SDSRV and STMGT databases

On the **Database** workspace, log into the SDSRV and STMGT databases from the two xterms and from the appropriate hosts:

- **rlogin t1acs03**
- **isql –U sdsrv_role –St1acg04_srvr**
- **<password>**
- **use EcDsScienceDataServer1_<MODE>**

- **go**
- **rlogin t1dps01**
- **isql –U stmgt_role –St1acg04_srvr**
- **<password>**
- **use stmgtdb1_<MODE>**
- **go**

5. The Distribution and Storage Management GUIs are displayed on the screen
On the **GUIs** workspace, invoke the **DDIST and STMGT GUIs**:
- **rlogin t1dps01**
- **cd /usr/ecs/<MODE>/CUSTOM/utilities**
- **setenv DISPLAY <TerminalId:0.0>**
- **ls *Gui***
- **EcDsDdistGuiStart <MODE>**
- **EcDsStStmgtGuiStart <MODE>**

6. The following step may not be necessary. Check with your DB Manager.
a. Show Detailed Steps to Reset the Lock on the DsMdDeletedGranulesTable, when necessary**.**
In the **SDSRV** workspace,
- **cd /usr/ecs/<MODE>/CUSTOM/utilities**
- **ll *Lock***
- **EcDsResetLock.pl**
- Enter Mode of Operation : **<MODE>**
- Enter Log File name : **ResetLock.log**
- Enter Sybase User Name : **sdsrv_role**
- Enter Sybase Password : **<password>**
- Enter Sybase SQL Server Name : **t1acg04_srvr**

b. Enter SDSRVs database name : **EcDsScienceDataServer1_<MODE>**
The above steps will be performed over and over again, as directed in subsequent steps of this procedure

2. Perform a query in the **Database** workspace against the SDSRV database to determine which granules will be deleted when the Granule Deletion Client is run next:
- **select dbID from DsMdGranules where ShortName = "<ShortName>" and VersionID = <VersionID> and insertTime > "<insertbegdate/time>" and insertTime < <insenddate/time>"**
- **go**

A list of dbIDs is returned. These will be referenced in the next few steps to compare with what the Granule Deletion Client deletes using the same search criteria.

3. On the **SDSRV** workspace, execute the Granule Deletion Client specifying ESDT ShortName, ESDT version and granule insert time coverage:

EcDsGranuleDelete –name <ShortName> -version <version no.> -insertbegin <insbegdate/time> –insertend <insenddate/time> –log

/usr/ecs/<MODE>/CUSTOM/logs/GranDel3.log -display ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>

The script runs to completion. A list of geoIDs and local granule IDs is returned. Check to see if this list compares favorably with the isql query previously performed

View the Deletion log file and the tail of the SDSRV ALOG file:

- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule insert time of the request. These messages also demonstrate that the display option was referenced so the granules to be deleted are listed (geoidID and local granule ID) but not deleted.

A delete with ESDT Name/Version and by Granule Insert Time and specify Data From Archive (DFA), Granules are tagged for deletion

4. On the **SDSRV** workspace, run the Granule Deletion Client specifying ESDT ShortName, ESDT version and granule insert time coverage:
EcDsGranuleDelete –name <ShortName> -version <version no.> -insertbegin <insbegdate/time> –insertend <insenddate/time> –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel7.log –DFA ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>

The script is executed, and displays the number of granules for deletion, and prompts the user as to whether to continue.

When prompted, type: **y,** The script continues to completion.

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule insert time of the request, and a successful completion of the deletion.

5. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel7.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

6. On the **Database** workspace, view the SDSRV database to verify the granules were tagged for deletion:

- **Select * from DsMdGranules where ShortName = "<ShortName>" and VersionID = <VersionID> and insertTime < "<insenddatetime>" and insertTime > "<insbegdate/time>"**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdFileStorage where granuleId = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of NULL for DeleteEffectiveDate and a value of "Y" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and the granule's original insertTime (from the DsMdGranules table) for the insertTime, and a DFA flag of 1.

Delete with ESDT Name/Version and by Granule Insert Time and specify physical delete, Granules are tagged for deletion.

---

1. On the **SDSRV** workspace, run the Granule Deletion Client specifying ESDT ShortName, ESDT version and granule insert time coverage:

- **EcDsGranuleDelete –name <ShortName> -version <version no.> -insertbegin <insbegdate/time> –insertend <insenddate/time> –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel8.log –physical ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script displays the number of granules for deletion, and prompts the user as to whether to continue. The script completes successfully.

2. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel8.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule insert time of the request, and that the deletion completed successfully.

3. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **Select * from DsMdGranules where ShortName = "<ShortName>" and VersionID = <VersionID> and insertTime < "<insenddate/time>" and insertTime > "<insbegdate/time>"**
- **go**
- **select * from DsMdDeletedGranules**

- **go**
- **select * from DsMdFileStorage where granuleId = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for DeleteEffectiveDate and a value of "N" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and the granule's original insertTime (from the DsMdGranules table) for the insertTime, and a DFA flag of 1.

Delete with ESDT Name/Version and by Granule Time and specify a physical delete, Granules are tagged for deletion

---

1. Perform a query in the **Database** workspace against the SDSRV database to determine which granules will be deleted when the Granule Deletion Client is run next:

- **select dbID from DsMdGranules where ShortName = "<ShortName>" and VersionID = <VersionID> and BeginningDateTime > "<granbegdate/time>" and EndingDateTime < "<granenddate/time>"**
- **go**

A list of dbIDs is returned. These will be referenced in the next few steps to compare with what the Granule Deletion Client deletes using the same search criteria.

This script runs to completion and the user is prompted with the number of granules to be deleted before confirming the delete.

2. On the **SDSRV** workspace, run the Granule Deletion Client specifying ESDT ShortName, ESDT version and granule time coverage:

- **EcDsGranuleDelete –name <ShortName> -version <version no.> -BeginDate <granbegdate/time> –EndDate <granenddate/time> –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel9.log –physical ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

3. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel9.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the

ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were deleted.

4. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select dbID from DsMdGranules where ShortName = "<ShortName>" and VersionID = <VersionID> and BeginningDateTime > "<granbegdate/time>" and EndingDateTime < "<granenddate/time>"**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdFileStorage where granuleId = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for DeleteEffectiveDate and a value of "N" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and a DFA flag of 0.

Delete with ESDT Name/Version and by Granule Time and specify a DFA, granules are taged for deletion

1 Perform a query in the **Database** workspace against the SDSRV database to determine which granules will be deleted when the Granule Deletion Client is run next:

- **select dbID from DsMdGranules where ShortName = "<ShortName>" and VersionID = <VersionID> and BeginningDateTime > "<granbegdate/time>" and EndingDateTime < "<granenddate/time>"**
- **go**

A list of dbIDs is returned. These will be referenced in the next few steps to compare with what the Granule Deletion Client deletes using the same search criteria.

2 On the **SDSRV** workspace, run the Granule Deletion Client specifying ESDT ShortName, ESDT version and granule time coverage:

- **EcDsGranuleDelete –name <ShortName> -version <version no.> -BeginDate <granbegdate/time> –EndDate <granenddate/time> –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel10.log –DFA ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script runs to completion and the user is prompted with the number of granules to be deleted before confirming the delete.

3. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel10.log**
- **<esc>**
- **:q**

- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were deleted.

4. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select dbID from DsMdGranules where ShortName = "<ShortName>" and VersionID = <VersionID> and BeginningDateTime > "<granbegdate/time>" and EndingDateTime < "<granenddate/time>"**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdFileStorage where granuleId = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of NULL for DeleteEffectiveDate and a value of "Y" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and a DFA flag of 1.

Run SDSRV Deletion Cleanup Utility

Run SDSRV Deletion Cleanup Utility to notify STMGT of desired deletions and to physically remove granules as requested.

1. In the SDSRV database window on the **Database** workspace, view the granule IDs that will be physically deleted:

- **select dbID, ShortName, VersionID, deleteEffectiveDate from DsMdGranules where deleteEffectiveDate < <Today-lagtime>**
- **go**
- **select * from DsMdDeletedGranules where transactionTime < <Today-lagtime>**

go

A list of dbIDs and ESDTs is returned. These are the granules that will be physically deleted when the Deletion Cleanup Utility is executed in the next few steps.

2. On the **SDSRV** host and in the SDSRV window, execute the Deletion Cleanup Utility, first not entering the lag time, and then changing the lag time entered:

- **cd /usr/ecs/<MODE>/CUSTOM/utilities**
- **ll**
- **EcDsDeletionCleanup.pl**
- Enter lag time in days: **<enter>**
- Enter lag time in days: **5**

- Confirm lag time: **N**
- Re-enter lag time: **2**
- Confirm lag time: **y**
- Enter mode of operation: **<MODE>**
- Enter log file name: **DelCleanup1.log**
- Enter Sybase User: **EcDsScienceDataServer**
- Enter Sybase User Password: **<password>**
- Enter sql server: **t1acg04_srvr**
- Enter DBName: **EcDsScienceDataServer1_<MODE>**
- Enter STMGT DBName: **stmtdb1_<MODE>**
- Enter Batch size: **3**

Do you wish to continue deleting these granules?: **y**

Execution of the Deletion Cleanup Utility begins and continues to prompt the user for the lag time until it is entered, and then executes unsuccessfully because the dbo userid was not specified.

a. View the Deletion Cleanup log file:

- **cd /usr/ecs/<MODE>/CUSTOM/logs**

vi DelCleanup1.log

There are a number of options to run concerning Lag Time such as:

- **Execute Deletion Cleanup Utility with Lag Time of a # of Days and deny the confirmation.**
- **Execute Deletion Cleanup Utility with Lag Time of a # of Days.**

b. Example: On the SDSRV host and in the SDSRV window, execute the Deletion Cleanup Utility, and confirm the lag time of two days:

- **cd /usr/ecs/<MODE>/CUSTOM/utilities**
- **EcDsDeletionCleanup.pl**
- Enter lag time in days: **2**
- Confirm lag time: **y**
- Enter mode of operation: **<MODE>**
- Enter log file name: **DelCleanup1.log**
- Enter Sybase User: **sdsrv_role**
- Enter Sybase User Password: **<password>**
- Enter sql server: **t1acg04_srvr**
- Enter DBName: **EcDsScienceDataServer1_<MODE>**
- Enter STMGT DBName: **stmgtdb1_<MODE>**

Enter Batch size: **3**

c. Execution of the Deletion Cleanup Utility begins and the number of DFA and physically deleted granules is returned with a confirmation prompt.

