305-CD-026-002

**EOSDIS Core System Project**

# Release B SDPS Planning Subsystem Design Specification for the ECS Project

March 1996

Hughes Information Technology Systems
Upper Marlboro, MD

# Release B SDPS Planning Subsystem Design Specification for the ECS Project

**March 1996**

Prepared Under Contract NAS5-60000
CDRL Item #046

**SUBMITTED BY**

Rick Kochhar /s/ 3/20/96

Rick Kochhar, Release B CCB Chairman Date
EOSDIS Core System Project

**Hughes Information Technology Systems**
Upper Marlboro, Maryland

This page intentionally left blank.

# Preface

This document is one of eighteen comprising the detailed design specifications of the SDPS and CSMS subsystem for Release B of the ECS project. A complete list of the design specification documents is given below. Of particular interest are documents number 305-CD-020, which provides an overview of the subsystems and 305-CD-039, the Data Dictionary, for those reviewing the object models in detail.

The SDPS and CSMS subsystem design specification documents for Release B of the ECS Project include:

| | |
|---|---|
| 305-CD-020 | Release B Overview of the SDPS and CSMS Segment System Design Specification |
| 305-CD-021 | Release B SDPS Client Subsystem Design Specification |
| 305-CD-022 | Release B SDPS Interoperability Subsystem Design Specification |
| 305-CD-023 | Release B SDPS Data Management Subsystem Design Specification |
| 305-CD-024 | Release B SDPS Data Server Subsystem Design Specification |
| 305-CD-025 | Release B SDPS Ingest Subsystem Design Specification |
| 305-CD-026 | Release B SDPS Planning Subsystem Design Specification |
| 305-CD-027 | Release B SDPS Data Processing Subsystem Design Specification |
| 305-CD-028 | Release B CSMS Segment Communications Subsystem Design Specification |
| 305-CD-029 | Release B CSMS Segment Systems Management Subsystem Design Specification |
| 305-CD-030 | Release B GSFC Distributed Active Archive Center Design Specification |
| 305-CD-031 | Release B LaRC Distributed Active Archive Center Design Specification |
| 305-CD-033 | Release B EDC Distributed Active Archive Center Design Specification |
| 305-CD-034 | Release B ASF Data Center Distributed Active Archive Center Design Specification |
| 305-CD-035 | Release B NSIDC Distributed Active Archive Center Design Specification |
| 305-CD-036 | Release B JPL Distributed Active Archive Center Design Specification |
| 305-CD-037 | Release B ORNL Distributed Active Archive Center Design Specification |
| 305-CD-038 | Release B System Monitoring and Coordination Center Design Specification |
| 305-CD-039 | Release B Data Dictionary for Subsystem Design Specification |

Object models presented in this document have been exported directly from CASE or DBMS tools and in some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these drawings on line using the Portable Document Format (PDF) electronic copy available via the ECS Data Handling System (EDHS) at:
URL http://edhs1.gsfc.nasa.gov.

This document is a formal contract deliverable with an approval code of 2; as such it requires Government review and approval prior to acceptance and use. This document is under ECS contractor configuration control. Once this document is approved, Contractor approved changes are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by document change notice (DCN) or by complete revision.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Hughes Information Technology Systems
1616 McCormick Drive
Upper Marlboro, MD 20774-5372

# Abstract

This document describes the Release B CDR design for the SDPS Planning subsystem. It defines the Planning Subsystem (PLS) computer software and hardware architectural design, as well as subsystem design based on Level 4 requirements. The subsystem comprises of two CIs: the PLANG CSCI and the PLNHW HWCI. The PLANG CSCI is used to plan production processing of science data. That production is carried out by the Data Processing Subsystem (DPS). The PLANG CSCI also performs resource planning for a site. Object model diagrams and event traces are used to illustrate the current design of the PLANG CSCI.

***Keywords:*** Planning, SDPS, Release B, OMT, PDPS, PLS, Production, Resource Planning

This page intentionally left blank.

# Change Information Page

| List of Effective Pages | |
|---|---|
| **Page Number** | **Issue** |
| Title | Submitted as Final |
| iii through xvi | Submitted as Final |
| 1-1 and 1-2 | Submitted as Final |
| 2-1 and 2-2 | Submitted as Final |
| 3-1 through 3-16 | Submitted as Final |
| 4-1 through 4-308 | Submitted as Final |
| 5-1 through 5-10 | Submitted as Final |
| A-1 through A-24 | Submitted as Final |
| AB-1 and AB-2 | Submitted as Final |

| Document History | | | |
|---|---|---|---|
| **Document Number** | **Status/Issue** | **Publication Date** | **CCR Number** |
| 305-CD-026-001 | Preliminary | October 1995 | 95-0741 |
| 305-CD-026-002 | Submitted as Final | March 1996 | 96-0231 |

This page intentionally left blank.

# Contents


## Preface

## Abstract

## 1. Introduction

1.1 Identification ..... 1-1
1.2 Purpose and Scope ..... 1-1
1.3 Status and Schedule ..... 1-1
1.4 Organization ..... 1-1

## 2. Related Documentation

2.1 Parent Documents ..... 2-1
2.2 Applicable Documents ..... 2-1

## 3. Subsystem Overview

3.1 Context ..... 3-1
3.2 Subsystem Overview ..... 3-6
3.2.1 Use of COTS within the Planning and Data Processing Subsystems ..... 3-6
3.2.2 Summary of Changes to the Planning Subsystem ..... 3-9
3.2.3 Key Design Drivers ..... 3-11
3.2.4 Planning Subsystem Use of Key Design Mechanisms ..... 3-15
3.2.5 Subsystem Structure ..... 3-16

## 4. PLANG - Production Planning CSCI

4.1 CSCI Overview ..... 4-1
4.1.1 PDPS Database ..... 4-2
4.1.2 Production Request Editor ..... 4-2
4.1.3 Production Planning Workbench ..... 4-3
4.1.4 Planning Subscription Editor ..... 4-4
4.1.5 Subscription Manager ..... 4-4
4.1.6 On-Demand Manager ..... 4-4
4.1.7 Resource Planning Workbench ..... 4-4

4.2 CSCI Object Model ..... 4-5
4.2.1 PGE Profile View ..... 4-5
4.2.2 Production Request View ..... 4-8
4.2.3 Subscription Submission View ..... 4-11
4.2.4 Production Planning View ..... 4-13
4.2.5 Production Planning User Interface View ..... 4-15
4.2.6 Publishing Plans View ..... 4-17
4.2.7 Resource Planning View ..... 4-19
4.2.8 Resource Management View ..... 4-22
4.2.9 Plan Activation View ..... 4-24
4.2.10 Subscription Manager View ..... 4-26
4.2.11 Data Activity Times View ..... 4-28
4.2.12 DBMS Proxy Agent View ..... 4-30
4.2.13 On-Demand Manager View ..... 4-32
4.3 Production Planning Class Descriptions ..... 4-34
4.3.1 DmLmClRequestServer Class ..... 4-34
4.3.2 DpPrScheduler Class ..... 4-34
4.3.3 DsClCommand Class ..... 4-36
4.3.4 DsClESDTReference Class ..... 4-37
4.3.5 DsClESDTReferenceCollector Class ..... 4-37
4.3.6 DsClQuery Class ..... 4-38
4.3.7 DsClRequest Class ..... 4-38
4.3.8 DsClSubscription Class ..... 4-39
4.3.9 EcEvent Class ..... 4-39
4.3.10 EcMpMsgCb Class ..... 4-40
4.3.11 EcPfClient Class ..... 4-40
4.3.12 EcPfManagedServer Class ..... 4-41
4.3.13 EcRequest Class ..... 4-44
4.3.14 EcUrUR Class ..... 4-44
4.3.15 GlCallback Class ..... 4-45
4.3.16 GlParameterList Class ..... 4-45
4.3.17 IoAdProduct Class ..... 4-46
4.3.18 IoAdProductList Class ..... 4-46
4.3.19 IoAdProductSearchCommand Class ..... 4-46
4.3.20 IoAdProvider Class ..... 4-47
4.3.21 IoAdService Class ..... 4-47
4.3.22 PlActivePlan Class ..... 4-48
4.3.23 PlActivities Class ..... 4-50
4.3.24 PlActivity Class ..... 4-52

4.3.25 PlAlternateDataGranuleNB Class ..... 4-54
4.3.26 PlAlternateNB Class ..... 4-55
4.3.27 PlCluster Class ..... 4-57
4.3.28 PlComputer Class ..... 4-58
4.3.29 PlDASDelta Class ..... 4-60
4.3.30 PlDASDifferent Class ..... 4-61
4.3.31 PlDASNB Class ..... 4-62
4.3.32 PlDATRecord Class ..... 4-64
4.3.33 PlDBMSProxyAgent Class ..... 4-66
4.3.34 PlDPRB Class ..... 4-66
4.3.35 PlDPRs Class ..... 4-73
4.3.36 PlDataAvailabilityTimes Class ..... 4-74
4.3.37 PlDataDependencies Class ..... 4-77
4.3.38 PlDataGranule Class ..... 4-78
4.3.39 PlDataScheduled Class ..... 4-81
4.3.40 PlDataSchedules Class ..... 4-83
4.3.41 PlDataSource Class ..... 4-83
4.3.42 PlDataSourceFactory Class ..... 4-85
4.3.43 PlDataTranferHistory Class ..... 4-85
4.3.44 PlDataTypeB Class ..... 4-87
4.3.45 PlDataTypeCatalogue Class ..... 4-93
4.3.46 PlDataTypeReq Class ..... 4-94
4.3.47 PlDiskPartition Class ..... 4-96
4.3.48 PlEDASModeRecordNB Class ..... 4-97
4.3.49 PlEDASRecordNB Class ..... 4-98
4.3.50 PlEntryScreenNB Class ..... 4-100
4.3.51 PlExportedPlanNB Class ..... 4-101
4.3.52 PlFOSDASFile Class ..... 4-103
4.3.53 PlFile Class ..... 4-104
4.3.54 PlGroundActivity Class ..... 4-104
4.3.55 PlGroundEvent Class ..... 4-106
4.3.56 PlGroundEventAllocation Class ..... 4-109
4.3.57 PlGroundEventExecutable Class ..... 4-109
4.3.58 PlGroundEvents Class ..... 4-110
4.3.59 PlImportedActivity Class ..... 4-111
4.3.60 PlInstModeRecords Class ..... 4-112
4.3.61 PlInstrumentModes Class ..... 4-113
4.3.62 PlMetaDataChecks Class ..... 4-115
4.3.63 PlNetwork Class ..... 4-117

4.3.64 PlOnDemandExceed Class ..... 4-117
4.3.65 PlOnDemandManagerNB Class ..... 4-119
4.3.66 PlOnDemandPRNB Class ..... 4-121
4.3.67 PlOnDemandReplanValues Class ..... 4-123
4.3.68 PlOrbitModelNB Class ..... 4-124
4.3.69 PlOrbitScheduledNB Class ..... 4-125
4.3.70 PlOtherTypes Class ..... 4-126
4.3.71 PlOutputYield Class ..... 4-127
4.3.72 PlPDASFile Class ..... 4-128
4.3.73 PlPDASMetaData Class ..... 4-129
4.3.74 PlPDASRecords Class ..... 4-131
4.3.75 PlPGE Class ..... 4-132
4.3.76 PlPGEActivity Class ..... 4-136
4.3.77 PlPGECollection Class ..... 4-137
4.3.78 PlPGEPriorityNB Class ..... 4-138
4.3.79 PlPGEProfile Class ..... 4-139
4.3.80 PlPRCollectionNB Class ..... 4-140
4.3.81 PlPRPriorityNB Class ..... 4-142
4.3.82 PlPerformance Class ..... 4-143
4.3.83 PlPgeFactory Class ..... 4-146
4.3.84 PlPlanASCIIReportFile Class ..... 4-146
4.3.85 PlPlanB Class ..... 4-147
4.3.86 PlPlanBinaryReportFile Class ..... 4-155
4.3.87 PlPlanGenerationUIB Class ..... 4-156
4.3.88 PlPlanMetadataFile Class ..... 4-157
4.3.89 PlPlanningWorkbenchUI Class ..... 4-159
4.3.90 PlPopupMessage Class ..... 4-161
4.3.91 PlProdStratNB Class ..... 4-162
4.3.92 PlProdStratUINB Class ..... 4-164
4.3.93 PlProductionPlannersUI Class ..... 4-165
4.3.94 PlProductionRequestB Class ..... 4-166
4.3.95 PlProductionRequestUI Class ..... 4-170
4.3.96 PlPublishedPlan Class ..... 4-172
4.3.97 PlReplanCriteria Class ..... 4-173
4.3.98 PlReplanCriteriaUI Class ..... 4-174
4.3.99 PlRescUseThreshNB Class ..... 4-175
4.3.100 PlRescUseThreshUINB Class ..... 4-176
4.3.101 PlResource Class ..... 4-177
4.3.102 PlResourceChange Class ..... 4-177

4.3.103 PlResourceManager Class ..... 4-178
4.3.104 PlResourceRequirement Class ..... 4-179
4.3.105 PlRoutineArrival Class ..... 4-180
4.3.106 PlService Class ..... 4-182
4.3.107 PlSourcetoDsHistoryNB Class ..... 4-182
4.3.108 PlString Class ..... 4-184
4.3.109 PlSubMsgCb Class ..... 4-185
4.3.110 PlSubscriptionManager Class ..... 4-188
4.3.111 PlSubscriptionSubmitIF Class ..... 4-189
4.3.112 PlTile Class ..... 4-190
4.3.113 PlTileScheduledNB Class ..... 4-192
4.3.114 PlTimeLineDisplay Class ..... 4-195
4.3.115 PlTimeScheduled Class ..... 4-196
4.3.116 PlTimer Class ..... 4-197
4.3.117 PlUserParameters Class ..... 4-199
4.3.118 PlUserPriorityNB Class ..... 4-201
4.4 PLANG Dynamic Model ..... 4-202
4.4.1 Production Request Scenario ..... 4-202
4.4.2 Data Availability Scenario ..... 4-207
4.4.3 Subscription Submission Scenario ..... 4-210
4.4.4 Subscription Withdrawal Scenario ..... 4-212
4.4.5 Plan Creation Scenario ..... 4-214
4.4.6 Assigning a Priority to an Activity Scenario ..... 4-217
4.4.7 Publishing a Plan Scenario ..... 4-220
4.4.8 Importing a Plan from the Data Server Scenario ..... 4-223
4.4.9 Plan Activation Scenario ..... 4-226
4.4.10 Statusing a Plan Scenario ..... 4-229
4.4.11 Canceling a Plan Scenario ..... 4-232
4.4.12 Deleting a Plan Scenario ..... 4-234
4.4.13 Subscription Notification Scenario ..... 4-235
4.4.14 Subscription Notification with Spatial Based Input Scenario ..... 4-238
4.4.15 Subscription Notification with Alternate Inputs Scenario ..... 4-240
4.4.16 Data Availability Times Scenario ..... 4-242
4.4.17 Limited Automatic Replan based on a New DAS ..... 4-245
4.4.18 User Logon and Authentication Scenario ..... 4-248
4.4.19 On-Demand Production Request Scenario (Processed) ..... 4-249
4.4.20 Limited Automatic Replan based on an On-demand Production Request Scenario ..... 4-253

4.4.21 On-Demand Production Request Scenario (Delayed) ..... 4-256
4.4.22 Entering and Approving a Resource Reservation Scenario ..... 4-259
4.4.23 Creating a New Resource Plan ..... 4-262
4.5 CSCI Structure ..... 4-267
4.5.1 Planning Subscription Editor CSC ..... 4-267
4.5.2 Production Request Editor CSC ..... 4-268
4.5.3 Subscription Manager CSC ..... 4-269
4.5.4 Production Planning Workbench CSC ..... 4-270
4.5.5 Planning Object Library ..... 4-272
4.5.6 PDPS Database CSC ..... 4-295
4.5.7 On-Demand Manager CSC ..... 4-297
4.5.8 Resource Planning Workbench CSC ..... 4-297
4.6 PLANG CSCI Management and Operation ..... 4-299
4.6.1 PLANG CSCI Operation ..... 4-299
4.6.2 System Management Strategy ..... 4-302
4.6.3 Operator Interfaces ..... 4-304
4.6.4 Reports ..... 4-307

## 5. Planning Subsystem Hardware CI

5.1 Hardware Design Drivers ..... 5-1
5.1.1 Key Trade-Off Studies and Prototypes ..... 5-2
5.1.2 Sizing and Performance Analysis ..... 5-3
5.1.3 Scalability, Evolvability, Migration to Release B ..... 5-4
5.2 HWCI Structure ..... 5-4
5.2.1 Connectivity ..... 5-6
5.2.2 HWCI Components ..... 5-6
5.3 Fail-over and Recovery Strategy ..... 5-9

## Figures

3.1-1. Context Diagram ..... 3-2
3.2-1. AutoSys Production Sequence ..... 3-7
3.2-2. Graphical Schedule Displays Provided by AutoXpert ..... 3-8
4.1-1. CSCI Overview ..... 4-1
4.1-2. Planning SCSI Component Diagram ..... 4-2
4.2-1. PGE Profile View ..... 4-7
4.2-2. Production Request View ..... 4-10
4.2-3. Subscription Submission View ..... 4-12
4.2-4. Production Planning View ..... 4-14

4.2-5. Production Planning User Interface View ..... 4-16
4.2-6. Publishing Plans View ..... 4-18
4.2-7. Resource Planning View ..... 4-21
4.2-8. Resource Management View ..... 4-23
4.2-9. Plan Activation View ..... 4-25
4.2-10. Subscription Manager View ..... 4-27
4.2-11. Datat Activity Times View ..... 4-29
4.2-12. DBMS Proxy Agent View ..... 4-31
4.2-13. On-Demand Manager View ..... 4-33
4.4-1. Production Request Scenario ..... 4-206
4.4-2. Data Availability Scenario ..... 4-209
4.4-3. Subscription Submission Scenario ..... 4-211
4.4-4. Subscription Withdrawal Scenario ..... 4-213
4.4-5. Plan Creation Scenario ..... 4-216
4.4-6. Assigning a Priority to an Activity Scenario ..... 4-219
4.4-7. Publishing a Plan Scenario ..... 4-222
4.4-8. Importing a Plan from the Data Server Scenario ..... 4-225
4.4-9. Plan Activation Scenario Scenario ..... 4-228
4.4-10. Statusing a Plan Scenario ..... 4-231
4.4-11. Cancelling a Plan Scenario ..... 4-233
4.4-12. Subscription Notifications Scenario ..... 4-237
4.4-13. Subscription Notification with Spatial Based Input Scenario ..... 4-239
4.4-14. Subscription Notification with Alternate Inputs Scenario ..... 4-241
4.4-15. Data Availability Times Schedule ..... 4-244
4.4-16. Limited Automatic Replan Based on a New DAS ..... 4-247
4.4-17. On-Demand Production Request Scenario (Processed) ..... 4-252
4.4-18. Limited Automatic Replan Based on an On-Demand Production Request Scenario ..... 4-255
4.4-19. On-Demand Production Request Scenario (Delayed) Request Scenario ..... 4-258
4.4-20. Entering and Approving a Resource Reservation Scenario ..... 4-261
4.4-21. Create Resource Plan Scenario ..... 4-263
4.4-22. Commit Resource Plan Scenario ..... 4-266
4.5-1. Delphi Scheduler Object Model ..... 4-277
4.5-2. Delphi Resource Object Model ..... 4-280
4.5-3. Delphi Timeline Object Model ..... 4-284
4.5-4. Planning Workbench Architecture ..... 4-286
4.5-5. Planning Workbench System Resource Object Model ..... 4-289
4.5-6. Planning Workbench System Scheduling Object Model ..... 4-291
4.5-7. Delphi Schedule Event Trace ..... 4-294

5-1. Planning Block Diagram ..... 5-5
5-2. Planning Generic Network Connectivity ..... 5-6
5-3. Primary and Secondary Server Concept Overview ..... 5-10

## Table

3.1-1. Planning Subsystem Interfaces ..... 3-3
4.5-1. Production Planning CSCs ..... 4-267
5-1. PLNHW Logical Components and Equipment Classes ..... 5-8

## Appendix A. Requirements Trace

## Abbreviations and Acronyms

# 1. Introduction

## 1.1 Identification

This Release B SDPS Planning Subsystem Design Specification for the ECS Project, Contract Data Requirement List (CDRL) Item 046, with requirements specified in Data Item Description (DID) 305/DV2, is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000. This publication is part of a series of documents comprising the Science and Communications Development Office design specification for the Communications and System Management segment (CSMS) and the Science Data Processing Subsystem (SDPS) for Release B.

## 1.2 Purpose and Scope

The Release B SDPS Planning Subsystem Design Specification defines the progress of the design. It defines the Planning Subsystem computer software and hardware architectural design, as well as subsystem design based on Level 4 requirements.

This document reflects the February 14, 1996 Technical Baseline maintained by the ECS Configuration Control Board in accordance with ECS Technical Direction No.11 dated December 6, 1994.

## 1.3 Status and Schedule

This submittal of DID 305/DV2 meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. The present version of this document will be reviewed during the CSMS Critical Design Review (CDR).

## 1.4 Organization

The document is organized to describe the Planning subsystem design as follows:

Section 1 provides information regarding the identification, scope, status, and organization of this document.

Section 2 provides a listing of the related documents, which were used as source information for this document.

Section 3 provides an overview of the subsystem, focusing on the high-level design concept. This provides general background information to put the Planning subsystem into context.

Sections 4 contain the structure of the computer software configuration item (CSCI) comprising the Planning subsystem.

Section 5 contains the hardware configuration item (HWCI) design of the Planning subsystem.

Appendix A contains a table mapping each Level 4 Requirement to the design components which will implement that requirements' functionality.

The section Abbreviations and Acronyms contains an alphabetized list of the definitions for abbreviations and acronyms used in this volume.

This page intentionally left blank.

# 2. Related Documentation

## 2.1 Parent Documents

The parent document is the document from which the scope and content of this Planning Subsystem Design Specification is derived.

| | |
|---|---|
| 194-207-SE1-001 | System Design Specification for the ECS Project |

## 2.2 Applicable Documents

The following documents are referenced within this Planning Subsystem Design Specification, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document.

| | |
|---|---|
| 209-CD-001-003 | Interface Control Document Between EOSDIS Core System (ECS) and the NASA Science Internet (NSI) |
| 209-CD-002-003 | Interface Control Document Between EOSDIS Core System (ECS) and ASTER Ground Data System |
| 209-CD-003-003 | Interface Control Document Between EOSDIS Core System (ECS) and EOS-AM Project for AM-1 Spacecraft Analysis Software |
| 209-CD-004-003 | Data Format Control Document for the Earth Observing System (EOS) AM-1 Project Data Base |
| 209-CD-005-005 | Interface Control Document Between EOSDIS Core System (ECS) and Science Computing Facilities (SCF) |
| 209-CD-006-005 | Interface Control Document Between EOSDIS Core System (ECS) and National Oceanic and Atmospheric Administration (NOAA) Affiliated Data Center (ADC) |
| 209-CD-007-003 | Interface Control Document Between EOSDIS Core System (ECS) and TRMM Science Data and Information System (TSDIS) |
| 209-CD-008-004 | Interface Control Document Between EOSDIS Core System (ECS) and the Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC) |
| 209-CD-009-002 | Interface Control Document Between EOSDIS Core System (ECS) and the Marshall Space Flight Center (MSFC) Distributed Active Archive Center (DAAC) |
| 209-CD-011-004 | Interface Control Document Between EOSDIS Core System (ECS) and the Version 0 System |
| 305-CD-002-002 | Science Data Processing Segment (SDPS) Segment Design Specification for the ECS Project |
| 305-CD-003-002 | Communications and System Management Segment (CSMS) Design Specification for the ECS Project |

| | |
|---|---|
| 308-CD-001-005 | Software Development Plan for the ECS Project |
| 313-CD-004-001 | Release A CSMS/SDPS Internal Interface Control Document for the ECS Project |
| 423-41-03 | Goddard Space Flight Center, EOSDIS Core System (ECS) Contract Data Requirements Document |

# 3. Subsystem Overview

The Planning subsystem is responsible for supporting operation staff in managing the data production activities at a site. The Planning subsystem assists the operations staff in performing two major functions:

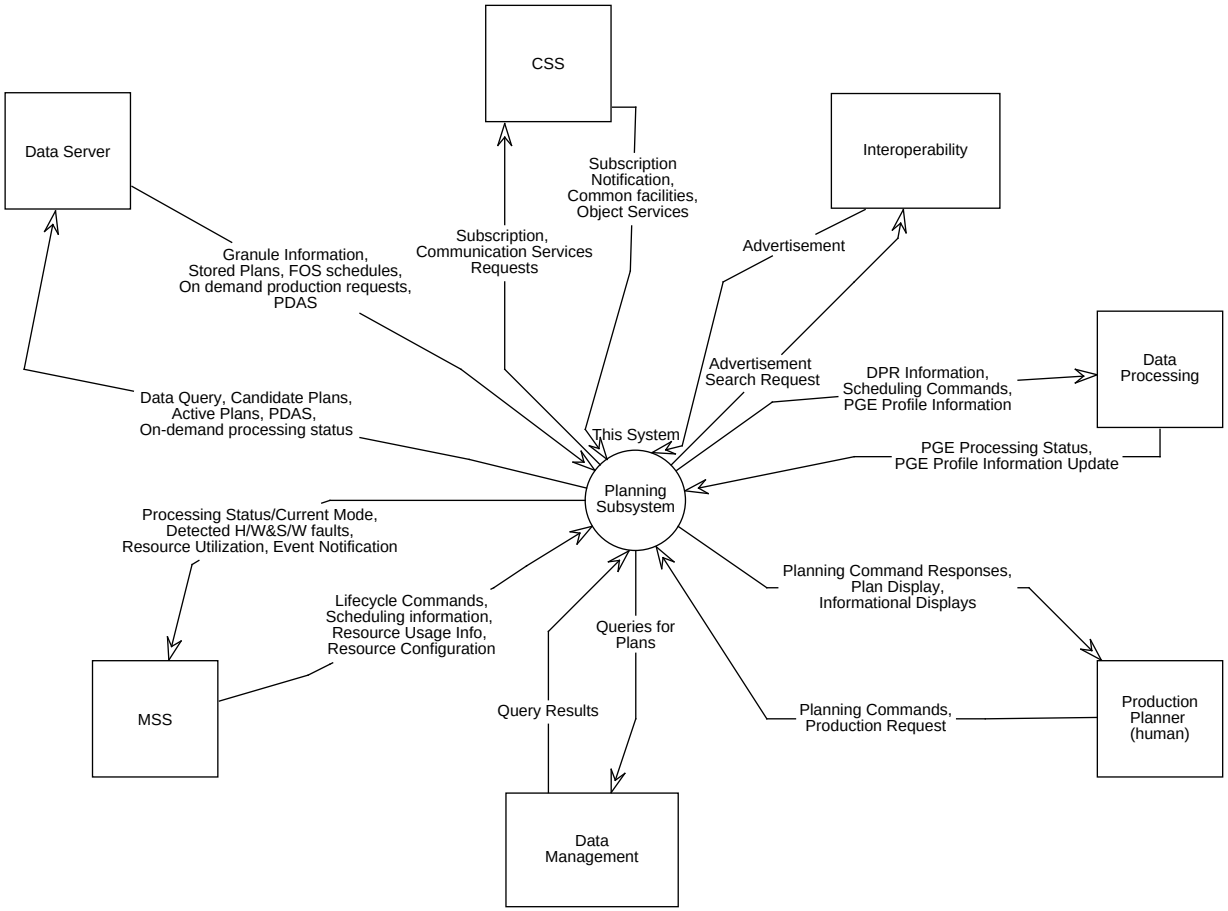
- Defining the data processing tasks to be performed at a site; and
- Generating efficient plans for the scheduling of those tasks.

In addition, the Planning subsystem is responsible for coordinating the production with the Data Server and Data Processing subsystems to achieve a highly automated production system.

## 3.1 Context

A context diagram illustrating the relationships between the Planning subsystem and other subsystems is shown in Figure 3.1-1, and summarized in Table 3.1-1.

The context diagram shows those interfaces which are a part of the EOS AM-1 release Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS). Interfaces that provide additional capabilities will be described in the detailed design document at the appropriate future release.