Confirm the deletion when prompted:

Do you wish to continue deleting these granules?: **y** Execution of the Deletion Cleanup Utility continues to completion.

3. In the SDSRV database window on the **Database** workspace, view the DsMdStagingTable:

- **select * from DsMdStagingTable**
- **go**
- **select * from DsMdStagingTable**
- **go**
- **select * from DsMdStagingTable**

go

The SDSRV Staging table has data in it that is being transferred to the STMGT database in increments of 3 (the specified batch size). When all data has been transferred to STMGT, the table is empty. All granules from the DeletedGranules table that were placed there before the lag time, are transferred, whether they are DFA or physical deletes.

4. In the STMGT database window of the **Database** workspace, view the DsStPendingDelete table:

- **select * from DsStPendingDelete**
- **go**
- **select * from DsStPendingDelete**
- **go**
- **select * from DsStPendingDelete**

go

The STMGT Pending Delete table receives data from the SDSRV in increments of 3 (the specified batch size). All the granules expected to be transferred to STMGT (all those specified in the pretest delete requests) have been transferred.

a. View the Deletion Cleanup log. Messages in the log file include the information about the above request, and they indicate that information has been placed in the STMGT database in groups of 3 (requested batch size).

b.• Execute the Deletion Cleanup Utility with a Lag Time of 0 is another option with a Batch size of 1.

Attempt a physical delete with a file containing a list of granule IDs and requesting no confirmation prompt. The granules are tagged for deletion

1. Perform a query in the **Database** workspace against the SDSRV database to determine which granules will be deleted when the Granule Deletion Client is run next:

- **select * from DsMdGranules where dbID in (xxx, yyy)**
- **go**

A list of granules is returned based on the supplied dbIDs. These will be referenced in the next few steps to compare with what the Granule Deletion Client deletes based on the input file.

2. On the **SDSRV** workspace, run the Granule Deletion Client specifying the input file containing SDSRV Granule Ids:

EcDsGranuleDelete –geoidfile dbids3.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel17.log –physical –noprompt ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>

The script runs to completion and the operator is not prompted to confirm the delete before it occurs.

3. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel17.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were tagged for deletion.

4. On the **Database** workspace, view the SDSRV database to verify the granules were tagged for deletion:

- **Select * from DsMdGranules where dbID in (xxx, yyy)**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdFileStorage where granuleId in (xxx, yyy)**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for DeleteEffectiveDate and a value of "N" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion and a DFA flag of 0.

Attempt deletion with a file containing a list of granule IDs and specify DFA. The Granules are tagged for deletion.

---

1. Perform a query in the Database workspace against the SDSRV database to determine which granules will be deleted when the Granule Deletion Client is run next:

- **select * from DsMdGranules where dbID in (xxx, yyy)**
- **go**

A list of granules is returned based on the supplied dbIDs. These will be referenced in the next few steps to compare with what the Granule Deletion Client deletes based on the input file.

2. On the **SDSRV** workspace, run the Granule Deletion Client specifying the input file containing SDSRV Granule Ids:

- **EcDsGranuleDelete –geoidfile dbids4.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel18.log –DFA ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script runs to completion and the operator is again prompted to confirm the delete before it occurs

3. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel18.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

4. View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were tagged for deletion.

5. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **Select * from DsMdGranules where dbID in (xxx, yyy)**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdFileStorage where granuleId in (xxx, yyy)**

go

The DsMdGranules table shows the granules are still in the database and they have a value of NULL for DeleteEffectiveDate and a value of "Y" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and a DFA flag of 1.

Attempt deletion with a file containing a list of local granule IDs specifying a DFA. Data is tagged for deletion.

---

1. Perform a query in the Database workspace against the SDSRV database to determine which granules will be deleted when the Granule Deletion Client is run next:

- **select dbID, LocalGranuleID from DsMdGranules where LocalGranuleID in ("aaa", "bbb", "ccc")**
- **go**

A list of granules is returned based on the supplied LocalGranuleIDs. These will be referenced in the next few steps to compare with what the Granule Deletion Client deletes based on the input file.

2. On the **SDSRV** workspace, run the Granule Deletion Client specifying the input file containing Local Granule Ids:

- **EcDsGranuleDelete –localgranulefile locgrn1.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel19.log –DFA ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script runs to completion and the operator is again prompted to confirm the delete before it occurs.

3. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel19.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were tagged for deletion.

4. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select dbID, LocalGranuleID, deleteEffectiveDate, DeleteFromArchive from DsMdGranules where LocalGranuleID in ("aaa", "bbb", "ccc")**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdFileStorage where granuleId = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of NULL for DeleteEffectiveDate and a value of "Y" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and a DFA flag of 1.

Attempt deletion with a file containing a list of local granule IDs specifying a physical delete

1. Perform a query in the Database workspace against the SDSRV database to determine which granules will be deleted when the Granule Deletion Client is run next:

- **select dbID, LocalGranuleID from DsMdGranules where LocalGranuleID in ("aaa", "bbb", "ccc")**
- **go**

A list of granules is returned based on the supplied LocalGranuleIds. These will be referenced in the next few steps to compare with what the Granule Deletion Client deletes based on the input file.

2. On the **SDSRV** workspace, run the Granule Deletion Client specifying the input file containing SDSRV Granule Ids:

- **EcDsGranuleDelete –localgranulefile locgrn4.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel22.log –physical ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script runs to completion and the operator is again prompted to confirm the delete before it occurs.

3. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel22.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

4. View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and time of the request. These messages also demonstrate that the requested granules were deleted.

5. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select dbID, LocalGranuleID, deleteEffectiveDate, DeleteFromArchive from DsMdGranules where LocalGranuleID in ("aaa" "bbb", "ccc")**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdFileStorage where granuleId in (dbid1, dbid2, dbid3)**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for the DeleteEffectiveDate column and a value of "N" for the DeleteFromArchive column. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and the granule's original insertTime (from the DsMdGranules table) for the insertTime, and a DFA flag of 0.

REPEAT AS BEFORE: Run SDSRV Deletion Cleanup Utility

to notify STMGT of desired deletions and to physically remove granules as requested. STMGT is notified of granules to delete and only the physically deleted granules are removed from the SDSRV DATABASE.

1. In the SDSRV database window on the **Database** workspace, view the granule IDs that will be physically deleted:

- **select dbID, ShortName, VersionID, deleteEffectiveDate from DsMdGranules where deleteEffectiveDate != NULL**

go

A list of dbIDs and ESDTs is returned. These are the granules that will be physically deleted when the Deletion Cleanup Utility is executed in the next few steps.

2. On the **SDSRV** workspace execute the Deletion Cleanup Utility, and confirm the lag time:

- **cd /usr/ecs/<MODE>/CUSTOM/utilities**
- **ll *Del***
- **EcDsDeletionCleanup.pl**
- Enter lag time in days: **0**
- Confirm lag time: **y**
- Enter mode of operation: **<MODE>**
- Enter log file name: **DelCleanup2.log**
- Enter Sybase User: **sdsrv_role**
- Enter Sybase User Password: **<password>**
- Enter sql server: **t1acg04_srvr**
- Enter DBName: **EcDsScienceDataServer1_<MODE>**
- Enter STMGT DBName: **stmgtdb1_<MODE>**
- Enter Batch size: **10**

Do you wish to continue deleting these granules?: **y**

The Deletion Cleanup Utility executes successfully, prompting the user with the number of granules to be DFA or physically deleted and whether or not to continue.

3. In the SDSRV database window on the **Database** workspace, view the DsMdStagingTable:

- **select * from DsMdStagingTable**
- **go**
- **select * from DsMdStagingTable**
- **go**
- **select * from DsMdStagingTable**

go

The SDSRV Staging table has data in it that is being transferred to the STMGT database in increments of 10 (the specified batch size). When all data has been transferred to STMGT, the table is empty. All granules from the DeletedGranules table are transferred (lag time is 'ignored' and all granules are deleted), whether they are DFA or physical deletes.

4. In the STMGT database window of the **Database** workspace, view the DsStPendingDelete table:

- **select * from DsStPendingDelete**
- **order by CreationTime**
- **go**
- **select * from DsStPendingDelete**
- **order by CreationTime**
- **go**
- **select * from DsStPendingDelete**
- **order by CreationTime**

go

The STMGT Pending Delete table receives data from the SDSRV in increments of 10 (the specified batch size). All the granules expected to be transferred to STMGT (all those specified in the above delete requests), have been transferred.

5. View the Deletion Cleanup log. Messages in the log file include the information about the above request, and they indicate that information has been placed in the STMGT database at the requested batch size.

6. In the **GUIs** workspace, view the STMGT GUI for the deletion requests just transferred to the STMGT DATABASE:

7. Select the **Delete** tab from the menu bar. The Delete screen displays number of files to be deleted for a given ESDT/Version ID pair (as noted in the above deletion requests).

8. In the SDSRV database window on the **Database** workspace, view the DsMdDeletedGranules table:

- **select * from DsMdDeletedGranules**

go

The SDSRV Deleted Granules table is again empty after all the granules have been deleted.

9. In the SDSRV database window on the **Database** workspace, view the DsMdGranules table and some additional tables to verify the correct granules where deleted:

- **select * from DsMdGranules where dbID = sss**
- **go**
- **select * from DsMdMeasuredParameter where granuleId = nnn (if applicable)**
- **go**
- **select * from DsMdFileStorage where granuleId = xxx**

go

The specified granules are not found in the various tables.

Note: where sss, nnn and xxx are dbIDs were just requested for a physical delete.

10. In the SDSRV database window on the **Database** workspace, view the DsMdGranules table and some additional tables to verify the correct granules where deleted:

- **select * from DsMdGranules where dbID = lll**
- **go**
- **select * from DsMdMeasuredParameter where granuleId = mmm (if applicable)**
- **go**
- **select * from DsMdFileStorage where granuleId = ooo**

go

The specified granules are found in the various tables.

Note: where lll, mmm and ooo are dbIDs just requested for DFA.

Attempt deletion using GranuleID filelist representing different ESDTs whose data resides in different archives and includes multi-file granules. Data is tagged for deletion.

---

1. In the **GUIs** workspace on the **STMGT GUI**, determine an ESDT that has data stored in multiple archives (an ESDT where the archive location has been changed, so it has a current path and previous path, and one where its data resides in a primary and a backup archive and where a granule has multiple files):

- select **VolGrpConfig tab**
- Click the **ESDT/Version** pair
- Click the **Display History** button

Record the **archive server(s)** and **paths**

The ESDTs archive servers and archive paths are found and recorded for use in the next few steps.

2. Perform a query in the **Database** workspace against the SDSRV database to determine which granules will be deleted when the Granule Deletion Client is run next:

- **select * from DsMdGranules where dbID in (aaa, bbb, ccc, ddd)**
- **go**

A list of granules is returned based on the supplied granule IDs. These will be referenced in the next few steps to compare with what the Granule Deletion Client deletes based on the input file.

3. On the **SDSRV** workspace, run the Granule Deletion Client specifying the input file containing SDSRV Granule Ids:

- **EcDsGranuleDelete –geoidfile dbids5.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel25.log –physical ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script runs to completion and the operator is again prompted with the number of granules for delete, and to confirm the delete before it occurs.

4. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel25.log**
- **<esc>**

- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were deleted.

5. On the **Database** workspace, view the SDSRV database to verify the granules were tagged for deletion:

- **select * from DsMdGranules where LocalGranuleId = aaa or LocalGranuleId = bbb or LocalGranuleId = ccc**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdFileStorage where granuleId = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for the DeleteEffectiveDate column and a value of "N" for the DeleteFromArchive column. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and the granule's original insertTime (from the DsMdGranules table) for the insertTime, and a DFA flag of 0.

Attempt DFA deletion of granules with Browse, QA and PH

1. Perform a query in the **Database** workspace against the SDSRV database to determine which Browse, QA and PH granules are associated with science granules:

- **select * from DsMdGranules a, DsMdBrowseGranuleXref b where a.dbID = b.granuleId and a.ShortName like "%<ESDTName>&"**
- **go**
- **select * from DsMdGranules a, DsMdQaGranuleXref b where a.dbID = b.granuleId and a.ShortName like "%<ESDTName>%"**
- **go**
- **select ShortName, dbID, processingHistoryId from DsMdGranules where processingHistoryId > 1 and ShortName like "%<ESDTName>%"**

go

A list of granules is returned for each query request. These granules will be referenced in the next few steps for use with deleting associated granules.

Browse/PH/QA granules not referenced DFA delete. Granules are not tagged for deletion.

2. On the **SDSRV** workspace, run the Granule Deletion Client specifying an input file containing SDSRV Granule Ids (these Granule Ids are granules that have at least one Browse, QA and PH granule associated with them and they're not referenced by any other granule):

- **EcDsGranuleDelete –geoidfile dbids6.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel26.log –DFA ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script runs to completion and the operator is again prompted to confirm the delete before it occurs.

3. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel26.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were not tagged for deletion.

4. On the **Database** workspace, view the SDSRV database to verify the granules were tagged for deletion:

- **select * from DsMdGranules where dbID = nnn or dbID = ooo or dbID = ppp**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdBrowse where dbID = qqq**
- **go**
- **select * from DsMdQaGranule where dbID = rrr**
- **go**
- **select * from DsMdProcessingHistory where dbID = sss**
- **go**
- **select * from DsMdGranules where dbID = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of NULL for DeleteEffectiveDate and a value of "Y" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion. The Browse, QA and PH granules associated with the deleted science granules are still found in their respective tables with no deletion tags associated with them.

Browse/PH/QA granules not referenced physical delete. Granules are tagged for deletion.

---

1. On the **SDSRV** workspace, run the Granule Deletion Client specifying an input file containing SDSRV Granule Ids (these Granule Ids are granules that have at least one Browse, QA and PH granule associated with them and they're not referenced by any other granule):

- **EcDsGranuleDelete –geoidfile dbids7.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel27.log –physical ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script runs to completion and the operator is again prompted with the number of granules for deletion, and to confirm the delete before it occurs.

2. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel27.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were deleted.

3. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select * from DsMdGranules where dbID = nnn**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdBrowse where dbID = qqq**
- **go**
- **select * from DsMdQaGranule where dbID = rrr**
- **go**
- **select * from DsMdProcessingHistory where dbID = sss**
- **go**
- **select * from DsMdFileStorage where granuleId = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for DeleteEffectiveDate and a value of “N” for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion (Science, Browse, QA and Processing History granules), including a current time for TransactionTime and the granule's original insertTime (from the associated Granules, Browse, QA or PH table) for the insertTime, and a DFA flag of 0. The Browse, QA and PH granules associated with the deleted science granules are not found in their respective tables.

Note: where ooo, ppp, qqq, rrr, sss are granules just requested for deletion.

Browse/PH/QA granules referenced DFA. Science Granules are tagged for deletion, but associated granules are not.

1. On the **SDSRV** workspace, run the Granule Deletion Client specifying an input file containing SDSRV Granule Ids (these Granule Ids are granules that have at least one Browse, QA and PH granule associated with them and they are referenced by other granules):

- **EcDsGranuleDelete ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE> –geoidfile dbids8.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel28.log –DFA**

y (when prompted)

The script runs to completion and the operator is again prompted with the number of granules for deletion and to confirm the delete before it occurs.

2. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel28.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were not tagged for deletion.

3. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select * from DsMdGranules where dbID = nnn**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdBrowse where dbID = qqq**
- **go**
- **select * from DsMdQaGranule where dbID = rrr**
- **go**
- **select * from DsMdProcessingHistory where dbID = sss**
- **go**
- **select * from DsMdGranules where dbID = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of NULL for DeleteEffectiveDate and a value of "Y" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and the granule's original insertTime (from the DsMdGranules table) for the insertTime, and a DFA flag of 1. The Browse, QA and PH granules associated with the deleted science granules are still found in their respective tables and no deletion tags are associated with them.

Browse/PH/QA granules referenced physical. Granules are not tagged for deletion.

(Deletes Science Granules and not Associated Granules)

1. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select * from DsMdGranules where dbID = nnn or dbID = ooo or dbID = ppp**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdBrowse where dbID = qqq**
- **go**
- **select * from DsMdQaGranule where dbID = rrr**
- **go**
- **select * from DsMdProcessingHistory where dbID = sss**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for DeleteEffectiveDate and a value of "N" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion, including a current time for TransactionTime and the granule's original insertTime (from the DsMdGranules table) for the insertTime, and a DFA flag of 0. The Browse, QA and PH granules associated with the deleted science granules are still found in their respective tables because they are referenced by other granules.

Note: where ooo, ppp, qqq, rrr, sss are granules just requested for deletion.

Attempt deletion for granules and suppress the deletion of the associated Browse, QA and PH. Granules are tagged for deletion but associated Browse, QA and PH granules are not.

1. Browse/PH/QA referenced physical delete -noassoc

2. On the **SDSRV** workspace, run the Granule Deletion Client specifying an input file containing SDSRV Granule Ids (these Granule Ids are granules that have at least one Browse, QA and PH granule associated with them and they are referenced by other granules):

- **EcDsGranuleDelete ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE> –geoidfile dbids10.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel30.log –physical -noassoc**

y (when prompted)

The script runs to completion and the operator is again prompted with the number of granules for deletion and to confirm the delete before it occurs.

3. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel30.log**

- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were deleted.

4. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select * from DsMdGranules where dbID = nnn or dbID = ooo or dbID = ppp**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdBrowse where dbID = qqq**
- **go**
- **select * from DsMdQaGranule where dbID = rrr**
- **go**
- **select * from DsMdProcessingHistory where dbID = sss**
- **go**
- **select * from DsMdFileStorage where granuleId = nnn**
- **go**
- **select * from DsMdGranules where dbID = qqq**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for the deleteEffectiveDate column and a value of "N" for the DeleteFromArchive column The DsMdDeletedGranules table contains the granules just requested for deletion. The Browse, QA and PH granules associated with the deleted science granules are still found in their respective tables and have no deletion times associated with them.

Note: where ooo, ppp, qqq, rrr, sss are granules just requested for deletion.

Browse/PH/QA not referenced physical delete -noassoc

1. On the **SDSRV** workspace, run the Granule Deletion Client specifying an input file containing SDSRV Granule Ids (these Granule Ids are granules that have at least one Browse, QA and PH granule associated with them and they are not referenced by other granules):

- **EcDsGranuleDelete ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE> –geoidfile dbids11.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel31.log –physical -noassoc**

y (when prompted)

The script runs to completion and the operator is again prompted to confirm the delete before it occurs.

2. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel31.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were deleted.

3. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select * from DsMdGranules where dbID = nnn**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdBrowse where dbID = qqq**
- **go**
- **select * from DsMdQaGranule where dbID = rrr**
- **go**
- **select * from DsMdProcessingHistory where dbID = sss**
- **go**
- **select * from DsMdFileStorage where granuleId = nnn**
- **go**
- **select * from DsMdGranules where dbID = qqq**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for DeleteEffectiveDate and a value of "N" for DeleteFromArchive. The DsMdDeletedGranules table contains the granules just requested for deletion. The Browse, QA and PH granules associated with the deleted science granules are still found in their respective tables and have no deletion times associated with them.

Determine granules that are still being referenced as input by other granules. Attempt a physical deletion of granules still being referenced as inputs by other granules. Granules are not tagged for deletion but are listed.