Planning Commands
Create Plan Command, Activate Plan Command, Cancel Plan Command,
Limited Automatic Replan Criteria, On-Demand Resource Usage Thresholds,
Production Stategies and Subscriptions

Scheduling Commands
Schedule Job Command, Cancel Job Command, Update Job Command,
Release Job Command, Suspend Job Command, Resume Job Command

Planning Command Response
Production Request Response, Create Plan Command Response,
Active Plan Command Response, Cancel Plan Command Response

Informational Displays
PDAS and FOS DAS notices, warning messages of delayed processing,
cross-DAAC dependencies, descriptions of job causing scheduling conflicts,
notices of on-demand production request deferrals

***Figure 3.1-1. Context Diagram***

*Table 3.1-1. Planning Subsystem Interfaces (Page 1 of 2)*

| Source | Destination | Data Types | Data Volume | Frequency |
|---|---|---|---|---|
| Production Planner | Planning | Subscriptions | low | as required |
| Production Planner | Planning | On-Demand Resource Usage Thresholds | low | as required |
| Production Planner | Planning | Production Strategies | low | as required |
| Production Planner | Planning | Limited Automatic Replan Criteria | low | as required |
| Production Planner | Planning | Create Plan Command | low | as required |
| Production Planner | Planning | Activate Plan Command | low | as required |
| Production Planner | Planning | Cancel Plan Command | low | as required |
| Production Planner | Planning | Production Requests | low | as required |
| Planning | Production Planner | PDAS and FOS DAS notices (display) | low | when input data is available |
| Planning | Production Planner | Warning messages of delayed processing (display) | low | when processing will not complete per original schedule |
| Planning | Production Planner | Cross-DAAC data dependencies (display) | low | as required |
| Planning | Production Planner | Description of job causing scheduling conflicts (display) | low | when scheduling conflict occurs |
| Planning | Production Planner | Notice of On-Demand Production Request deferrals | low | when on-demand data product exceeds a predefined resource usage threshold |
| Planning | Production Planner | Production Request Response | low | when queried |
| Planning | Production Planner | Create Plan Command Response | low | when queried |
| Planning | Production Planner | Activate Plan Command Response | low | when queried |
| Planning | Production Planner | Cancel Plan Command Response | low | when queried |
| Planning | Production Planner | Plan Display | low | when queried |
| Data Server | Planning | FOS schedules | low | as required |
| Data Server | Planning | On-demand Production Requests | low | as required |
| Data Server | Planning | PDAS | low | as required |
| Data Server | Planning | Stored active or candidate plans | low | as queried |
| Data Server | Planning | Granule Information | low | as required |

*Table 3.1-1. Planning Subsystem Interfaces (Page 2 of 2)*

| Source | Destination | Data Types | Data Volume | Frequency |
|---|---|---|---|---|
| Planning | Data Server | PDAS | low | created from active plans |
| Planning | Data Server | On-demand processing status notices | low | as required |
| Planning | Data Server | Data Query | low | as required |
| Planning | Data Server | Candidate Plans | low | as required |
| Planning | Data Server | Active Plans | low | as required |
| Processing | Planning | PGE Processing Status | low | as required |
| Processing | Planning | PGE Profile Information Update | low | as required |
| Planning | Processing | DPR Information | low | as required |
| Planning | Processing | Schedule Job Command | low | as required |
| Planning | Processing | Cancel Job Command | low | as required |
| Planning | Processing | Update Job Command | low | as required |
| Planning | Processing | Release Job Command | low | as required |
| Planning | Processing | Suspend Job Command | low | as required |
| Planning | Processing | Resume Job Command | low | as required |
| Planning | Processing | PGE Profile Information | low | as required |
| MSS | Planning | Lifecycle Commands | low | as required |
| MSS | Planning | Resource Usage Information | low | as required |
| MSS | Planning | Resource Configuration | low | as required |
| MSS | Planning | Scheduling Information | low | as required |
| Planning | MSS | Processing Status | low | as required |
| Planning | MSS | Current Mode | low | as required |
| Planning | MSS | Detected H/W and S/W Faults | low | as required |
| Planning | MSS | Resource Utilization | low | as required |
| Planning | MSS | Event Notification | low | as required |
| CSS | Planning | Subscription Notification | low | as required |
| CSS | Planning | Common Facilities | low | as required |
| CSS | Planning | Object Services | low | as required |
| Planning | CSS | Subscription | low | as required |
| Planning | CSS | Communication Services Requests | low | as required |
| Planning | Interops | Advertisement Search Request | low | as required |
| Interops | Planning | Advertisement | low | as required |
| Data Management | Planning | Query Results | low | as required |
| Planning | Data Management | Queries for Plans | low | as required |

In the table, where an exact number is unavailable, the data volume is estimated as low (less than 1 MB), medium (between 1 MB and 1 GB), or high (greater than 1 GB) per use defined in the frequency column. The frequency information will be updated as the interfaces are fully defined.

The Planning subsystem interfaces with the Algorithm Integration and Test Tools Computer Software Configuration Item (CSCI) within the Data Processing subsystem for information on Product Generation Executives (PGE). The information that is collected is referred to as a PGE Profile. This includes information on the PGE executable, the input data type(s) it requires, the output data type(s) it generates, the PGE resource requirements - hardware platform, memory, disk storage, etc. The Planning subsystem provides storage for the profiles to permit the Algorithm Integration and Test Tools to insert, retrieve and modify PGE information.

The primary interface of the Planning subsystem with the Data Processing subsystem is to describe the PGEs that need to be run in order to fulfill the production goals. A Data Processing Request (DPR) describes a run of a PGE to the Data Processing subsystem. A DPR describes the specific input and output data, filenames, run-time parameters, dependencies and predicted run-times. The Data Processing subsystem provides status and processing completion information to the Planning subsystem.

The Planning User Interface provides a means of human interaction with the Planning Subsystem. Through this interface an operator can enter Production Requests. A Production Request describes an order for data that is to be produced by the Data Processing subsystem. Production Requests may signify the need for processing of new data (Routine Production Requests, a.k.a. standing orders) or the need for the reprocessing of data (Reprocessing Production Requests). The planning subsystem takes these Production Requests and uses the PGE profile information to work out the Data Processing Requests that will be required to fulfill the Production Request. The system also supports a hierarchy of Production Requests called Production Request Collections. These allow for operator-defined and named groupings of Production Requests. In addition, the system cost (in terms of resource utilization) for a particular collection can be provided. This functionality is especially helpful for large Reprocessing requests (e.g., six months of data). The Planning User Interface is also used to issue commands to initiate plan creation, plan activation and the canceling of a plan, as well as providing reports/status of progress within a plan. Resource planning for the entire DAAC will also be done using the Planning User Interface; the impact of ground events on DAAC resources will be taken into account when developing production plans. To aid in handling inter-DAAC data dependencies, the Planning User Interface will also provide the ability to display another DAAC's plan at the local site. The data dependencies between the two sites will be identified by the system.

The Planning subsystem queries the Data Server holdings for existence of data required for processing. If the data exists, the Data Server responds with granule information (identification, metadata, etc.). The Data Server also provides the subscription services needed by Planning to determine when new data are available for processing. The operator can use a Planning

Subsystem utility to generate the subscriptions to be triggered by the arrival of data at the Data Server. The Data Server sends a notification when data that fulfill the subscription are inserted into the Data Server. Subscriptions are also submitted on Planning Data Availability Schedules (PDAS) and FOS Detailed Activity Schedules (DAS) so plans can be based on more accurate predictions of when data will arrive.

In addition, On-Demand Production Requests are received from the Data Server. On-Demand Production Requests (OPRs) are checked against operator-defined resource usage thresholds. If it exceeds a resource usage threshold, it will be stored in the database, and will need to be added to a production plan just as any other routine or reprocessing Production Request. OPR's that do not exceed the resource usage threshold will be sent directly to the Data Processing subsystem.

The Ingest subsystem also provides a subscription service (via the Ingest Data Server) for notification of the arrival of L0 data from SDPF and EDOS. The ECS Advertising services are used to provide the advertisement data that is required by the Planning subsystem in order to generate subscriptions.

The Planning subsystem uses the Document Data Server within the Data Server subsystem to store production plans. The Document Data Server then makes those plans available to the user community.

The Planning Subsystem also has an interface to the Management Subsystem (MSS). The Planning subsystem is responsible for sending MSS fault management data and resource utilization data. The Planning Subsystem will exchange mode management information, and will receive event notifications from MSS. For Resource Planning, the Planning Subsystem will receive configuration management resource information with which to initialize it's database from MSS as well as the actual times of ground events from the MSS log.

Lastly, the Planning Subsystem interfaces with Data Management to query about plans from other DAACs to support inter-DAAC planning

## 3.2 Subsystem Overview

This section describes the key drivers for the Planning subsystem and how the design responds to them. The section starts with a discussion of the use of Commercial-off-the-shelf (COTS) software within Planning and Data Processing and describes the key changes to the Planning subsystem since the preliminary design specification, brought about by the selection of COTS.

### 3.2.1 Use of COTS within the Planning and Data Processing Subsystems

Between Release A's Preliminary Design Review (PDR) and Critical Design Review (CDR) the Planning and Data Processing subsystems focused much attention on the evaluation and selection

of COTS. Managing systems involving large numbers of processing tasks is not a problem unique to ECS; robust vendor provided software exists to facilitate a high degree of automation in these systems. When carefully selected, COTS significantly reduces both the schedule and technical risks and can provide capabilities that would otherwise take considerable effort to develop. The evaluation and selection process is described in the Data Processing Subsystem Design Specification. It is important to emphasize that the selection of COTS that provides maximum benefits to ECS has resulted in a change to the PDR architecture and design of both the Planning and Data Processing subsystems. This will be outlined in Section 3.2.2.

#### 3.2.1.1 Brief Description of Job Scheduling COTS

The main focus of COTS evaluation within the Planning and Data Processing subsystems has been to select a Job Scheduling Engine to provide the automated production management capabilities required. The COTS that have been selected is a combination of packages called AutoSys and

AutoXpert.

AutoSys provides robust capabilities for managing schedules of Jobs. Jobs are defined within the AutoSys database and are initiated on remote machines when the dependencies of a Job have been met, and the resources required to complete the Job are available. AutoSys basically is comprised of three components:

- AutoSys RDBMS - contains the job definition and dependency information
- Event Processor - reads RDBMS to determine actions to be taken
- AutoSys Remote Agent - initiates a job on a remote processor

The basic production sequence of AutoSys is shown diagrammatically in Figure 3.2-1.

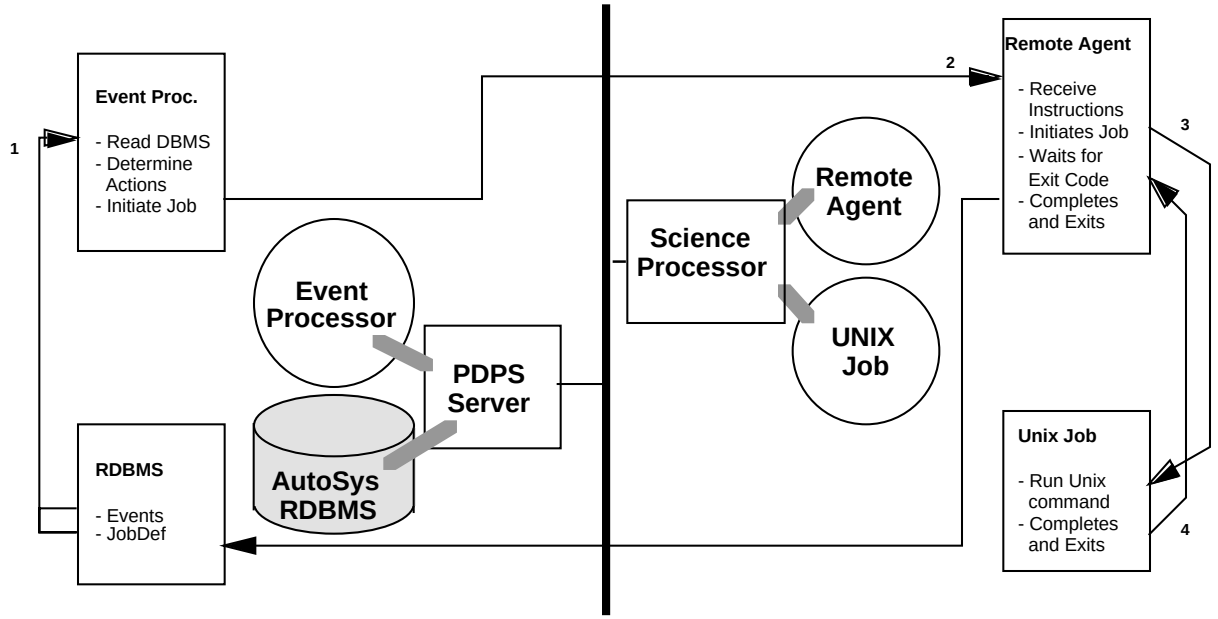



***Figure 3.2-1. AutoSys Production Sequence***

AutoXpert provides advanced capabilities for monitoring the schedule of Jobs and predicting their completion time. The graphic display provided by the AutoXpert provides for three views onto the production schedule (also shown in Figure 3.2-2):

- Timeline or GANTT style view, showing the current status according to schedule and estimated completion times for tasks
- Job Network or PERT style view, showing the current status according to the task dependencies
- Resource style view, showing the current status according to allocation of tasks to machines

AutoXpert also provides capabilities to perform “what-if” scenarios on the schedule to predict the impact of failures within the job network on the downstream dependent tasks.