---

1. In the **SDSRV** database window on the **Database** workspace, view the DsMdInputGranule table to determine which granules have pointers to other granules:

- **select granuleId, inputGranule, InputPointer from DsMdInputGranule where InputPointer like "%L7IGS%"**

go

Several granules that are inputs to other granules in the database are noted. Some of these granules are linked via UR and others are linked via LGID as noted in the query results.

2. On the **SDSRV** workspace, run the Granule Deletion Client specifying the input file containing SDSRV Granule Ids:

- **EcDsGranuleDelete –geoidfile dbids40.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel40.log –physical ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script runs to completion after prompting the user with the number of granules to be deleted.

3. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel40.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request, and the nature of the error encountered. These messages also demonstrate that the requested granules were not tagged for deletion because they have associated input granules.

4. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **Select * from DsMdGranules where dbID = www dbID = xxx**
- **go**
- **select * from DsMdDeletedGranules**

go

The DsMdGranules table shows the granules are still in the database and they are not tagged for deletion. The DsMdDeletedGranules table does not contain the granules just requested for deletion.

Attempt a DFA delete of granules still being referenced as inputs by other granules. Granules are not tagged for deletion.

1. On the **SDSRV** workspace, using granules from the above database query, run the Granule Deletion Client specifying the input file containing SDSRV Granule Ids:

EcDsGranuleDelete –geoidfile dbids41.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel41.log –DFA –noprompt ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>

The script runs to completion after displaying to the user the number of granules to be deleted

2. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel41.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request, and the nature of the error received. These messages also demonstrate that the requested granules were tagged for deletion.

3. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **Select * from DsMdGranules where dbID = www or dbID = xxx**
- **go**
- **select * from DsMdDeletedGranules**

go

The DsMdGranules table shows the granules are still in the database and they are tagged for deletion (contain a "Y" in the DeleteFromArchive field). The DsMdDeletedGranules table contains the granule just requested for deletion.

Where www and xxx are granule IDs just requested for deletion.

Attempt a physical delete of granules still being referenced as inputs by other granules and override the check. Granules are tagged for deletion.

---

1. On the **SDSRV** workspace, using granules from the above database query, run the Granule Deletion Client specifying the input file containing SDSRV Granule Ids:

- **EcDsGranuleDelete –geoidfile dbids43.in –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel43.log –physical –delref ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script runs to completion after prompting the user with the number of granules to be deleted.

2. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel43.log**
- **<esc>**
- **:q**
- **View the ScienceDataServer log tails**

View the EcDsGranuleDelete ALOG

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed. Included in these messages are the user id of the requester, the ShortName, VersionID and granule time of the request. These messages also demonstrate that the requested granules were tagged for deletion, even though they're referenced by other granules.

3. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **select * from DsMdGranules where dbID = yyy dbID = zzz**
- **go**
- **select * from DsMdDeletedGranules**
- **go**
- **select * from DsMdFileStorage where granuleId = nnn**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for the DeleteEffectiveDate and a value of "Y" for the DeleteFromArchive flag. The DsMdDeletedGranules table contains the granules just requested for deletion.

Delete files from the archive using the STMGT GUI

Single ESDT/single path deletion

1. In the **GUIs** workspace, view the ESDTs with granules targeted for deletion via the STMGT GUI:

- select **Delete** from the menu bar

All the granules tagged for deletion should be listed as ready for deletion.

2. From the list of ESDTs displayed on the delete screen, determine where a given ESDTs data is stored in the archive:

- select **Close** button
- select **VolGrpConfig tab**
- Click the **ESDT/Version pair**

Record the **archive server** and **path**

The ESDTs archive server and archive path are found and recorded for use in the next few steps.

ESDT Archive/path:

_____

_____

3. Select data for deletion:

- select **Delete** from the menu bar
- Click an **ESDT/Version pair**

Click the **Reset** button

The ESDT/Version pair is selected for deletion, and then the screen is reset, and no ESDTs are selected for deletion.

4. Select data for deletion:

- Click an **ESDT/Version pair**

Click the **Delete** button

The ESDT/Version pair is selected for deletion, the delete is initiated and the operator is prompted to confirm the delete.

5. Confirm the delete when prompted:

Click on **OK** button when prompted

The delete request continues to completion.

6.• In the **STMGT** workspace, view the archive server log tails. The logs contain messages pertaining to the data deletion.

7. When the Delete Request is complete, view the archive path to verify the data was deleted in the **STMGT** workspace:

- **cd /dss_stk3/<MODE>/<dir>**

ll *<dbID>*

All requested data for the given ESDT has been deleted.

8. In the **GUIs** workspace, verify the ESDT/Version just deleted is no longer displayed on the delete page:

Select **Delete** from the menu bar

The ESDT/Version pair is not listed on the Delete screen as one of those ready to be deleted.

9. In the **Database** workspace, view the Pending Delete table to verify it no longer displays the data type just deleted:

- **select * from DsStPendingDelete**
- **go**

The database table listing does not include the ESDT/Version just deleted.

Single ESDT/Multiple archive and path deletion

1. In the **GUIs** workspace, view the ESDTs with granules targeted for deletion via the STMGT GUI:

select **Delete** from the menu bar

Granules tagged for deletion are listed as ready for deletion.

2. From the list of ESDTs displayed on the delete screen, determine an ESDT that has data stored in multiple archives and was used in verification of criterion 1706.11 above (an ESDT

where the archive location has been changed, so it has a current path and a previous path, and one who's data resides in a primary and a backup archive and has multiple files):

- Click **Ok** button
- select **VolGrpConfig tab (if necessary)**
- Click the **ESDT/Version pair**

Record the **archive server(s)** and **paths**

The ESDTs archive servers and archive paths are found and recorded for use in the next few steps.

ESDT/Version:

_____

_____

Archives/paths:

_____

_____

3. In the **STMGT** workspace, view the paths identified above to verify data for the ESDT is archived there:

- **cd /dss_stk3/<MODE>/<dir>**
- **ls -altr *<dbID>***

on other archive window:

- **cd /dss_stk3/<MODE>/<dir>**

ls -altr *<dbID>*

Data for the given data type resides in both archives.

4. In the **GUIs** workspace on the STMGT GUI, select the ESDT with data stored in different archives for deletion:

- Select **Delete** from the Menu Bar
- Click the **ESDT/Version pair**
- Click the **Delete** button

Click the **OK** button when prompted

The ESDT/Version pair is selected for deletion, the delete is initiated, and the operator is prompted to confirm the delete.

In the STMGT workspace, view the archive server log tails for both archive servers. The logs contain messages pertaining to the data deletion.

---

1. When the Delete Request is complete, view the archive paths to verify the data was deleted in the **STMGT** workspace:

- **cd /dss_stk3/<MODE>/<dir1>**
- **ll** *<dbID>*
- **cd /dss_stk3/<MODE>/<dir2>**

ll *<dbID>*

The requested data for the given ESDT has been deleted.

2. In the **GUIs** workspace, verify the ESDT/Version just deleted is no longer displayed on the delete page:

Select **Delete** from the menu bar

The ESDT/Version pair is not listed on the Delete screen as one of those ready to be deleted.

3. In the **Database** workspace, view the Pending Delete table to verify it no longer displays the data type just deleted:

- **select distinct VersionedDataType from DsStPendingDelete**
- **go**

4. The database table listing does not include the ESDT/Version just deleted.

- On the **GUIs** workspace in the STMGT GUI, select the ESDTs with primary/backup data for deletion: Click the **ESDT/Version pair**
- Click the **Delete** button

  Click the **OK** button when prompted

5.• The ESDT/Version pairs are selected for deletion, the delete is initiated and the operator is prompted to confirm the delete.

6. In the **STMGT** workspace, view the archive server log tails for both archive servers.

The logs contain messages pertaining to the data deletion.

7. When the Delete Request is complete, view the archive paths to verify the data was deleted in the **STMGT** workspace:

- **cd /dss_stk3/<MODE>/<dir1>**
- **ll** *<ESDT.Version>*
- **cd /dss_stk3/<MODE>/<dir2>**

ll *<ESDT.Version>*

All requested data for the given ESDT has been deleted, but the backup data still resides in the archive.

8. In the **GUIs** workspace, verify the ESDT/Version just deleted is no longer displayed on the delete page:

Select **Delete** from the menu bar.

The ESDT/Version pair is not listed on the Delete screen as one of those ready to be deleted.

In the **Database** workspace, view the Pending Delete table to verify it no longer displays the data type just deleted:

- **select * from DsStPendingDelete**
- **go**

The database table listing does not include the ESDT/Version just deleted.

Single ESDT/Multiple files

1. On the **GUIs** workspace in the STMGT GUI, select an ESDT with multiple files for deletion:
- Click the **ESDT/Version pair**
- Click the **Delete** button

Click the **OK** button when prompted

2.• The ESDT/Version pair is selected for deletion, the delete is initiated and the operator is prompted to confirm the delete.

3. In the **STMGT** workspace, view the archive server log tail for the archive server.

The logs contain messages pertaining to the data deletion.

4. When the Delete Request is complete, view the archive paths to verify the data was deleted in the **STMGT** workspace:
- **cd /dss_stk3/<MODE>/<dir1>**

ll *<dbID>*

Check that all data files for the requested ESDT have been deleted.

5. In the **GUIs** workspace, verify the ESDT/Version just deleted is no longer displayed on the delete page:

Select **Delete** from the menu bar

The ESDT/Version pair is not listed on the Delete screen as one of those ready to be deleted.

6. In the **Database** workspace, view the Pending Delete table to verify it no longer displays the data type just deleted:
- **select * from DsStPendingDelete**
- **go**

Check that the database table listing does not include the ESDT/Version just deleted.