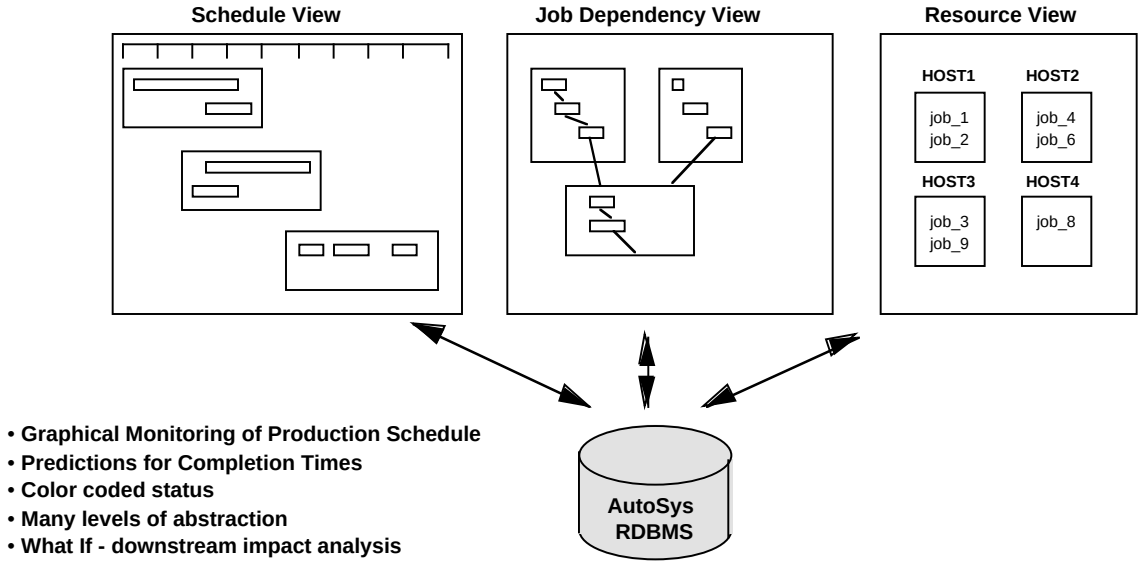



***Figure 3.2-2. Graphical Schedule Displays Provided by AutoXpert***

### 3.2.2 Summary of Changes to the Planning Subsystem

#### 3.2.2.1 From Release A's PDR to CDR

As described in the COTS evaluation paper, the goal of the selection was to obtain the packages which provide the maximum of the capabilities required by the ECS. No COTS solution exists to cover the entire ECS Planning and Data Processing subsystems. The design presented in the rest of this document describes those areas which are ECS specific or areas where the requirements can not be covered by COTS. In the area where the requirements can not currently be covered by COTS it should be emphasized that the ECS is also actively encouraging the vendors to improve their capabilities.

The basic capabilities of the Planning subsystem described in the preliminary design have not changed. Rather, these capabilities have been enhanced by the AutoSys/AutoXpert selection. What has changed is the division of responsibilities within the Planning and Data Processing subsystems. A major goal in re-designing the system after the selection was to ensure that the interfaces to COTS are within one subsystem. This ensures that interfaces to the COTS can be appropriately encapsulated to give later flexibility augmenting, modifying or replacing the underlying COTS as ECS matures.

The Job Scheduling COTS is appropriately made a component of the Data Processing subsystem, which accounts for the reallocation of some of the Planning capabilities. The key changes to the Planning design since PDR are within the Production Management capabilities. The two main activities performed here were:

- To coordinate the production by providing a Data Processing Request to the Data Processing subsystem when all the data required for the task are present at the local Data Server
- To provide a display of the active production, and its status according to the plan

These two capabilities have been split into separate CSCs. The (new) subscription manager CSC, which is part of this subsystem, provides the first of these capabilities. The graphic capabilities of AutoXpert in the Data Processing subsystem provides for the second.

Other changes that have been brought on due to the selection of AutoSys and AutoXpert pertain to the sequence of events “activating” a plan. This now involves rolling a portion of the “long term” plans generated in the Planning subsystem into AutoSys. The design and scenarios that describe this procedure are presented in Section 4 of this document.

#### 3.2.2.2 From Release A CDR to Release B IDR

In terms of the overall architecture, no major changes are planned to be made from Release A to Release B. As was stated earlier, Release B will continue to monitor developments of the COTS products, and will influence to the best of our ability the direction the vendors head.

In terms of added functionality, the following major enhancements have been made:

- Acceptance of On-Demand Production Requests from the Data Server. We have added a new CSC, the On-Demand Manager, which receives these requests and forwards them on to the Data Processing Subsystem

- The ability to compare plans from one DAAC to another, emphasizing the data dependencies involved.
- Limited automatic replanning. This is the ability to have the software automatically prompt the operator based on certain configurable events which may indicate that the plan and actual status are drifting apart.
- Distribution and use of Planning Data Availability Schedules (PDAS). These are schedules of when a site predicts it will make data sets available. They are used by other DAACs in generating local plans.
- Use of the FOS Detailed Activity Schedule (DAS) to predict when L0 data from EDOS will be available, and to capture any necessary instrument mode information.
- The ability for an operator to define and use simple production strategies. These strategies will allow specification of priorities and resources to help control processing.
- The ability to construct Production Request Collections from existing Production Requests and to provide an estimated cost of any single or collected Production Request. This feature is especially useful for large Reprocessing Requests (e.g., 6 months of data).
- Additional Resource Planning capabilities which will incorporate inter-DAAC resource scheduling concerns.

#### 3.2.2.3 From Release B IDR to Release B CDR

No major additions or modifications were made to the Planning subsystem architecture during this period. As was stated earlier, ECS will continue to monitor developments of the COTS products, and has influenced, to the best of our ability, the direction the vendors have gone.

All of the new Release B functionality items listed in the previous section have be refined in their design, especially:

- Production Rules (see section 3.2.3.1.2 for description). Most of the functionality to implement production rules has been added to the Production Request Editor and the Subscription Manager. In addition, the AITTL CI of Data Processing also has substantial changes to allow for the entry of the new production rules into the PDPS system.
- Limited Automatic Replan. The types of limited automatic replan have been defined and incorporated into the design. The operator-defined limits for these checks are configured in the Planning Workbench UI while the actual check against the particular threshold will be made by the Subscription Manager and On-Demand Manager CSCs.
- Resource Planning. There has been a lot of activity in this CSC based on feedback from the Ops Concept Workshop. Most of the changes have been in the Release A implementation. The design documented in section 4.1.7 has incorporated these modifications and the new Release B features.
- FOS Data Availability Schedule (DAS). We have refined our approach to extracting the necessary collection times and mode information from this schedule. There are a number of new objects to help us do this.

### 3.2.3 Key Design Drivers

As outlined in the overview the planning subsystem has three main responsibilities:

- Defining the goals for data production,
- Preparing plans for production, and
- Coordinating the production from those plans.

Each of these areas are discussed in turn in order to describe how the subsystem responds to the key drivers in these areas.

#### 3.2.3.1 Defining Production Goals

##### 3.2.3.1.1 PGE Profiles

The Planning subsystem stores a description of the PGE in order to be able to determine the correct PGEs and associated input required to generate a product. Each PGE has individual data requirements. These data requirements are captured within the PGE Profile to allow the Planning subsystem to select the appropriate data for a PGE either from the Data Server (in the case of reprocessing) or from its own predictions of data availability (in the case of routine production). The PGE profile is a key component of the Planning subsystem that allows the Planning subsystem to determine which PGEs have to be run to create a given data set (the term Data Processing Requests is used within the ECS to describe an individual run of a PGE).

##### 3.2.3.1.2 Production Rules

There is a fair degree of diversity in the scheduling characteristics of PGEs that are being developed for ECS. The most standard category of PGE describes those that require a well defined period of data as input and produce one or more output data products. For example, the CERES subsystem 1 geolocation and earth radiance calibration PGE requires one day's worth of L0 data and ephemeris data and produces a single product file of bi-directional scan data, as well as 24 files of instrument earth scans, each containing 1 hour of data. However, there are many other categories with more complex scheduling requirements (PGEs scheduled per orbit, PGEs that mosaic data into geographical tiles, PGEs with alternative data sets that are determined at run time). These types of PGE have been characterized for the ECS through the work of the Ad Hoc Working Group on Production (AHWGP) and are described in terms of production rules. All of the production rules identified at the Production Rules Workshop will be supported within the Planning subsystem at the AM-1 Release of ECS. If additional production rules surface at a later date, we will be able to incorporate them by using specializations of the various production rule objects.

##### 3.2.3.1.3 Production Requests

The ECS supports both routine production and ad-hoc or on-demand production. The goal of ECS is to produce science data sets to facilitate earth science research. Consistently generated data spanning long time frames are required for much of this research; therefore, many of the tasks performed within the data processing subsystem will be highly routine with the same task being performed (albeit on different data) a large number of times per month. As described above, the formulation of a Data Processing Request is a non-trivial task, and therefore a high degree of automation is required to aid in this process. The Production Request is a key abstraction which

describes an “order” for the generation of a data set over a period of time. The Production Request is used along with the PGE profile to generate the Data Processing Requests; typically one production request will generate many data processing requests. The Planning subsystem will determine a default PGE to generate this product; however, this predetermination may be overridden within the Production Request. Similarly, user parameters may be provided to override their defaults defined within the PGE profile. In addition, the Planning subsystem provides the capabilities to review and modify the individual Data Processing Requests created from the Production Request to ensure that the correct task is defined to achieve the production goal. Not all Production Requests have to be put into production, Production Requests can be defined to describe backup options for processing. On identification of a problem within the production chain, the tasks defined for a backup Production Request may be quickly scheduled.

#### 3.2.3.2 Preparing Plans for Production

##### 3.2.3.2.1 Planning

The ability to generate production plans is important to both operations and users for the following reasons:

- *Staffing Projections*—A long term projection describing the data products that will be generated at a site allows organizations to manage staffing to support the anticipated production, for example, to cover on-site or off-site QA requirements.
- *Disconnects*—Dependencies on data not planned for production can be identified, which then allows corrective steps to be taken.
- *Efficiency*—Proper organization of the data production activities within a plan ensures that the production will utilize resources in an efficient manner. In the AM-1 release, the operator will be able to enter simple production strategies to control this organization.
- *Inter-DAAC data dependencies*—Allows coordination of processing throughout the DAACs to ensure that there are no data dependency conflicts in the plans.

There is no explicit planning horizon that is hard coded into the Planning subsystem software. The nominal Operations Concept is to plan 30 days in advance. However, this may prove to be unreasonable at DAACs with a large number of Data Processing Requests, and therefore they will likely choose a shorter horizon.

#### 3.2.3.2.2 Publishing Plans

The Planning subsystem provides the capability to publish plans that are generated at a site. The Document Data Server is the most appropriate repository for these plans, since the plans can then be made available to WWW access. The published plans will also be stored within a WAIS database so that, using a simple HTML forms interface, a user may select the plans for a specific period. A small set of metadata will describe the plans, to allow the user to select a plan based on period, data product types which are scheduled in the plan, as well as operator comments. A number of plan formats will be stored within the Document Data Server including a simple ASCII report as well as a more electronically parsable format. For Inter-DAAC planning, the published plan will also include an exported plan, that will allow one site to import and view plans from other DAACs. The ability to include meaningful graphics within another possible format will be investigated. The ability to subscribe to plans is supported so that users may be automatically alerted to delays in production.

### 3.2.3.3 Managing the Production

#### 3.2.3.3.1 Schedule/Data Driven Production

The plans generated in the Planning subsystem describe a prediction of the activities that will occur in the Data Processing subsystem. This prediction is only a forecast of the activities that will occur and a recipe for the production under nominal conditions. It is important to emphasize that the production isn't driven directly from the plan. As an example, if a PGE is predicted to start at 3.00 in the afternoon it doesn't mean that the PGE will be executed in the Data Processing subsystem at 3.00 precisely. The Planning subsystem always ensures that the data are present within the system before an activity is released to run. The Data Processing subsystem ensures that all of the required resources are available to permit that activity to complete. Jobs are queued within the Data Processing subsystem so that (within the constraints of priority and job dependencies) the resources will be kept efficiently occupied.

#### 3.2.3.3.2 Using the Plan

There are many events that can impact a plan: delays in data arrival, poor performance of system components, etc. There is no need to replan in circumstances like these. The Planning and Data Processing architecture and design are robust to many fault situations. A replan is only really required when the production goals need to be redefined to accommodate an unanticipated circumstance. For example, a repeated failure within the science data processing might indicate that it would be appropriate to switch from routine production to concentrate on some reprocessing objectives.

As previously stated, plans generated in the Planning subsystem describe a prediction of the activities that will occur in the Data Processing subsystem. The system is tolerant to changes in schedule, but these changes usually indicate a circumstance that needs to be brought to the attention of the operators. The forecasts provided by the Planning subsystem are used within the Data Processing subsystem to define alarms to alert the operator of deviations against the schedule. Delays in the arrival of data, which may not provoke any exception in the system that would be identified as an error, will thus still be noticed and flagged.

#### 3.2.3.3.3 Bypassing the Plan

It is understood that, especially during early mission activities, it is frequently desirable to be able to schedule jobs individually. Operators require the capability to insert a job into the stream without necessarily planning this task. The net result of inserting a job into the production stream in an ad-hoc manner is that the forecast times for completion as determined by Planning may no longer be accurate. However, the benefits that this capability bring are clear. The planning subsystem will permit authorized operations staff to enter a Production Request and schedule the resulting Data Processing Requests to the Data Processing subsystem.

The other major method of bypassing the Plan is via the On-Demand request mechanism from the Data Server. These requests are automatically handled by the Planning subsystem, turned into Data Processing Requests, and sent to the Data Processing subsystem. Normally, no operator intervention is necessary. These requests are first validated and then checked against operator defined resource thresholds. If the request exceeds a resource threshold, it is deferred until the next planning period so its resource usage can be better accounted for. In any case, status information is passed back to the Data Server. In general, On-Demand requests will not completely throw off the plan with respect to actuals, since the system takes On-Demand requests into account during candidate plan creation by inserting dummy requests which consume resources proportional to the expected load. In addition to this, all Data Processing Requests, including On-Demand requests, are tracked by the Planning subsystem as they occur.

### 3.2.3.4 Distribution of Planning Capabilities

#### 3.2.3.4.1 SCF access to Planning Capabilities

It is acknowledged that, the definition of Production Requests - specifying the data processing goals - is tightly coupled to science objectives of calibration and validation of instruments and algorithms, especially during early mission life. Since the responsibility of production ultimately resides with the Distributed Active Archive Center (DAAC), there is no intent to distribute the Planning subsystem capabilities away from the DAAC. However, to encourage the collaboration between the different teams with stakes in the production of science data, the Planning subsystem capabilities can be made available by remote access. Kerberized remote login procedures for special users and limited access permissions to the Planning subsystem database tools will ensure that DAACs can maintain secure authority over use of these capabilities.

#### 3.2.3.4.2 Inter-DAAC Data Dependencies

The Planning subsystem also needs to account for data dependencies between DAACs. For this reason, the ability to export and import plans from other DAACs is available. Once a plan is imported by a remote DAAC, data dependencies are identified and any conflicts can be noted. When plans have been generated that are acceptable to all parties, these plans can be baselined; future plans created will be compared with this baseline to report on how this plan meets all inter-DAAC commitments. This allows the DAACs to coordinate production so that all production goals can be met in a timely manner.

### 3.2.4 Planning Subsystem Use of Key Design Mechanisms

The Planning Subsystem uses a number of the key design mechanisms developed by ECS to encapsulate much of the interprocess communication and to handle the infrastructure services. Inheriting from or reusing this code saves on design and programming effort and increases maintainability.

#### 3.2.4.1 Managed Process Framework

All Planning applications use the managed process framework to handle mode management, MSS agent interface and the OODCE infrastructure interface. Utilities, such as the Production Planning Workbench, fall under the category of 'clients', and inherit from the 'EcPfClient' object; they do not need as much control as a server process because they are not executing continuously and do not use complicated interprocess communications. The Subscription Manager and the On-Demand Manager are managed servers and inherit from the 'EcPfManagedServer' object. They are started, stopped and monitored by MSS. Errors or faults within the managers are automatically handled. These lifecycle services ensure that any subscription notifications or on-demand production requests are handled at all times.

#### 3.2.4.2 Event Handling

Another key mechanism used by all the CSCs within Planning is event handling, which is encapsulated by the EcEvent object. Events include abnormal conditions (errors or faults) as well as normal occurrences. Not all events that occur within the system are of interest from a system operations and management perspective; those that are are called **managed events**, which must be reported and logged. This mechanism handles any messages that require operator notification with the Planning Subsystem, as well as notification to a centralized MSS event logging facility.

#### 3.2.4.3 Server Request Framework and Message Passing

Both the Subscription Manager and the On-demand Manager use the interprocess communication provided by the key mechanisms. The On-demand Manager uses the Server Request Framework to provide management capabilities in the form of modification, deletion and status of the On-demand Production Requests. The subscription manager receives subscription notifications from the Data Server using message passing, a subset of the Server Request Framework. Callbacks in the managers, which handle messages from other subsystems, inherit much of their code from the 'EcMpMsgCb' object.

#### 3.2.4.4 Advertisements and Subscriptions

The Advertising Service within ECS allows the different service providers within ECS to advertise their services and products. Subscriptions to these services can be submitted to allow outside systems to be aware of changes to the system. The Planning Subsystem uses the Advertising Service to find these services for the various data types it needs as input. These advertisements are then used to create subscriptions on these data by the Production Request Editor and the On-demand Manager, as well as the Subscription Editor, to allow the Subscription Manager to be notified when input data becomes available. The Subscription Manager uses these notifications to determine which Data Processing Requests are ready to begin processing. This process is only done when a new subscription needs to be added, not every time.

#### 3.2.4.5 Universal References

Universal references are the key mechanism used by ECS to locate services and products; they are encapsulated by the object 'EcUrUR'. These references allow different parts of ECS to communicate about the services and products they provide. Universal references for data granules are used by the Production Request Editor, the On-demand Manager, the Subscription Manager, and the Data Processing Subsystem to communicate where a product can be found. The Subscription Editor, the Production Request Editor, and the On-demand Manager use universal references to set up subscriptions on data collections within the Data Server. Universal references are also the result from queries to the Data Server and to the Data Management Subsystem performed by the Planning Workbench, the Production Request Editor and the Subscription Manager.

#### 3.2.4.6 Request Status Tracking

The On-demand Manager also uses the request status tracking mechanism, encapsulated in the EcRequest object. This key mechanism allows a client (a user or another subsystem) to receive the status of a request, such as an On-demand Production Request, at any time.

### 3.2.5 Subsystem Structure

The Planning subsystem consists of two CIs: the Production Planning CSCI and the Production Planning HWCI. The Planning CSCI consists of software to meet the subsystem requirements. The Planning HWCI consists of the hardware resources to support the Planning software.

This page intentionally left blank.

# 4. PLANG - Production Planning CSCI

## 4.1 CSCI Overview

The Production Planning CSCI consists of a number of utilities and server applications as shown in Figure 4.1-1. The distinction between these two classes of programs is useful when describing the software design:

- Utilities are programs that perform well defined tasks and are invoked at the operator's requests. A utility is usually exited when the task is complete; the system does not depend on the utility being active at any given time. As an example, a mail editor is a common utility program, invoked whenever an operator wishes to generate a mail message.
- Servers are programs that perform ongoing tasks and have to be active at all times for the subsystem to carry out its allocated functions. An example here is the mail daemon; the mail won't be delivered unless the daemon is running.

The allocation of the Planning subsystem capabilities to distinct applications accounts for the need to provide distinct interfaces for distinct activities, and to restrict access to planning functions to classes of users. The applications define an integrated set of tools with well-defined functions to supplement the COTS components. The allocation of capabilities to applications within the Planning CSCI also accounts for the need to permit independent development of the components as the capabilities of the ECS evolve. The hardware on which the various applications will run is shown in Figure 4.1-2 and the purpose of each application is briefly outlined below.

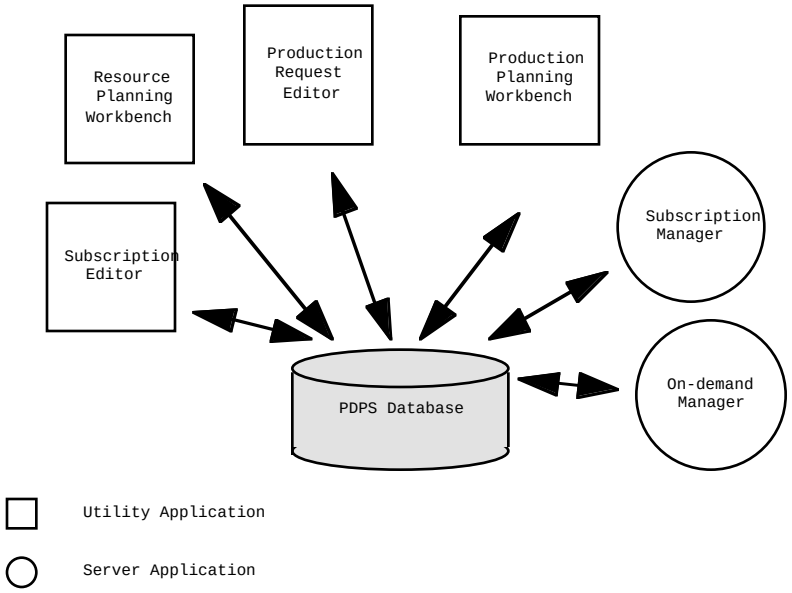



***Figure 4.1-1. CSCI Overview***

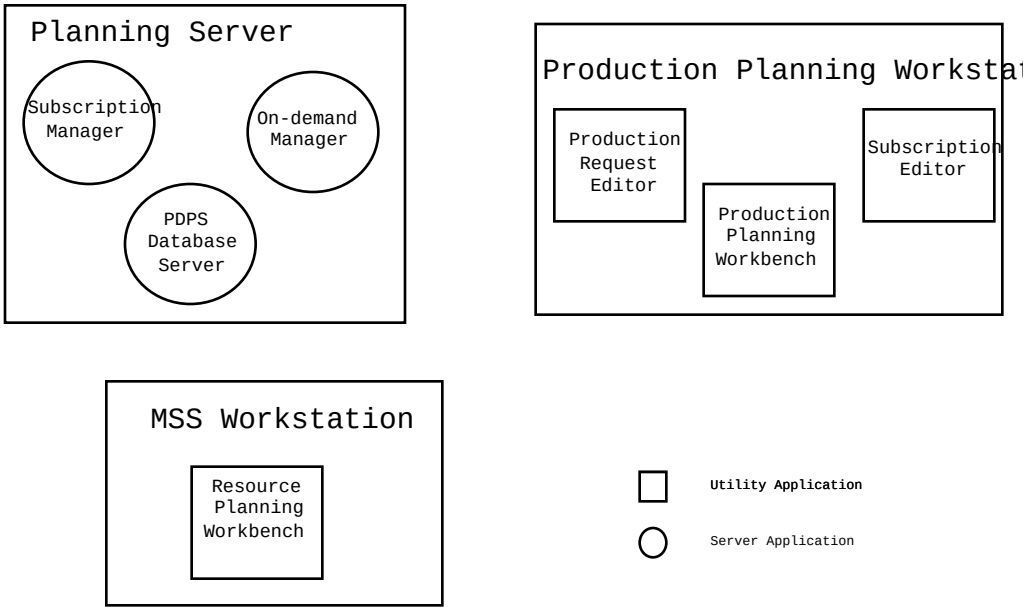



***Figure 4.1-2. Planning SCSI Component Diagram***

### 4.1.1 PDPS Database

At the heart of the Planning CSCI is the Planning and Data Processing System (PDPS) Database (and the implied database server). This provides for the persistent storage of data and facilitates the sharing of this data between the applications. The database server affords built-in security, fault tolerance, and a locking mechanism for managing concurrent access to data. The objects which are built from the persistent data within the PDPS database are indicated within the object design. The schema for the database has been derived from the object model and is described in the SDPS Database Design and Schema Specifications CDRL.

Access to the PDPS Database from the application code is via special database interface classes. Complete definitions of these classes may be found in the Release B SDPS Data Processing Subsystem Design Specification for the ECS Project.

### 4.1.2 Production Request Editor

This application allows the user to submit production requests that describe the data products to be produced. The application uses the PGE descriptions (profiles) entered during AI&T in order to work out the tasks - Data Processing Requests - that in sum meet the request. The application provides the capabilities to add, modify, and delete Production Requests, as well as review and modify the resulting Data Processing Requests. The Production Request Editor is identified as a part of the planning workbench although the process of defining production requests will be a discrete activity, unrelated to the “planning” of these events. Multiple Production Requests can be grouped together in a Production Request collection, especially in the case of performing related reprocessing requests.

The Production Request Editor may also be used by authorized operations staff to schedule Data Processing Requests derived from a production request directly to the Data Processing subsystem.

The Production Request Editor is one of the CSCs that plays a large part in implementing production rules. See Section 4.3.2 for the objects that participate in this.

### 4.1.3 Production Planning Workbench

This application is used to prepare a plan for the production at a site, and forecast the start and completion times of the activities within the plan. These functions provided by the workbench include the following high-level activities:

- Candidate Plan Creation—from the production requests prepared by the Production Request Editor
- Plan Activation—activating a candidate plan
- Updating the Active Plan—feedback from the processing into the active plan
- Canceling/Modifying the Active Plan
- Publishing and Importing Plans - to share plans between DAACs and with the outside world
- Cross-DAAC planning—identifying data dependencies between DAACs

As described previously, activating a plan entails rolling a portion of a selected plan into the AutoSys COTS. This “schedule” is then managed within the Data Processing subsystem. The forecast times generated within the planner are used to set up operator alerts that would make the operator aware of gross departures from the predicted plan. The production planning workbench can periodically update it's predictions using feedback from the AutoSys.

One of the new capabilities of the Production Planning Workbench is cross-DAAC planning. This will allow a production planner at one DAAC to display data dependencies with other DAACs and to identify any conflicts. This will essentially allow the production planners to view multiple DAAC plans as one plan. The priorities and scheduled execution times of jobs causing conflicts can be noted and the DAACs can work together, and if necessary, with the SMC, to resolve the conflicts.

In addition, the Planning Workbench has a number of configuration items that allow the Production Planner to control production. Production Strategies, entered by the Production Planner at a high level, allow the Workbench to determine the priorities of the individual runs of the PGE. Limited Automatic Replan criteria entered here determine when the Production Monitor should be alerted that a replan should be done. On-demand resource thresholds allow the Production Planner to throttle on-demand production requests received so they don't overly impact the production plan.

### 4.1.4 Planning Subscription Editor

This application provides the capabilities required to submit subscriptions to the Data Servers responsible for the storage of ingested data. Registration of a subscription at a Data Server is required for the Planning CSCI to receive notification when data arrive within the ECS. In previous releases, the submission of subscriptions was managed as an operator initiated activity. This has now been mostly automated, and only manual subscriptions for PDASs from other DAACs and for the FOS DAS are now required. These both need only be done once, however, and the Subscription Editor will remain as a tool to view the subscriptions that are in place.

### 4.1.5 Subscription Manager

The Subscription Manager is used to manage the receipt of subscription notifications from the Data Server. These notifications inform Planning of the arrival of input data required for PGE execution, Detailed Activity Schedules for predicting data arrival from EDOS, and Planning Data Availability Schedules for predicting data arrival from other DAAC's. The Subscription Notification contains Universal References (URs) which are pointers to the data objects stored on the Data Server.

The Subscription Manager updates the PDPS database to indicate data availability. When all input data for a Data Processing Request is available, the job defined for that Data Processing Request is released for execution within the Data Processing subsystem.

Data objects that are not associated with data granules (e.g. PDAS' and FOS Detailed Activity Schedules) are acquired from the Data Server by the Subscription Manager. When a schedule is received, the associated PlDataAvailabilityTimes object is created indicating predicted arrival times of data needed at the DAAC.

The Subscription Manager is one of the CSCs that plays a large part in implementing production rules. See Section 4.3.10 for the objects that participate in this.

### 4.1.6 On-Demand Manager

The On-Demand Manager is used to manage the on-demand production requests received from the Data Server. Upon receipt of a production request the On-Demand Production Request class verifies that resource usage thresholds are not exceeded and that the data is available to generate the product. If all criteria is satisfied then the production request is added to the list of On-Demand requests maintained by the manager, the processing request is submitted to the Data Processing subsystem and the product is produced. Based on certain threshold values a replan may be triggered. Additional possible On-Demand notifications are a request to modify or cancel a production request or a request for the status of a production request.

### 4.1.7 Resource Planning Workbench

The Resource Planning Workbench allows the Resource Planner to submit reservations for hardware resources and allocate and plan their use. Reservations may be made for activities that include major ECS data processing services as well as ground events. Ground events include activities such as maintenance, test, and training. The hardware resources include CPUs, storage devices, (sub)networks, as well as platforms that are made up of CPUs and storage devices. A list of resources is configuration managed by MSS. Reports of planned versus actual resource usage can be generated and timelines for the resource plans may be displayed. This application is

separate and distinct from the Production Planning Workbench. Its plan, however, is used as input to the Production Planning Workbench, so that production plans can take into account planned resource usage.

## 4.2 CSCI Object Model

The CSCI object model is shown in a number of views; Figures 4.2-1 to 4.2-13. Each view aims to capture a different aspect of the CSCI capabilities. Classes will appear in several views, only the operations and attributes pertinent to the capability being illustrated are included in the figures in order to reduce confusion. Complete class descriptions are provided in text in Section 4.4.