Multiple ESDT/Multiple archive and path deletion

1. In the **GUIs** workspace, view the ESDTs with granules targeted for deletion via the STMGT GUI:

a. select **Delete** from the menu bar

Granules tagged for deletion are listed as ready for deletion.

b. From the list of ESDTs displayed on the delete screen, determine two ESDTs that have data stored in multiple archives:

- select **VolGrpConfig tab**
- Click the first **ESDT/Version pair**
- Record the **ESDT, archive server(s)** and **paths**
- Click the second **ESDT/Version pair**

c. Record the **ESDT, archive server(s)** and **paths**

The ESDTs archive servers and archive paths are found and recorded for use in the next few steps.

ESDT/Version:

_____

_____

Archives/paths:

_____

_____

2. From the list of ESDTs displayed on the delete screen, determine two ESDTs that have data stored in multiple archives:

- select **VolGrpConfig tab**
- Click the first **ESDT/Version pair**
- Record the **ESDT, archive server(s)** and **paths**
- Click the second **ESDT/Version pair**

a. Record the **ESDT, archive server(s)** and **paths**

The ESDTs archive servers and archive paths are found and recorded for use in the next few steps.

3. In the **STMGT** workspace, view the paths identified above to verify data for the ESDT is archived there:

- **cd /dss_stk3/<MODE>/<dir>**
- **ls -altr *<dbID>***

on other archive window:

- **cd /dss_stk3/<MODE>/<dir>**

ls -altr *<dbID>*

Data for the given data type resides in both archives.

4. In the **GUIs** workspace on the STMGT GUI, select the ESDTs just identified for deletion:

- Click the first **ESDT/Version pair**
- Hold down the **ctrl key** and click the second **ESDT/Version pair**
- Click the **Delete** button

a. Click the **OK** button when prompted

The two ESDT/Version pairs are selected for deletion, the delete is initiated and the operator is prompted to confirm the delete.

5. In the **STMGT** workspace, view the archive server log tails for both archive servers.

The logs contain messages pertaining to the data deletion.

6. When the Delete Request is complete, view the archive paths to verify the data was deleted in the **STMGT** workspace:

- **cd <path/dir>**
- **ll** *<dbID>*
- **cd <path/dir>**

ll *<dbID>*

Requested data for the given ESDT has been deleted.

7. In the **GUIs** workspace, verify the ESDT/Version just deleted is no longer displayed on the delete page:

a. Select **Delete** from the menu bar.

The ESDT/Version pairs are not listed on the Delete screen as several of those ready to be deleted.

8. From the list of ESDTs displayed on the delete screen, determine an ESDT that has data stored in primary and backup archives:

- select **VolGrpConfig tab**
- Click the first **ESDT/Version pair**
- Record the **ESDT, archive server(s)** and **paths**
- Click the second **ESDT/Version pair**
- Record the **ESDT, archive server(s)** and **paths**

a. The ESDTs archive servers and archive paths are found and recorded for use in the next few steps.

ESDT/Version:

_____

_____

Archives/paths:

_____

_____

9. In the **STMGT** workspace, view the paths identified above to verify data for the ESDT is archived there:

- **cd <path/dir>**
- **ls -altr *<dbID>***

on other archive window:

- **cd <path/dir>**

ls -altr *<dbID>*

Data for the given data type resides in both archives.

10. In the **GUIs** workspace on the STMGT GUI, select the ESDTs just identified for deletion:

- Click the first **ESDT/Version pair**
- Hold down the **ctrl key** and click the second **ESDT/Version pair**
- Click the **Delete** button

a. Click the **OK** button when prompted.

The two ESDT/Version pairs are selected for deletion, the delete is initiated and the operator is prompted to confirm the delete.

Note: the ESDT/Version pair will be the same, but the backup will be suffixed with a 'B'

11. In the **STMGT** workspace, view the archive server log tails for the appropriate archive servers.

The logs contain messages pertaining to the data deletion.

12. When the Delete Request is complete, view the archive paths to verify the data was deleted in the **STMGT** workspace:

- **cd path1/dir1>**
- **ll *<dbID>***
- **cd <path2/dir2>**

ll *<dbID>*

Requested primary and backup data for the given ESDT has been deleted.

13. In the **GUIs** workspace, verify the ESDT/Versions just deleted are no longer displayed on the delete page:

- Select **Delete** from the menu bar

The ESDT/Version pairs are not listed on the Delete screen as one of those ready to be deleted.

14. In the **Database** workspace, view the Pending Delete table to verify it no longer displays the data type just deleted:

- **select * from DsStPendingDelete**
- **go**
- **select FileName from DsStFile where FileName like "%dbID%"**
- **go**

The database table listing does not include the ESDT/Versions just deleted.

Physically delete all files on an archive tape volume and recycle the tape

(MASS ADMIN needed for this scenario)

---

1. On the **SDSRV** workspace, run the Granule Deletion Client specifying ESDT ShortName, ESDT version and granule insert time coverage:

- **EcDsGranuleDelete –name <ShortName> -version <version no.> -insertbegin <insbegdate/time> –insertend <insenddate/time> –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel47.log –physical ConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**

y (when prompted)

The script is executed and the user is prompted with the number of granules for deletion.

Times Used:

_____

2. View the Deletion log file and the tail of the SDSRV ALOG file:

- **vi GranDel47.log**
- **<esc>**
- **:q**

a. View the tail of the ScienceDataServer ALOG file

The Deletion log and the Science Data Server ALOG file display messages related to the deletion just performed.

3. On the **Database** workspace, view the SDSRV database to verify the granules were deleted:

- **Select deleteEffectiveDate, DeleteFromArchive, dbID from DsMdGranules where ShortName = "<ShortName>" and VersionID = <versionid>**
- **go**
- **select * from DsMdDeletedGranules**

go

The DsMdGranules table shows the granules are still in the database and they have a value of current time for the DeleteEffectiveDate column and a value of "N" for the DeleteFromArchive column. The DsMdDeletedGranules table contains the granules just requested for deletion.

4. On the **SDSRV** workspace, execute the Deletion Cleanup Utility, and confirm the lag time:

- **cd /usr/ecs/<MODE>/CUSTOM/utilities**
- **ll *Del***
- **EcDsDeletionCleanup.pl**
- Enter lag time in days: **0**
- Confirm lag time: **y**
- Enter mode of operation: **<MODE>**
- Enter log file name: **<ent>**
- Enter Sybase User: **sdsrv_role**
- Enter Sybase User Password: **<password>**
- Enter sql server: **t1acg04_srvr**
- Enter DBName: **EcDsScienceDataServer1_<MODE>**
- Enter STMGT DBName: **stmgtdb1_<MODE>**
- Enter Batch size: **3**

Do you wish to continue deleting these granules?: **y**

The Deletion Cleanup Utility executes and the number of DFA and physically deleted granules is returned with a confirmation prompt. After confirming the deletion, the script completes. The number of granules corresponds to the number of granules that are on a given tape in the archive as noted during pretest activities.

a. View the Deletion Cleanup log. Messages in the log file include the information about the above request, and they indicate that information has been placed in the STMGT database.

b. In the **GUIs** workspace, view the STMGT GUI for the deletion requests just transferred to the STMGT DATABASE:

c. Select the **Delete** tab from the menu bar.

The Delete screen displays number of files to be deleted for a given ESDT/Version ID pair (as noted in the above deletion requests), and the number of files compares favorably with the number deleted (as noted in the above deletion requests).

On the **GUIs** workspace in the STMGT GUI, select for deletion the ESDT just deleted in SDSRV:

- Click the **ESDT/Version pair**
- Click the **Delete** button

Click the **OK** button when prompted.

The ESDT/Version pair is selected for deletion, the delete is initiated and the operator is prompted to confirm the delete.

5. In the **STMGT** workspace, view the archive server log tail for the appropriate archive server.

The logs contain messages pertaining to the data deletion.

a. When the Delete Request is complete, view the archive paths to verify the data was deleted in the **STMGT** workspace:

- **cd <path/dir1>**
- **ll *<ESDT.Version>***

All requested data for the given ESDT has been deleted.

b. In the **GUIs** workspace, verify the ESDT/Version just deleted is no longer displayed on the delete page:

c. Select **Delete** from the menu bar.

The ESDT/Version pair is not listed on the Delete screen as one of those ready to be deleted.

Note: Have the AMASS Administrator login to amass and determine if all the files have been removed from the amass archive tape identified earlier.

No files are found on the tape. **AMASS Administrator needed to perform this step.**

d. Have the AMASS Administrator perform any steps needed to recycle the tape and put the tape back in circulation for the same ESDT. The tape is recycled and ready for use.

Run Deletion Cleanup Utility on L7 Granule Deletion Output

Specifying both DFA and physical deletes.

---

1. On the SDSRV host and in the SDSRV window, execute the Deletion Cleanup Utility, and confirm the lag time:

- **cd /usr/ecs/<MODE>/CUSTOM/utilities**
- **ll**
- **EcDsDeletionCleanup.pl**
- Enter lag time in days: **0**

- Confirm lag time: **y**
- Enter mode of operation: **<MODE>**
- Enter log file name: **DelCleanup75.log**
- Enter Sybase User: **sdsrv_role**
- Enter Sybase User Password: **<password>**
- Enter sql server: **t1acg04_srvr**
- Enter DBName: **EcDsScienceDataServer1_<MODE>**
- Enter STMGT DBName: **stmgtdb1_<MODE>**
- Enter Batch size: **3**

Do you wish to continue deleting these granules?: **y**

2. Execution of the Deletion Cleanup Utility begins and the number of DFA and physically deleted granules is returned with a confirmation prompt. The script completes after confirming the delete.

3. View the Deletion Cleanup log tail. Execution of the Deletion Cleanup Utility begins and the number of DFA and physically deleted granules is returned with a confirmation prompt. The script completes after confirming the delete.

4. In the **GUIs** workspace, view the ESDTs with granules targeted for deletion via the STMGT GUI:

- select **Delete** from the menu bar.