### 4.2.1 PGE Profile View

The PGE profile view (Figure 4.2-1) describes the classes which are provided to the Algorithm Integration and Test Tools to allow the entry of information that describes a PGE to the Production Planning CSCI.

The key classes introduced within this model are:

- PlPGEProfile: This class defines the collection of information that describes a PGE to the Planning subsystem.
- PlPGE: This is the base class within a generalization hierarchy that describes PGEs. The class defines abstract operations required for the planning subsystem to determine when a PGE needs to be scheduled, as well as containing the key attributes defining the PGE.
- PlPgeFactory: This is a class used to create derived PlPGE objects without knowing the subclass at instantiation.
- PlDataScheduled: This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined directly from the period of some primary input data type. This object participates in the implementation of Production Rules.
- PlTimeScheduled: This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined from some regular time period, such as an hour, a day, a week, a month etc. This object participates in the implementation of Production Rules.
- PlOrbitScheduledNB: This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined based on spacecraft orbits. This object participates in the implementation of Production Rules.
- PlOrbitModelNB: This class is used by PlOrbitScheduled to determine the approximate extent of the orbit to be used. This object participates in the implementation of Production Rules.
- PlTileScheduledNB: This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined based on tile (spatial) criteria. This object participates in the implementation of Production Rules.
- PlCluster: This class is used by PlTileScheduled to determine a group of tiles arranged over the earth. A cluster of tiles will be scheduled together to optimize disk usage. This object participates in the implementation of Production Rules.

- PlTile: This class is used by PlTileScheduled to determine the spatial coordinates of a single tile to be processed by a Tile Scheduled PGE. This object participates in the implementation of Production Rules.
- PlDataTypeB: This class describes the data types used or output from PGEs. The PlDataType class can be thought of as a proxy to the Earth Science Data Type of the Data Server. The class contains the information required by the Planning subsystem to describe the inputs and outputs of a PGE.
- PlDataTypeReq: This describes the correlation between a PGE and the input data type. This class contains the data that allows a PGE to select the required granules of a given input type to perform a production task.
- PlAlternateNB: This specialization of the Data Type defines an alternate input data type for the PGE. It is used for PGEs that require only some of a long list of inputs, or a PGE that desires a secondary choice if one of its primary inputs does not exist. This object participates in the implementation of Production Rules.
- PlOutputYield: Specifies the recipe to describe the output data granules for a PGE.
- PlUserParameters: Describes any user defined parameters associated with a PGE.
- PlResourceRequirements: This class contains a description of the resource requirements of a PGE, which may be matched against the resource configuration of the processing hardware known to the Planning subsystem.
- PlPerformance: This class describes the performance statistics of a PGE. These performance statistics are established at AI&T. The class also contains attributes to describe the statistics updated from the Data Processing subsystem after PGE execution.

Since the data for these classes is established at AI&T, interactions for this view are described in Section 6 of the "Release B SDPS Data Processing Subsystem Design Specification for the ECS Project".

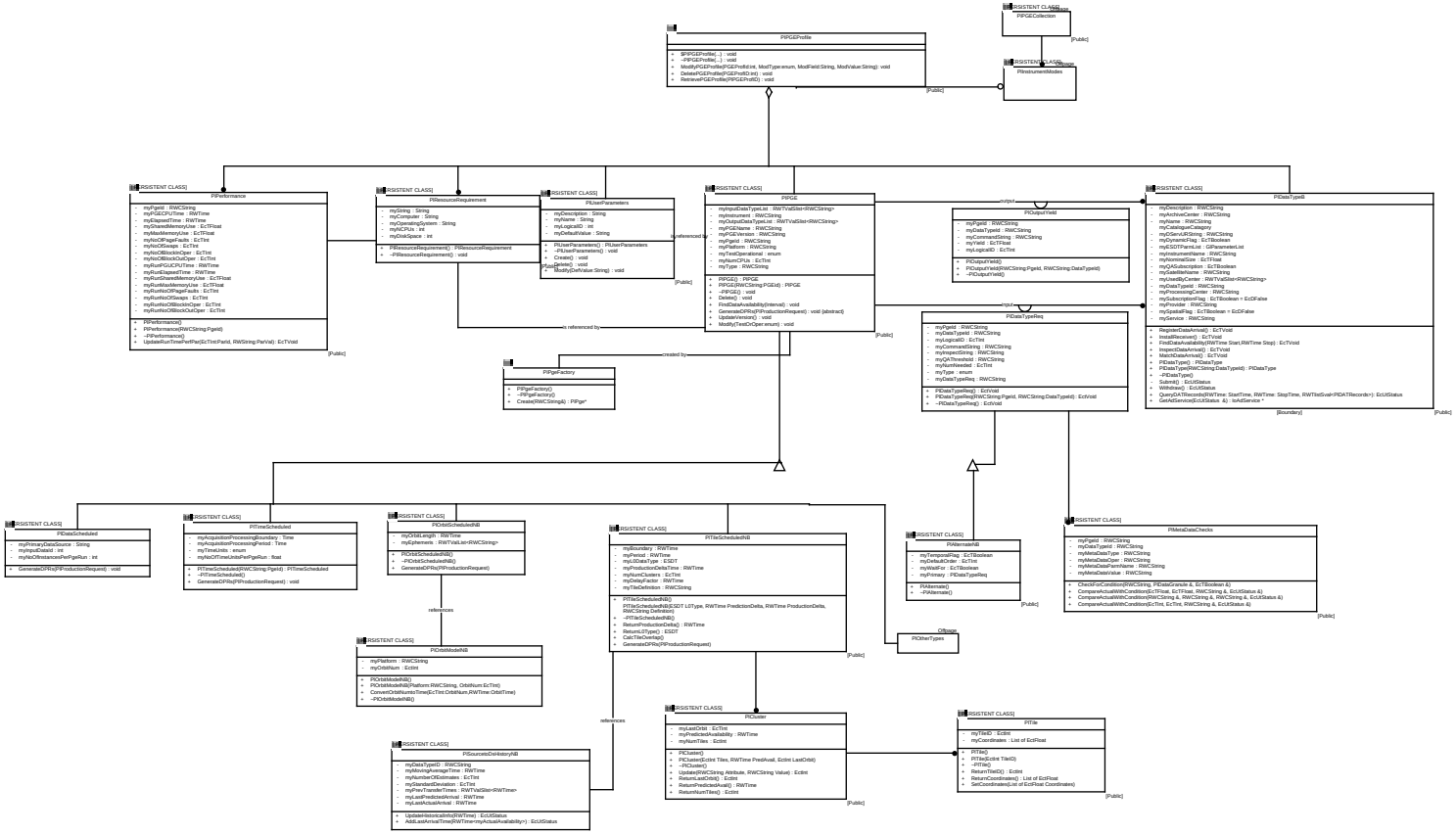



*Figure 4.2-1. PGE Profile View*

### 4.2.2 Production Request View

This view (Figure 4.2-2) describes the classes used during the entry of a production request and the subsequent process by which the Production Request is translated into Data Processing Requests.

The following key classes are introduced:

- PlProductionRequest: A production request describes an order for data products. A production request typically specifies a request for a data product to be produced for an extended period of time (e.g. a month's worth of some product).
- PlProductionRequestUI: This class provides the methods to support the user interface for entry/modification/deletion of production requests.
- PlPRCollectionNB: This collection class contains a group of production requests. This will allow for any number of Production Requests to be specified and managed in a convenient manner.
- PlOnDemandPRNB: This specialization of the PlProductionRequest contains attributes and operations specific to On-Demand or Ad hoc requests.
- PlDPRB: The DPR describes a single run of a PGE and has associations to the data granules required for that Data Processing Request.
- PlDataGranule: This class describes a data granule to the planning subsystem. This object is in effect a proxy to the real data granules within the ECS. The class describes the information that is needed in the Planning subsystem to allow selection of that granule by a PGE for a Data Processing Request. The PlDataGranule describes either data that is actually in the system or a prediction of data that will come into the system.
- PlAlternateDataGranule: This association class describes data granules that are alternate inputs for a PGE. Input data granules are either required (always used) or alternate. Alternate inputs are either primary (first choice) or backup (secondary choices in preference order). This object participates in the implementation of Production Rules.
- PlDataType: This class describes the data types used or output from PGEs. The PlDataType class can be thought of as a proxy to the Earth Science Data Type of the Data Server. The class contains the information required by the Planning subsystem to describe the inputs and outputs of a PGE.
- PlDataTypeCatalogue: This class describes a collection that lists the data types that may be generated by the production system. The class is constructed from the PlDataTypes class using the catalogue category attribute from that class.
- PlDataTypeReq: This class contains the data that associates a PGE to an input data type. It is also a base class for the derived PlAlternate class used for alternate input types.
- PlAlternate: This class is derived from the PlDataTypeReq class. It describes input data types for PGEs that expect alternate inputs. This object participates in the implementation of Production Rules.
- PlOutputYield: Specifies the recipe to describe the output data granules for a PGE.
- PlDATRecord: Contains the predicted availability for a particular data granule received from a source outside of the local DAAC.

- PlDataSource: This class is the base class that provides the methods for predicting when external data will arrive within the ECS, for example, from SDPF or National Oceanic and Atmospheric Administration (NOAA). The Data Sources are specialized to describe the different ways that predictions are obtained. This object participates in the implementation of Production Rules.
- PlDataSourceFactory: This class provides methods to create the derived PlDataSource classes from a DatatypeId without directly knowing the subclass class at instantiation time. This object participates in the implementation of Production Rules.
- PlMetaDataChecks: This class provides methods to perform conditional checks of metadata for input data granules for a PGE. This object participates in the implementation of Production Rules.
- PlRoutineArrival: This class is a specialization of the PlDataSource class and describes routinely arriving data sets. This class contains the attributes and operations required to describe routine ingest of external data. This object participates in the implementation of Production Rules.
- PlSubscriptionSubmitIF: This class encapsulates all the methods used for subscription submittal and withdrawal.

The interactions of these classes are described within the following scenarios:

- 4.4.1 Production Request
- 4.4.2 Data Availability

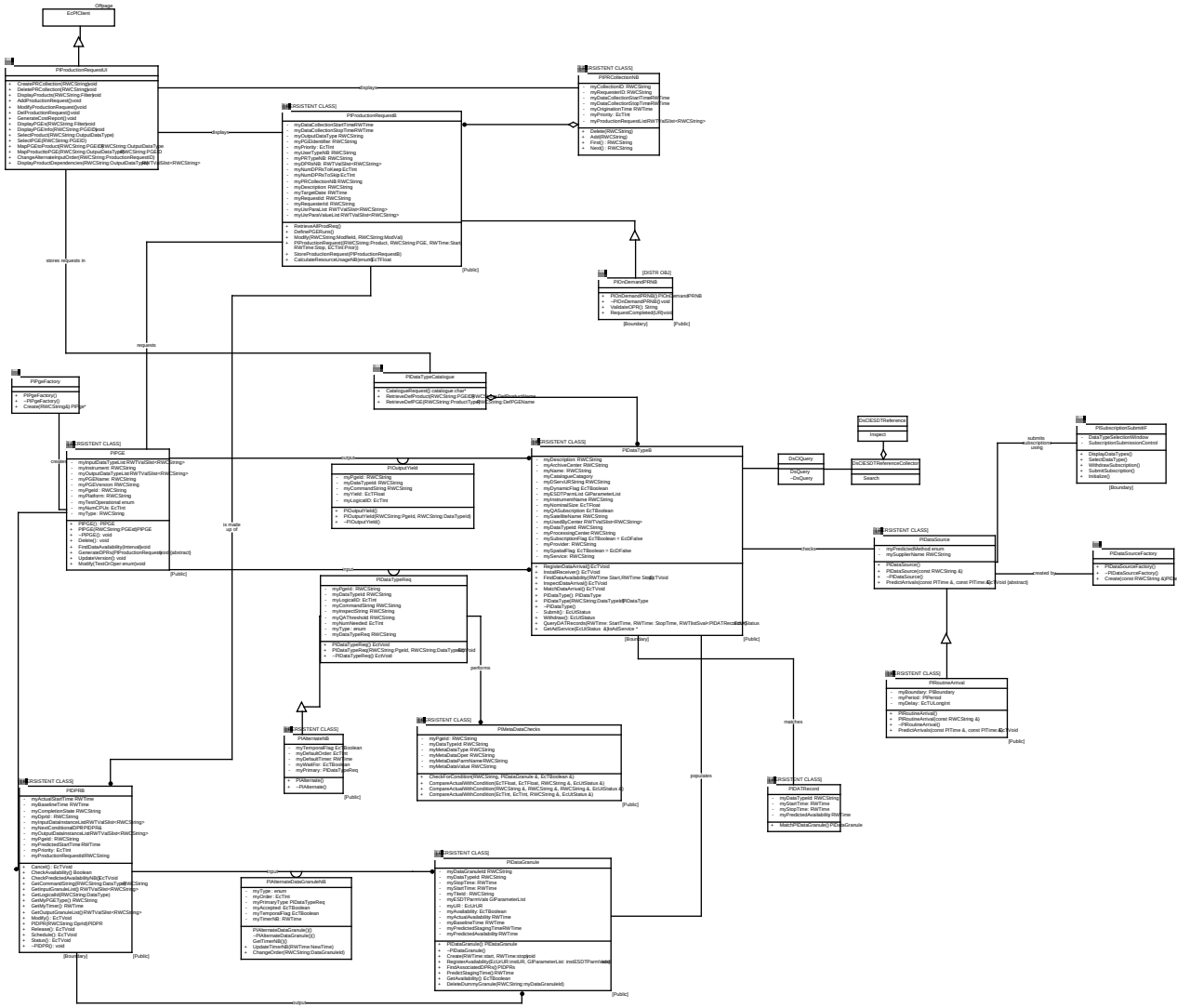



*Figure 4.2-2. Production Request View*

### 4.2.3 Subscription Submission View

This view (Figure 4.2-3) describes the classes used during the generation, submission and withdrawal of Subscriptions for the notification of data arrival. The diagram essentially describes the key components of the subscription submission application.