5. Check to see if all the granules tagged for deletion are listed as ready for deletion.

6. From the list of ESDTs displayed on the delete screen, determine where a given ESDTs data is stored in the archive:

- select **VolGrpConfig tab**
- Click the **ESDT/Version pair**

a. Record the **archive server** and **path**

7. The ESDTs archive server and archive path are found and recorded for use in the next few steps.

a. Select data for deletion:

- Click an **ESDT/Version pair**
- Click the **Delete** button

b. Click the **OK** button when prompted

8.• The ESDT/Version pair is selected for deletion, the delete is initiated and the operator is prompted to confirm the delete.

9. In the **STMGT** workspace, view the archive server log tails.

The logs contain messages pertaining to the data deletion, which is in progress.

10. When the Delete Request is complete, view the archive path to verify the data was deleted in the **STMGT** workspace:

- **cd /dss_stk3/<MODE>/<dir>**

ll *<dbID>*

Setup a Cron Job to run the Granule Deletion Client. (Operational execution verification - not in criteria)

---

1. On the **SDSRV** workspace, create a cron job for the Granule Deletion Client:

- **cd /usr/ecs/<MODE>/CUSTOM/utilities**
- **vi delfiles**
- **0,5,10,15,20,25,30,35,40,45,50,55 * * * * /usr/ecs/<MODE>/CUSTOM/bin/DSS/EcDsGranuleDelete –name TEST01 -version 001 -insertbegin <insbegdate/time> –insertend 12/31/2001 –log /usr/ecs/<MODE>/CUSTOM/logs/GranDel44.log –physicalConfigFile /usr/ecs/<MODE>/CUSTOM/cfg/EcDsGranuleDelete.CFG ecs_mode <MODE>**
-
- **esc**
- **wq!**
- **crontab delfiles**

crontab -l

2. Cron job is set up and is executed when specified in the files.

Note: A script may need to be written that will source the needed environment before the deletion script can run as a cron job. This Granule Deletion Client runs every 5 minutes starting on the hour.

3. Tail the deletion logs in the **SDSRV** workspace, so the logs can be viewed as the deletion scripts execute (via cron):

- **tail -f GranDel44.log**

tail -f DelCleanup.log

The log tails show the deletion scripts are running at the intervals in the cron file.

4. On the **Database** workspace, view the granules in the SDSRV database to verify they are being tagged for deletion and deleted according to the cron file:

- **Select dbID, ShortName, VersionID, deleteEffectiveDate, DeleteFromArchive from DsMdGranules where deleteEffectiveDate != NULL**
- **go**
- **select * from DsMdDeletedGranules**

go

The granules are either marked for deletion, or deleted from the database, depending on which cron jobs have completed.

a. On the **GUIs** workspace, view the STMGT GUI:

- select **Delete** from the menu bar

Note: Perioductly, repeat this process until all the cron jobs have executed.

Post Granule Deletion Actions

1. On the **GUIs** workspace, terminate each of the GUIs (DDIST, STMGT, SDSRV, MSS):

- Select **File/Close**

Each of the GUIs are terminated.

On the appropriate workspaces, stop the log tails:

ctrl c

2. Delete Granule Deletion log files after any necessary analysis has completed:

- **cd /usr/ecs/<MODE>/CUSTOM/logs**
- **ll *GranDel***
- **rm *GranDel***
- **ll *DelClea***
- **rm *DelClea***

3. Exit each of the xterms brought up in each of the workspaces.

Appendix B.

Learn more about SSI & T

References:

MISR Science Data Processing Software Test Plan, Volume 2, Detailed Procedures and Facilities Version 2.0, Part 1 (PGE 1) June 1998.

http://m0mss01.ecs.nasa.gov/smc/dc_master.html

URL for ECS Project Training Material Volume 16: SSI&T December 1997:

http://dmserver.gsfc.nasa.gov/relb_it/relbit.htm

SV DOC

REPOSITORY

Home

Drop Build Plans

Acceptance Test Plan

System Verification Test Plan

Access Database

Release B Testdata

Site Install and Checkout Test

End To End Test Procedures

Goddard DAAC M&O Status

VATC Status Page

ECS TEST

PAGES

Advertising Service (VATC)

User Registration Tool

(VATC)

V0 Web Client (VATC TS2)

V0 Web Client (DAAC

MODE TS2)

V0 Web Client (DAAC

MODE TS1)

V0 Web Client (DAAC

MODE OPS)

ECS TOOLS

EP7

RTM

CCR

EDHS

DDTS

Network Status Page

ECS Newsroom Server

Configuration Management

Release B Integration

ECS Telephone & Email Dir

ISO 9001

Business Manual Tab

Job Description Tab

Organization Charts Tab

Process Directives Tab

Training Tab

Miscellaneous Tab

Frequently Asked questions

EDHS

Raytheon Company

ESDT Basics - **http://dmserver.gsfc.nasa.gov/esdt/EsdtSection1/index.html**

Science Office Instructions:
http://dmserver.gsfc.nasa.gov/proj_instr/sopi_index.html

GDAAC Directory for SSI&T:http://gsfcsrvr8.gsfcmo.ecs.nasa.gov/SSIT/

The ESDT Process (updated for drop 4) by Karl W. Cox, 22 December 1997

MODIS - Science Data Processing Software Version 2 System Description, SDST-104, May 19, 1998 a**t http://ltpwww.gsfc.nasa.gov/MODIS/SDST/docs.html**

ECSINFO: **http://ecsinfo.hitc.com/iteams/Science/science.html**

PDPS how to are located on the EDF machines at: **/home/PDPS/docs/**

PDPS Web Page: **http//dmserver.gsfc,nasa.gov/ecsdev/relb/pdps/index.html**

For Troubleshooting or use the following EDF machines:

PDPS Troubleshooting Techniques are located on the EDF machines at **:**
/home/PDPS/troubleshooting/

DPREP README files located at **:/usr/ecs/TS1/CUSTOM/data/DPS/**

DPREP binary located: **:/usr/ecs/TS1/CUSTOM/bin/DPS/**

Updating the Leap Seconds and the Earth Motions files

The toolkit requires Leap Second and Earth Motion updates, weekly and twice weekly respectively, to accurately compute most time conversions. The following scripts have been established to accomplish these tasks as part of ECS support.

- **update_leapsec.sh**

This script updates the leapsec.dat file by ftp-ing to USNO and reformatting the information into the leap seconds file: $PGSHOME/database/common/TD/leapsec.dat
The present script, after obtaining the required file "tai-utc.dat" in the same Series 7 mentioned above, invokes PGS_TD_NewLeap, a C program that performs the actual update work. The function puts the current date in the header of the new leapsec.dat, with a remark that the file was either "Checked" (no new leap second) or "Updated" (new leap second). The date at which the USNO file used in the updating process was put on their server is also listed in the header.

- **update_utcpole.sh**

This script updates the **utcpole.dat** file on the basis of new data obtained by ftp to the U.S. Naval Observatory in Washington, D.C (USNO). Their data file is excerpted and the required fields are reformatted and written into the utcpole file: $PGSHOME/database/common/CSC/utcpole.dat

- **The Leap Seconds file:**

leapsec - file ID: **$PGSHOME/database/common/TD/leapsec.dat**

(Atomic time from International Earth Rotation Service)

Introduced every 12 to 24 months, announced almost 6 months in advance or as little as 90 days notice. Update available from U.S Navy Observatory (USNO).

Interval of update recommend: weekly, except Sundays 17:45 hours to 17:55 Eastern US time. Runtime is approximately 30 seconds**.**

- **The Earth Motion file:**

utcpole – file ID: **$PGSHOME/database/common/CSC/utcpole.dat**

(Record of the Earth's variable of slowing rotation with respect to UTC Time.) **Interval of update recommended: Twice weekly except Sundays 17:45 hours to 17:55 Eastern US time. Recommended scripts be run in the afternoon or evening each Tuesday and Thursday.**

Script Name: update_leapsec.sh

The following processing tasks are carried out automatically by the use of this script:

- **Update via: Ftp to USNO, "maia.usno.navy.mil"** file accessed for leapsec**: tai-utc.dat. (Tests connectivity by using "ping')**
- **Function to be applied: PGS_TD_NewLeap,** excerpts and reformats the new information and appends new data and date to **leapsec.dat** file. A remark that the file was either "Checked" (no new Leap second) or "Updated" (new leap second). 11.2.2 Script Name: update_utcpole.sh

The following processing tasks are carried out automatically by the use of this script

- **Update via: Ftp to USNO, "maia.usno.navy.mil"** file accessed for utcpole**: finals.data. (Tests connectivity by using "ping')**
  Function to be applied: PGS_CSC_UT1_update, excerpts and reformats the new information and appends new data to **utcpole.dat** file.

Guidelines:

1• The script must be run on a machine that has the Toolkit mounted and which can access the USNO site via ftp and access e-mail. (p0spg01 used at the minidaac)

2• For each installed Toolkit (including all modes, such as debug, F77, F90, etc.) the scripts need to be run only once, even if different platforms or operating systems are run. However, if entirely separate Toolkits exist at your installation, with different $PGSHOME home directories, then either the scripts need to be run in each, or the data files can be propagated from a primary Toolkit to the others.

3• It is highly desirable to have outgoing e-mail mounted on the machine of choice, so that error messages may be issued automatically from the scripts in case of failure.

4• If the updating process fails, then the script must be rerun. The Toolkit team should be contacted anytime the scripts are not giving the correct or accurate It is highly desirable to have outgoing e-mail mounted on the machine of choice, information. The 2 sets of scripts do also send an email message to SDP Toolkit mail address when a script fails

5• The Toolkit requires that the two data files not be too stale. Therefore the useful lifetime of the utcpole.dat and leapsec.dat files is 83 days. The Toolkit will issue an error message if no update was performed beyond 83 days. If this occurs you can expect geolocation accuracy to deteriorate to an extent that could require re-running for some of

the more stringent users. If Toolkit requires a leap second value after this date, an error message will be returned. This generally means that production will cease.

6• Keep the Latest files until your updates are completed! They are useful for a backup should they be needed.