The classes that are introduced in this model are:

- PlSubscriptionSubmitIF: This class is an abstraction for the user interface to the subscription submission application. The interface will be developed with a suitable GUI builder tool. See Section 4.5.3 for a fuller description of this user interface.
- IoAdProductSearchCommand: This class is used to submit commands to the advertising service.
- IoAdProductList: This list contains the results from a query on the advertising service.
- IoAdService: This is a pointer to a service on a product within ECS.
- DsClSubscription: This class is used to register and withdraw subscription in the Data Server. The constructor for this class is overloaded to permit the specification of the callback queue on which the generator of the subscription is notified of events which fulfill the subscription.
- PlDataType: The status of the subscription is recorded within the planning Data Type abstraction through modification of the subscription flag attribute.

The interactions of these classes are described within the following scenarios:

- 4.4.3 Subscription Submission
- 4.4.4 Subscription Withdrawal

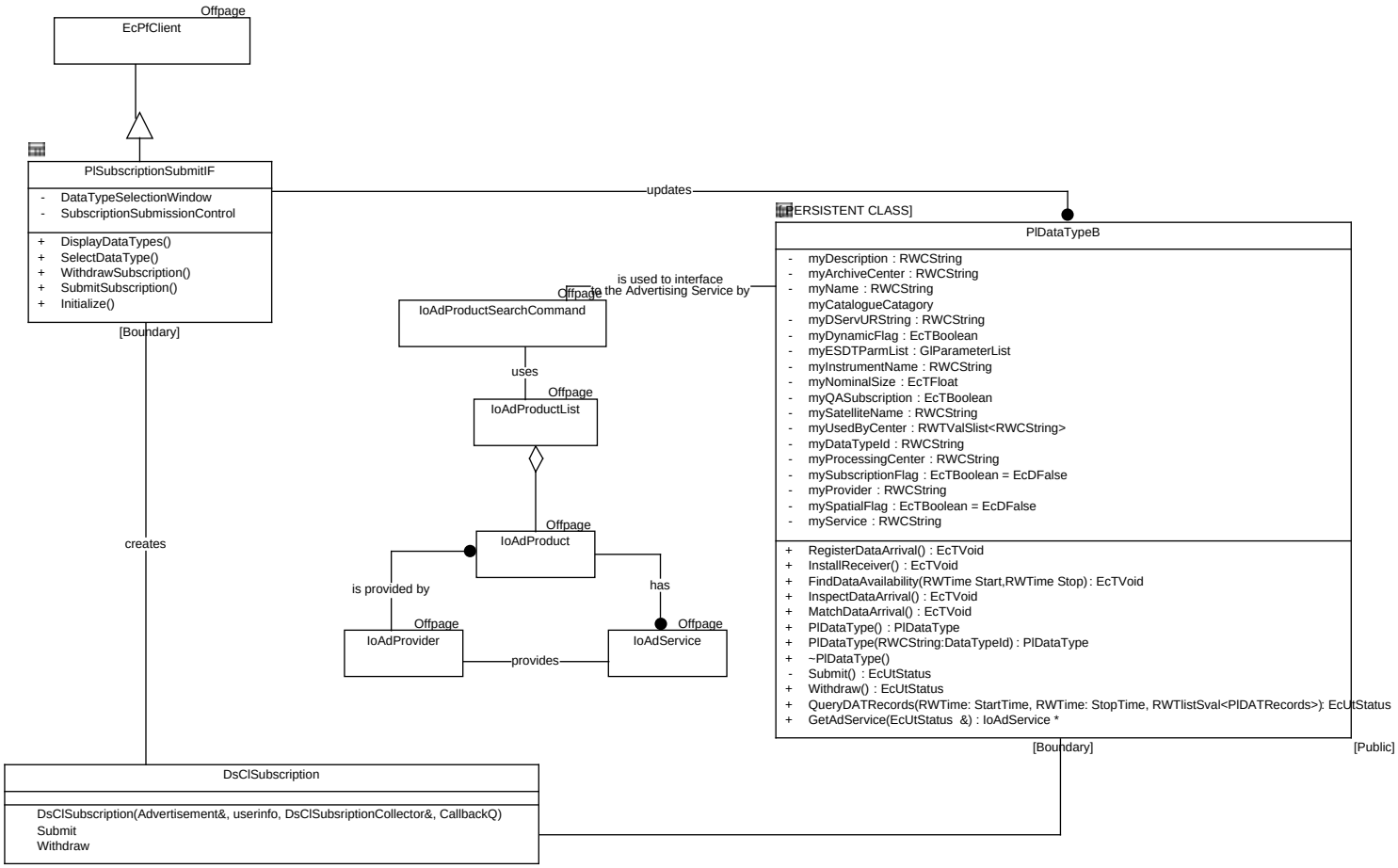



***Figure 4.2-3. Subscription Submission View***

### 4.2.4 Production Planning View

This view (Figure 4.2-4) describes the classes used in the creation and management of plans. The design presented here for a number of key classes is at a high level of abstraction. This level of abstraction is offered to explain the design without going into detail about the planning framework (Delphi) which supports the plan generation. The planning framework within the Production Planning CSCI is documented in detail within the Planning Object Library CSC, Section 4.5.5. That section contains object models, class descriptions and scenarios for the framework and maps the objects used in framework to the abstractions presented here.

The diagram introduces the following key classes:

- PlPlanningWorkbenchUI: This class is an abstraction for the user interface to the planning workbench application. The interface will be developed with a suitable GUI builder tool.
- PlPlan: This class represents an abstraction for a production plan. The class describes the metadata that will be stored for a plan within the PDPS database. The operations shown are an abstraction for those used within the planning framework.
- PlResourceManager: This class represents an abstraction for the resource management capabilities used when generating a plan, describing the operations required to match resource requirements of an activity to the available resources, and to allocate the resource for the activity.
- PlResourceRequirement: This class describes the resource requirements for a task. These requirements may then be matched against the actual resources available.
- PlActivity: This class describes an item within a plan. The activity class is a base class within a specialization hierarchy describing the different activities which occur in the production plan.
- PlPGEActivity: This class is a specialization of the PlActivity class. The class describes a Data Processing Request - a run of a PGE - within the plan.
- PlProdStratNB: This class contains the production strategies used to assign priorities to jobs when producing a production plan.
- PlGroundActivity: This class is a generalization of the PlActivity class. The class describes a Ground Event within the plan.
- PlGroundEvent: This class describes a Ground Event which is recorded in the PDPS database. A Ground Event marks the allocation of resources to some non-production task such as maintenance.
- PlTimelineDisplay: This class describes the user interface component that represents the graphical display of a plan. This will be implemented by a COTS or re-use component.

The interactions of these classes are described within the following scenarios:

•4.4.5 Plan Creation

•4.4.12 Deleting a Plan

•4.4.6 Assigning a Priority to an Activity

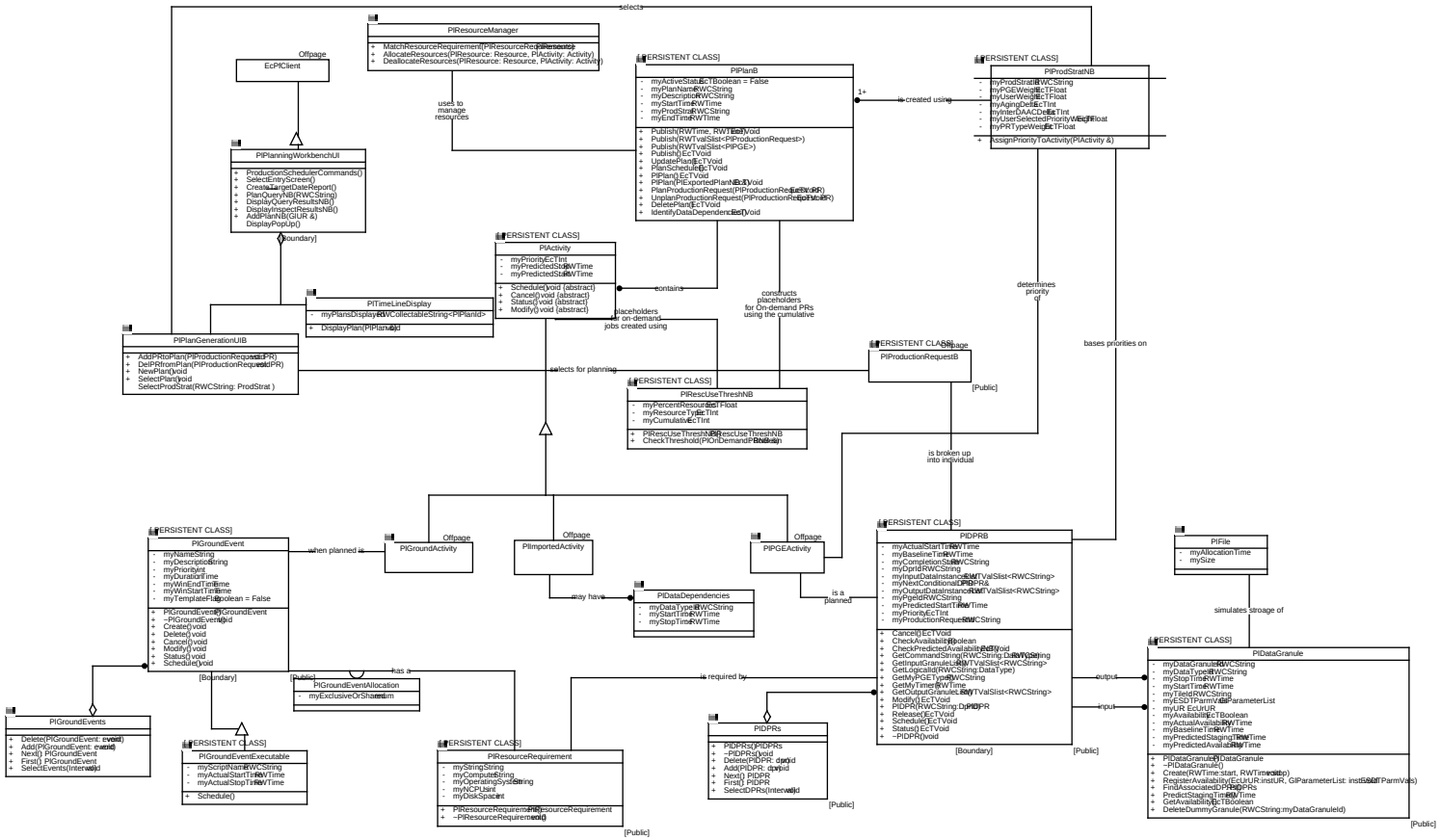



***Figure 4.2-4. Production Planning View***

### 4.2.5 Production Planning User Interface View

This view (Figure 4.2-5) describes the user interface classes associated with the Production Planning Workbench. The view is a continuation from the Production Planning Workbench view, presented separately to reduce the complexity of the Workbench diagram. There are a number of new capabilities in Release B that require operator input including production strategies, On Demand resource usage thresholds and limited automatic replan thresholds.

The diagram introduces the following key classes:

- PlRescUseThreshNB: This class controls the amount of resource that an on-demand production request can use without being deferred.
- PlProdStratNB: This class contains the production strategies used to assign priorities to jobs when producing a production plan.
- PlPGEPriorityNB: This class ties a PGE to a priority; it is used in the production strategies.
- PlUserPriorityNB: This class ties a user type to a priority; it is used in the production strategies.
- PlPRPriorityNB: This class ties a production request type to a priority; it is used in the production strategies. These 'types' can be DAAC defined; in addition to routine, reprocessing and on-demand production request types, an operator can add such types as 'Rush job', 'Customer Complaint' or 'Low priority'.
- PlReplan Criteria: This is an abstract base class for criteria for replan notification.
- PlDASDifferent: This class is used to compare a newly-received DAS or PDAS to the previous one of the same type.
- PlDASDelta: This list of the maximum acceptable delta times for each data type used for determining when to replan.
- PlResourceChange: This class is used to compare an old resource plan with the current resource plan.
- PlOnDemandExceed: This class checks each On-demand Production Request (OPR) against a number of limited automatic replan resource thresholds to determine if the operator should be notified that a replan should be considered.
- PlOnDemandReplanValues: The resource usage thresholds for replan notification - one per resource type (CPU, disk space, etc.).

This view groups together a number of configuration items that are entered in through the Planning Workbench GUIs. Each of the non-GUI classes appear in other object models and their events traces are listed there.

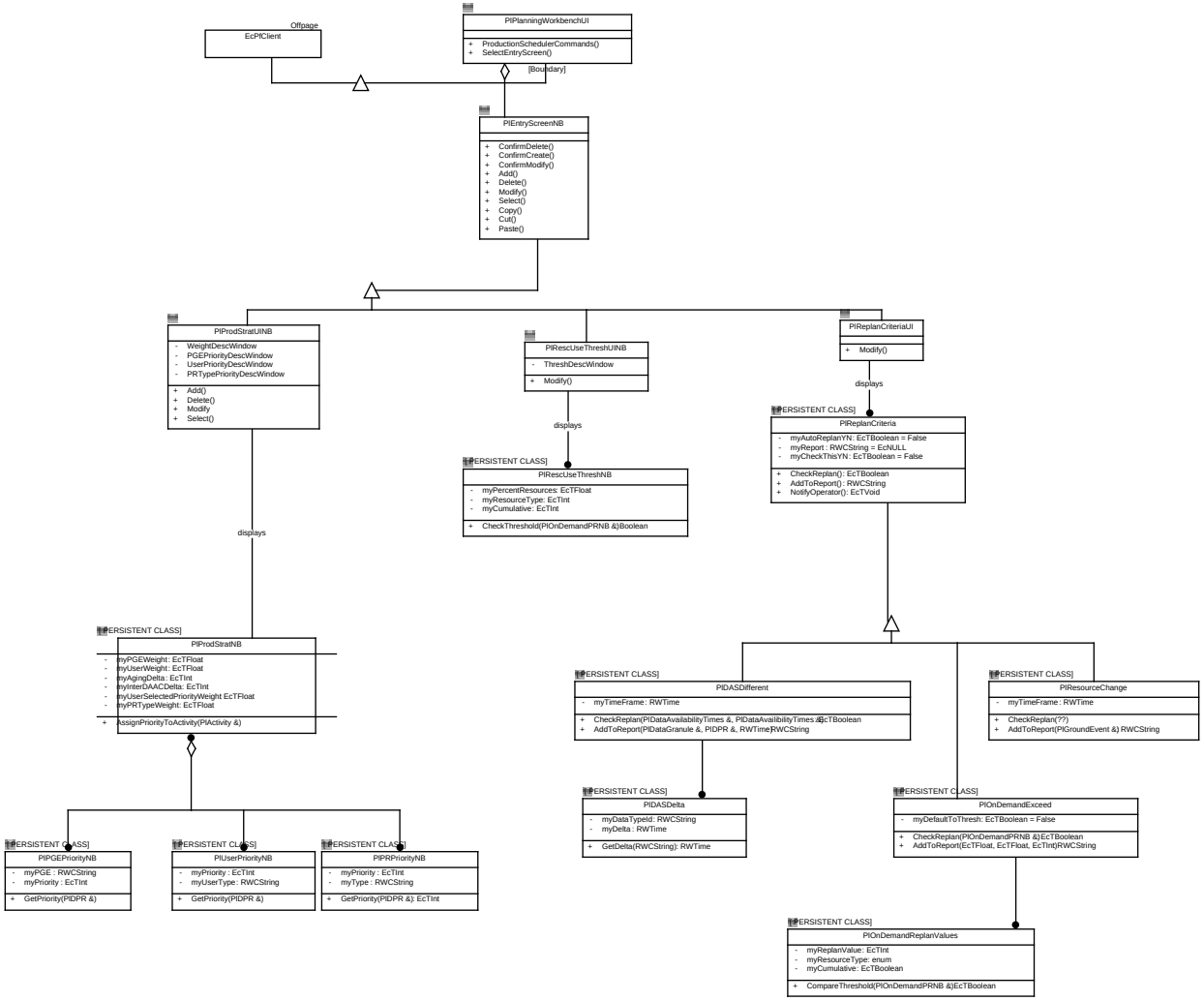



***Figure 4.2-5. Production Planning User Interface View***

### 4.2.6 Publishing Plans View

This view (Figure 4.2-6) describes the classes used in the publishing of plans to the Data Server and the retrieving and displaying of plans from remote DAACs.