Hardware Needed and Setup Procedures

The user's environment needs to be set up by running the script $PGSBIN/pgs-dev-env.csh or $PGSBIN/pgs-dev-env.ksh, depending on the shell being used. $PGSBIN stands for $PGSHOME/bin/mach, where "mach" stands for one of: sun5, sgi64, sgi, sgi32, ibm, dec, or hp. In other words it is a shorthand for the machine "flavor" you are using, and for sgi, the compiler option. Not all versions are necessarily at each DAAC or SCF, and in some cases the path may be more complicated. For example, at Goddard Space Flight Center DAAC, typical binary directories are /usr/ecs/OPS/CUSTOM/TOOLKIT/toolkit/bin/sgi64_daac_f77/, or **/usr/ecs/OPS/CUSTOM/TOOLKIT/toolkit/bin/sgi64_daac_f90_debug/, for example**.
Once the setup script is located and sourced, $PGSBIN is defined and your path includes it. Furthermore, a "PCF", or process control file, $PGS_PC_INFO_FILE is defined, which allows the executable functions invoked by the scripts to find the old data files, which are needed for the updates.

To run the scripts successfully, you must have write permission on the data files.

After the setup is done, just run the scripts. Both scripts (update_utcpole.sh and update_leapsec.sh) are located in the directory $PGSBIN, which will be in your path after the Setup script has been run.

To Update the Latest Leapsec.dat and Utcpole.dat files perform the following steps

1 telnet to a machine that supports the Toolkit. (**telnet p0spg01**)
2 login: ID, Password:
3 **setenv DISPLAY ….:0.0**
4 **setenv PGSHOME /usr/ecs/OPS/CUSTOM/TOOLKIT/toolkit**
5 **cd /usr/ecs/OPS/CUSTOM/TOOLKIT/toolkit/bin/sgi_daac_f77** then
6 **source pgs_dev-env.csh**
7 For **leapsec: cd /usr/ecs/OPS/CUSTOM/TOOLKIT/toolkit/database/common/TD**
8 **cp leapsec.dat leapsec.dat_old**
9 Know thread for Leap Second run:
10 **cd /usr/ecs/OPS/CUSTOM/TOOLKIT/toolkit/src/TD** then do **ls** – select: update_leapsec.sh or run script for Leap Second type in: **update_leapsec.sh**

11 A successful update will look like the following:

p0spg01{cmops}[288]->update_leapsec.sh
Status of PGS_TD_NewLeap call was (0)
Status of MOVE command was (0)

12 For **utcpole:**
13 **cd /usr/ecs/OPS/CUSTOM/TOOLKIT/toolkit/database/common/CSC**

14 utcpole.dat utcpole.dat_old

15 Know thread for utcpole run:

16 cd /usr/ecs/OPS/CUSTOM/TOOLKIT/toolkit/src/CSC then do **ls** – select: update_utcpole.sh or run script for utcpole type in: **update_utcpole.sh**

17• A successful update will look like the following:

p0spg01{cmops}[294]->update_utcpole.sh
Status of PGS_CSC_UT1_update call was (0)
Status of MOVE command was (0)

Server Node Names Convention:

The naming convention is as follows:

Machine names are defined to be equivalent to the network hostname of the

machine. Network hostnames are limited to eight characters. On ECS we are now formatting these hostnames as svcimnni

Wheres : Site

g – GSFC
e – EDC
l – LaRC
n – NSIDC
a – ASF
j – JPL
p - PVC
t – VATC

v : Version

0 -- Version 2.0 At-Launch COTS design
1 -- Stood for B.1 COTS design; OBE, but still used in VATC
s -- Used for special SSI&T machines set up at GSFC and EDC

ci : Hardware Configuration Item

sp -- Science Processing (SPRHW)
ai -- Algorithm Integration and Test (AITHW)
aq -- Algorithm Quality Assurance (AQAHW)
pl -- Planning (PLNHW)
ms -- Management Subsystem (MSSHW)
cs -- Communications Subsystem (CSSHW)
in -- Interface (INTHW)
dm -- Data Management (DMGHW)
dr -- Data Repository (DRPHW)

ac -- Access Control Management (ACMHW)

ic -- Ingest Client (ICLHW)

wk -- Working Storage (WKSHW)

di -- Distribution (DIPHW)

as -- ASTER (ASTHW) [Occurs only at EDC]

te -- Test Equipment

m : Manufacturer

s -- Sun

g -- SGI

h -- HP

x -- X Terminal

p -- PC

nn : One-up number (01, 02, et cetera -- should be unique for the CI)

i : Interface type

<null> -- Production network

u -- User network

h -- HiPPI

Note that the machine name leaves off the last letter (the interface); hence, we generally refer to machines as "g0spg01", vice "g0spg01h". A machine may have multiple interfaces -- production, user, and HiPPI. So a single machine may show up in network documentation multiple times (g0spg01, g0spg01h, g0spg01u).

HOWTO_SSIT HELPFUL NOTES

Xterm format to bring up xterm windows for servers all at once

This list of xterm identifiers should be assigned a file name and placed into your home directory. Then invoke the file name when you want to create the entire list of xterms.

Example of filename: - xterm_pls

```
xterm -sb -sl 10000 -fg green -bg black -name "Resource Editor" &
xterm -sb -sl 10000 -fg green -bg black -name "Resource Model" &
xterm -sb -sl 10000 -fg green -bg black -name "Production Request Editor" &
xterm -sb -sl 10000 -fg green -bg black -name "Planning Workbench" &
xterm -sb -sl 10000 -fg green -bg black -name "Database" &
```

```
xterm -sb -sl 10000 -fg green -bg black -name "Planning Timeline" &
xterm -sb -sl 10000 -fg green -bg black -name "Logs" &
xterm -sb -sl 10000 -fg green -bg black -name "ECS Assist" &
xterm -sb -sl 10000 -fg green -bg black -name "g0sps06" &
xterm -sb -sl 10000 -fg green -bg black -name "g0ais01" &
xterm -sb -sl 10000 -fg green -bg black -name "g0spg01" &
xterm -sb -sl 10000 -fg green -bg black -name "g0drg01" &
xterm -sb -sl 10000 -fg green -bg black -name "g0pls02" &
# setenv ECS_HOME /usr/ecs
# setenv MODE TS2
```

Alias files to use while conducting SSI&T:

p0spg01{emcleod}51: **alias**

```
+       pushd
-       popd
```

More more !* |grep -v "Msg: Caught dce error: No more bindings (dce / rpc)" |grep -v "MsgLink :0 meaningfulname :EcAgManager::Recovery" |grep -v "MsgLink :0 meaningfulname :DsShSRequestRealSetStateSettingState" |grep -v "Command 1/1 execution complete"

cdstagebin cd /ecs/formal/STAGE/DSS/bin/sun5.5

dbg debugger -bg NavajoWhite -fn 12x24 !* &

dbs /home/jzhuang/bin/dbrowser-syb &

disp setenv DISPLAY !*

mgr DpAtMgr ConfigFile /usr/ecs/TS1/CUSTOM/cfg/DpAtMG.CFG ecs_mode TS1&

ops cd /usr/ecs/OPS/CUSTOM

ts1 cd /usr/ecs/TS1/CUSTOM

xslq_autosys isql -Uautosys -Pautosys -Sp0sps06_srvr

xsql_css isql -Ucss_role -Pwelcome -Sp0ins01_srvr

xsql_dss isql -UsdsrvApp -Pwelcome -Sp0acg01_sqs222_srvr

xsql_ios isql -Uios_role -Pwelcome -Sp0ins02_srvr

xsql_pdps isql -UpdpsUsers -Pwelcome -Sp0pls01_srvr

alias xsql_dss 'isql -UsdsrvApp -Pwelcome -Sp0acg01_sqs222_srvr'

alias xsql_ios 'isql -Uios_role -Pwelcome -Sp0ins01_srvr'

alias xsql_css 'isql -Ucss_role -Pwelcome -Sp0ins02_srvr'

alias xsql_autosys 'isql -Uautosys -Pautosys -Sp0sps06_srvr'

alias xsql_pdps 'isql -UpdpsUsers -Pwelcome -Sp0pls02_srvr'

alias xsql_ing 'isql -UEcInPolling -P3nWK0fG1 -Sp0icg01_srvr'

alias xsql_stmgt 'isql -UEcDsStFtpDisServer -PS71Oq4y3 -Sp0icg01_srvr'

 # alias for browser

Note: On mini daac, dbbrowser has to originate from workstation P0PLS01 to execute alias db_pdps to reach PDPS DB on p0pls02.

alias db_dss '/home/opscm/dbr/dbrowser-syb -UsdsrvApp -Pwelcome -Sp0acg01_sqs222_srvr &'

alias db_ios '/home/opscm/dbr/dbrowser-syb -Uios_role -Pwelcome -Sp0ins02_srvr &'

alias db_css '/home/opscm/dbr/dbrowser-syb -Ucss_role -Pwelcome -Sp0ins01_srvr &'
alias db_autosys '/home/opscm/dbr/dbrowser-syb -Uautosys -Pautosys -Sp0sps06_srvr &'
alias db_pdps '/home/opscm/dbr/dbrowser-syb -UpdpsUsers -Pwelcome -Sp0pls02_srvr &'
alias db_ing '/home/opscm/dbr/dbrowser-syb -UEcInPolling -P3nWK0fG1 -Sp0icg01_srvr &'
alias db_stmgt '/home/opscm/dbr/dbrowser-syb -UEcDsStFtpDisServer -PS71Oq4y3 -Sp0icg01_srvr &'
ls -laF look at root and .cshrc, .alias

howto_setup_orbits_and_pathmaps

ORBITS & PATHMAPS
The Path is a an orbit swath, defined for Landsat-7 ("WRS"),
which MISR uses in its processing.
Because the earth rotates under it, the path the satellite
traverses for its next orbit is not path 2, but path 17.
This mapping of orbit number to path number is found in the
PATHMAP ODL file.
In that file ABSOLUTE_PATH is what we call orbit number here,
and MAPPED_PATH is the path number.
Now, this mapping is fixed, and never changes.
What does change is the time each orbit starts.
This is because the orbit may drift and be subject to
maneuvers.
Periodically (say, every 2 weeks), the Flight Dynamics
Facility (FDF) at GSFC issues a new Orbit Start Time,
with corresponding Orbit Number.
When this happens, the PDPS ORBIT ODL must be updated,
with a new ORBIT_MODEL object, containing the new
ORBIT_START and corresponding ORBIT_NUMBER.
The new ORBIT_PATH_NUMBER is determined manually by
the operator, using the lookup table in the PATHMAP ODL file.

3. Example (MISR PGE7 test data)

FDF issues a bulletin stating that imaginary platform MPGE7
orbit number 27 starts at 14:37:39Z 07-Jan-96.
SSIT operator receives the bulletin, looks up in
the corresponding PATHMAP_WRS7.odl file to find that
ABSOLUTE_PATH XX corresponds to MAPPED_PATH XX.
NEED TO FIX THIS
The SSIT operator then creates a new ORBIT_MODEL object in the
ORBIT ODL file as follows:

```
OBJECT = ORBIT_MODEL
    CLASS = 2
    ORBIT_NUMBER = 27
    ORBIT_PERIOD = "SECS=5932"
    ORBIT_START = "01/07/1996 14:37:39Z"
    ORBIT_PATH_NUMBER = 90
```

```
END_OBJECT = ORBIT_MODEL
*****************************************************************************
```

howto_register_pge- prior to 6A.

```
# First prepare PGE and ESDT ODL files
# Update database
xterm -sb -sl 256 -bg maroon -fg "papaya whip" -cr "papaya whip" -fn
"*l*s*type*b*r*140
*" -T 'SSIT: Science Metadata Database Update' -n 'Science Metadata Update' -e
/usr/ecs
//OPS/CUSTOM/bin/DPS/EcDpAtDefinePGE &
# Update performance info
/usr/ecs//OPS/CUSTOM/bin/DPS/EcDpAtOpDbGui ConfigFile
/usr/ecs//OPS/CUSTOM/cfg/EcDpAtOp
DbGui.CFG ecs_mode OPS &
*********************************************************************
```

howto_register_dpr-prior to 6A

```
#############################################################################
# Make sure STMGT and SDSRV are up
#############################################################################
# Use either ECS Assist on texas *and* p0drg01,
# Alternatively, use ps:
texas:> ps -ef | grep EcDsScienceDataServer | grep TS1
   sdsrv 24410     1  0 10:17:57 pts/3    0:33
/usr/ecs//TS1/CUSTOM/bin/DSS/EcDsScience
DataServer ConfigFile /usr/ecs//TS1/CUS
p0drg01:> ps -ef | grep  Server | grep TS1 | grep ConfigFile
   stmgt  3786     1  0   Dec 30 ?      10:39
/usr/ecs/TS1/CUSTOM/bin/DSS/EcDsStStaging
DiskServer ConfigFile /usr/ecs/TS1/CUS
   stmgt  3803     1  0   Dec 30 ?      10:31
/usr/ecs/TS1/CUSTOM/bin/DSS/EcDsStStaging
MonitorServer ConfigFile /usr/ecs/TS1/
   stmgt  3770     1  0   Dec 30 ?      16:18
/usr/ecs/TS1/CUSTOM/bin/DSS/EcDsStArchive
Server ConfigFile /usr/ecs/TS1/CUSTOM/
   stmgt  3815     1  0   Dec 30 ?      10:59
/usr/ecs/TS1/CUSTOM/bin/DSS/EcDsStFtpDisS
erver ConfigFile /usr/ecs/TS1/CUSTOM/c
# If any of these are not present, that server is down.
# You must arrange to have it up before continuing
*********************************************************************
```

Using Production Request Editor and Planning Workbench

PRE:
telnet p0pls01, cmops, ops, cd utilities,
then invoke EcplStartPRE_IF OPS 3 &
PWB: telnet p0pls01, cmops, ops, cd utilities, invoke EcPlSlayAll prior to using PWB.
then invoke EcPlStartAll OPS 3 & or 2
times used for plan activation: 01/01/1990 time 00:00:00
AutoSys- To Monitor DPS , telnet to: p0sps06_srvr
##

howto_Make a Production Request

##
rlogin p0pls01 -l pls
awhitele
cd /usr/ecs/TS1/CUSTOM/utilities
set path = (/usr/bin)
source .buildrc
setenv DISPLAY mojave:0.0
Make sure no one else is running PREditor, then run it
/usr/ucb/ps -auxwww | grep -i pre | grep TS1
EcPlStartPRE_IF TS1 1
Click PREdit
Click PGE, select it, OK
Set time to input, e.g. (CPGE1)
06 12 1997 00 00 00
06 14 1997 00 00 00
****

Additional SSI&T Howto_procedures and Production rules taken from
PDPS tests documented in the following files as new functionality is accomplished:
SCF: cd /home/PDPS/docs - look through list of 'howto_s'

SSIT Debugging Information:

Problem: Core dumps when running various DSS interface tools (Insert Test, Insert Static, Insert Exe Tar, Get MCF, SSAP Editor).

What to Do:

-- If the tool core dumps before it tries to contact DSS, check to make sure that the HOST environment variable is set correctly.

- Check the cfg file and make sure that the security file (under the name DataFileName) is pointing to the correct file. Also check that the ApplStartNum is correct.

Examples of SSEP, MCF and ESDT files:

/home/emcleod/SSEP
p0ais01{cmops}24: ls

MODPGE07.tar	PGE07.csh	PGS_1010	PGS_12	PGS_3	PGS_6
MOD_PR10.exe	PGS_1	PGS_1014	PGS_13	PGS_4	PGS_7
MOD_PR10.tar.met	PGS_10	PGS_11	PGS_14	PGS_5	PGS_9

```
/home/emcleod/MCF
MD10L2.MCF                MD35L2.met
MOD02HKM.A1997217.1730.002.hdf  MOD03.met
MD10L2.met                MOD02H.MCF
MOD03.A1997217.1730.002.hdf
MOD35_L2.A1997217.1730.002.hdf
MD35L2.MCF                MOD02H.met                MOD03.MCF
            MOD_PR10.Ayyyyddd.hhmm.vvv.hdf
p0ais01{cmops}49: /home/emcleod/ESDT
DsESDTMoMD10L2.001.desc                                    DsESDTMoMOD02H.001.desc
libDsESDTMoMOD03.001Sh.so
DsESDTMoMD35L2.001.desc    libDsESDTMoMD35L2Sh.so
DsESDTMoMOD03.001.desc     libDsESDTMoMOD02HSh.so
libDsESDTMoMD10L2.001Sh.so
```

Abbreviations and Acronyms

AIT	Algorithm Integration and Test
AIT	Algorithm Integration and Test (workstation)
ANSI	American National Standards Institute
API	Applications Programming Interface
ASCII	American Standard Code for Information Interchange
CDR	Critical Design Review
COTS	Commercial Off The Shell (hardware or software)
CSDT	Computer Science Data Type
DAAC	Distributed Active Archive Center
DAO	Data Assimilation Office
DAP	Delivered Algorithm Package
DBA	Database Administrator
DEM	Digital Elevation Model
DoD	Department of Defense
DPR	Data Processing Request
DPS	Data Processing Subsystem
DSS	Data Server Subsystem
ECS	EOSDIS Core System
EOSDIS	Earth Observing System Data and Information System
ESDIS	Earth Science Data and Information System
ESDT	Earth Science Data Type
FOS	Flight Operations Segment
FTP	file transfer protocol
GUI	Graphical User Interface
HDF	Hierarchical Data Format
IDL	Interactive Data Language
IMF	Integrated Metastorage Facility

IRIX	
IT	Instrument Team
MB	Megabytes
MCF	Metadata Configuration File
NASA	National Aeronautics and Space Administration
NMC	National Meteorological Center
NCEP	National Center for Environmental Prediction
ODL	Object Description Language
OPR	On-Demand Production Request
PLS	Planning Subsystem
PR	Production Request
PCF	Process Control File
PDPS	Planning and Data Processing System
PGE	Product Generation Executive
PGM	Portable Grey Map (image file format)
PH	Production History
PLN	Planning (Workstation)
PSA	Product Specific Attribute
QA	Quality Assurance
SA	System Administrator
SCF	Science Computing Facility
SDP	Science Data Processing (Toolkit)
SDPS	Science Data Processing System
SGI	Silicon Graphics, Inc.
SMF	Status Message Facility
SPR	Science Processor (Workstation)
SSI&T	Science Software Integration and Test
UR	Universal Reference
URL	Universal Resource Locator
VA	VOB Administrator

VADS	
VOB	Versioned Object Base
WWW	World Wide Web

This page intentionally left blank.

Glossary

Word or Phrase	Definition
ASCII File	A data file whose contents are encoded as ASCII characters.
Binary File	A data file whose contents are in binary form (i.e. not encoded).
Configuration Management	
Data Visualization	The ability to examine a data file as a raw data dump, a plot, or an image.
Delivered Algorithm Package (DAP)	An ensemble of PGE source code, control files, makefiles, documentation, and other related files delivered in a package from the SCF to the DAAC.
Earth Science Data Type (ESDT)	
HDF File	A data file whose format follows the NCSA Hierarchical Data Format standard.
HDF-EOS File	An HDF file which, at a minimum, has been created using the SDP Toolkit and containing the ECS Core metadata.
Makefile	Input file for the UNIX make facility which allows a user to specify dependencies between source (and other) files, primarily for compiling and linking programs, but also used for other activities such as building documentation and installing software.
Problem Report	
Product Generation Executive (PGE)	The smallest schedulable unit of science software. A PGE may consist of a single binary executable or a number of executables wrapped by a shell script. To the ECS system, a PGE is a black box.
Profiling	The measurement of the science software's resource usage requirements such as memory, disk space requirements, and CPU time.
Science Software	Software developed at the SCFs to generate standard (and other) science data.
Source MCF	
Standards Checking	The process of checking whether or not source code follow prescribed coding standards.
Target MCF	
Universal Reference (UR)	

This page intentionally left blank.