The diagram introduces the following key classes:

- PlPlanningWorkbenchUI: This class is an abstraction for the user interface to the planning workbench application. The interface will be developed with a suitable GUI builder tool.
- PlPlan: This class represents an abstraction for a production plan. The class describes the metadata that will be stored for a plan within the PDPS database. The operations shown are an abstraction for those used within the planning framework.
- PlPublishedPlan: This class represents the files published to the Data Server that represent a published plan.
- PlPlanMetadataFile: This is the information stored in the Data Server that can be queried when searching for the published plans.
- PlPlanBinaryReportFile: A representation of a plan designed to be read by office automation tools.
- PlPlanASCIIReportFile: A representation of a plan designed to be human-readable.
- PlExportedPlan: A representation of a plan that is designed to be passed between Planning Subsystems for InterDAAC planning. In addition to have a record for each activity in a plan, it will also have records for any inter-DAAC dependencies within the plan.
- PlActivity: This class describes an item within a plan. The activity class is a base class within a specialization hierarchy describing the different activities which occur in the production plan.
- PlImportedActivity: This class is a specialization of the PlActivity class. The class describes an activity from a plan imported from a remote DAAC.
- PlDataDependencies: This class describes any inter-DAAC dependencies within an imported Plan.
- PlTimelineDisplay: This class describes the user interface component that represents the graphical display of a plan. This will be implemented by a re-use component.

The interactions of these classes are described within the following scenarios:

- 4.4.8 Importing a Plan from the Data Server
- 4.4.7 Publishing a Plan

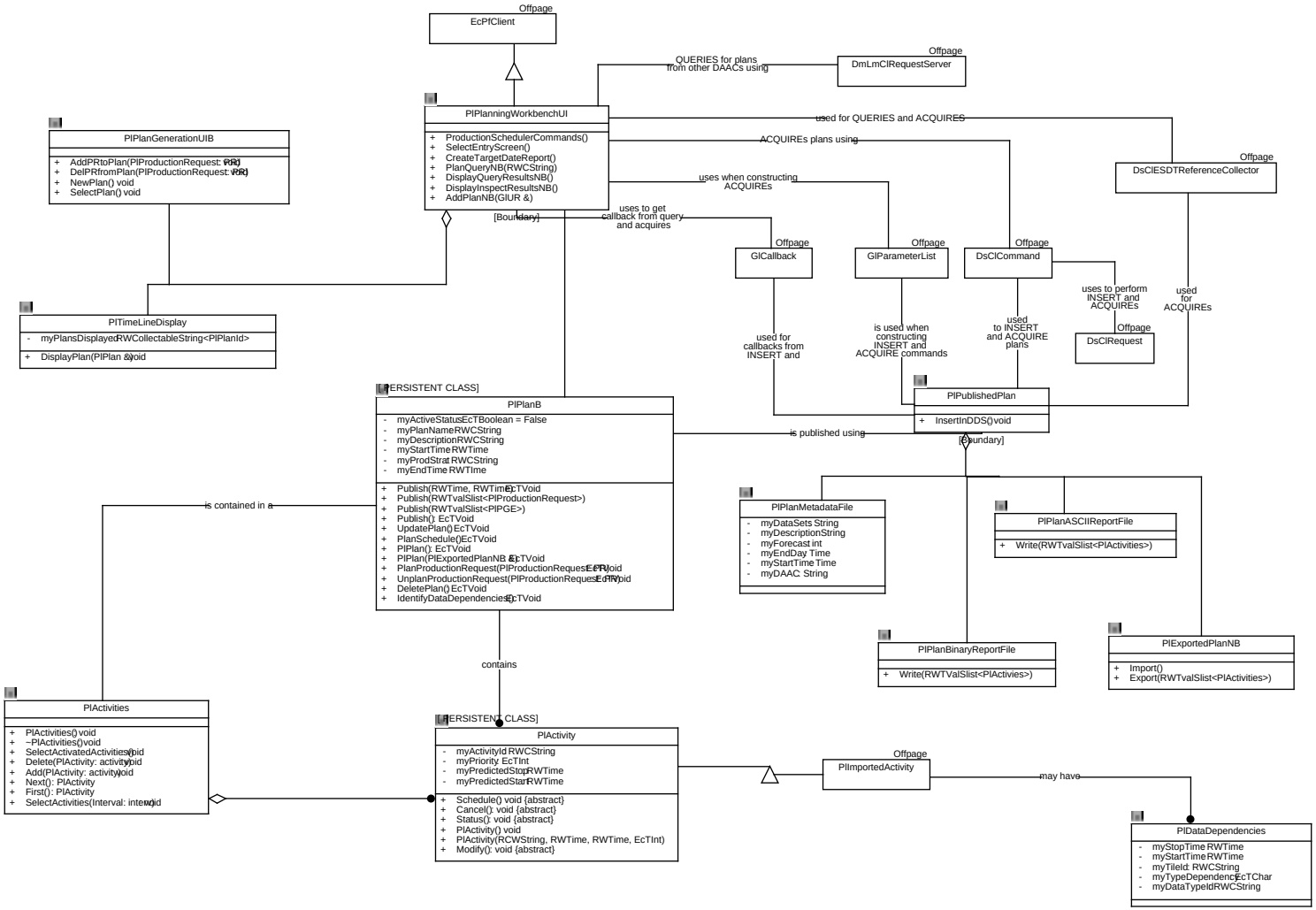



***Figure 4.2-6. Publishing Plans View***

### 4.2.7 Resource Planning View

This view (Figure 4.2-7) describes the classes used during the submission of a resource reservation and the subsequent allocation and planning that takes place. The view is a continuation from the Production Planning Workbench view, presented separately to reduce the complexity of the Workbench diagram.

The following key classes are introduced:

- PlRpResourcePlanningUI: This class provides the methods to support the user interface for defining, reviewing, committing resource reservations and defining and reviewing the resource configuration.
- PlRpResourceTimeline: This class provides the methods to graphically display the approved resource reservations and conflicts.
- PlRpResourcePlan: This collection of approved resource reservations defines the resource plan.
- PlRpPublishedPlan: This class provides the same interface to the Data Server as does the production plan. There are two forms: ASCII and binary.
- PlRpResourceReservationPool: This class manages the pool of resource reservation objects.
- PlRpResourcePool: This class manages the configurable pool of hardware resource objects.
- PlRpResourceReservation: This base class describes a resource reservation and the methods for validating, committing, approving, detecting conflicts, and transitioning between the various states a reservation must go through.
- PlRpResource: This base class describes a resource and provides methods for allocating the resource.
- PlRpResourceState: This class captures the allocation of a resource to a reservation.
- PlRpActivityType: This class is made up of a set of activity types (production, ground events, etc.). The types may include known activities such as production so that a reservation may be made for production.
- PlRpExecutable: This class defines an executable that may be run either before or after the reservation activity (e.g. a script to do something prior to a software upgrade).
- PlRpComplexResourceReservation: This class describes a repetitive resource reservation and generates many reservations when exploded.
- PlRpService: This class describes a logical resource which may be required by a PGE in production planning (e.g. a Data Server)
- PlRpPlatform: This class defines a computer and is derived from PlRpResource. It is made up of CPUs and disks.
- PlRpDiskPartition: This class defines a disk and is derived from PlRpResource.
- PlRpSubNetwork: This class defines a subnetwork component and is derived from PlRpResource.

- PlRpString: This class defines a collection of platforms that are allocated to a particular instrument team.
- PlRpCPU: This class defines a CPU and is derived from PlRpResource.
- PlRpDevice: This class defines a storage device such as a tape drive and is derived from PlRpResource.

The interactions of these classes are described in the following scenario

- 4.4.22 Entering and Approving a Resource Reservation Scneario
- 4.4.23 Creating a Resource Plan
- 4.4.24 Committing a Resource Plan

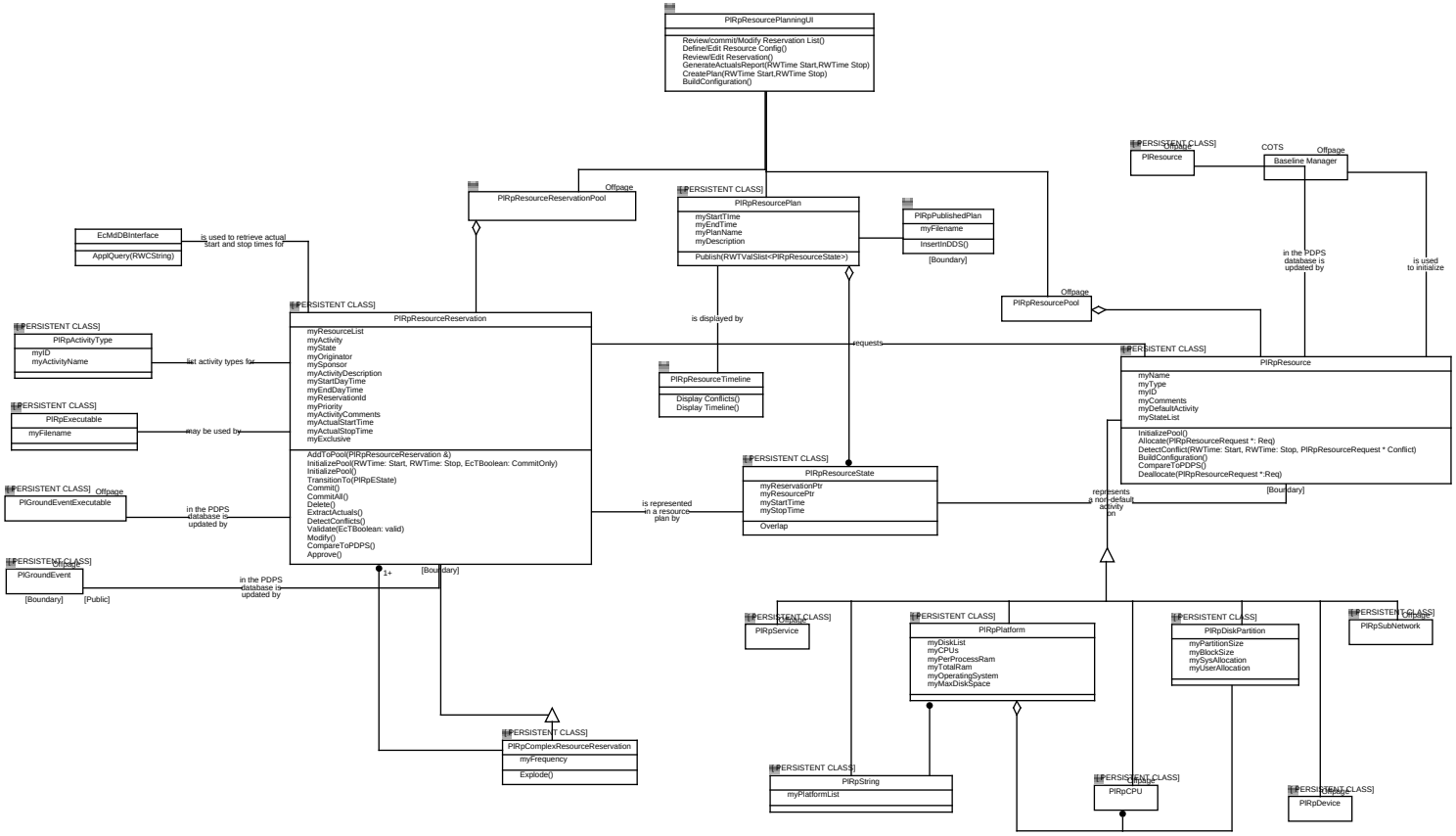



***Figure 4.2-7. Resource Planning View***

### 4.2.8 Resource Management View

This view (Figure 4.2-8) describes the classes used in the description of the production resource configuration. Resource Planning provides the planning subsystem with resource configuration information as described in the Committing a Resource Plan Scenario, Section 4.4.24. This information is used when producing plans, as presented in the Production Planning Workbench view. The details for how the resource configuration is integrated into the framework for the production planning workbench are presented in Section 4.5.4.

The diagram introduces the following key classes:

- PlResource: This class is the base class in a generalization hierarchy describing the production resources.
- PlString: A string describes the logical collection of a number of resources allocated for an instruments processing needs.
- PlComputer: This class describes the production computers
- PlServices: This class describes generic services provided by other subsystems within ECS, such as Data Server and Ingest services
- PlNetwork: This class describes any communication networks or subnetworks
- PlDiskPartition: This class describes the disk resources for data production.

The interactions of these classes are described in the following scenario

- 4.4.24 Committing a Resource Plan

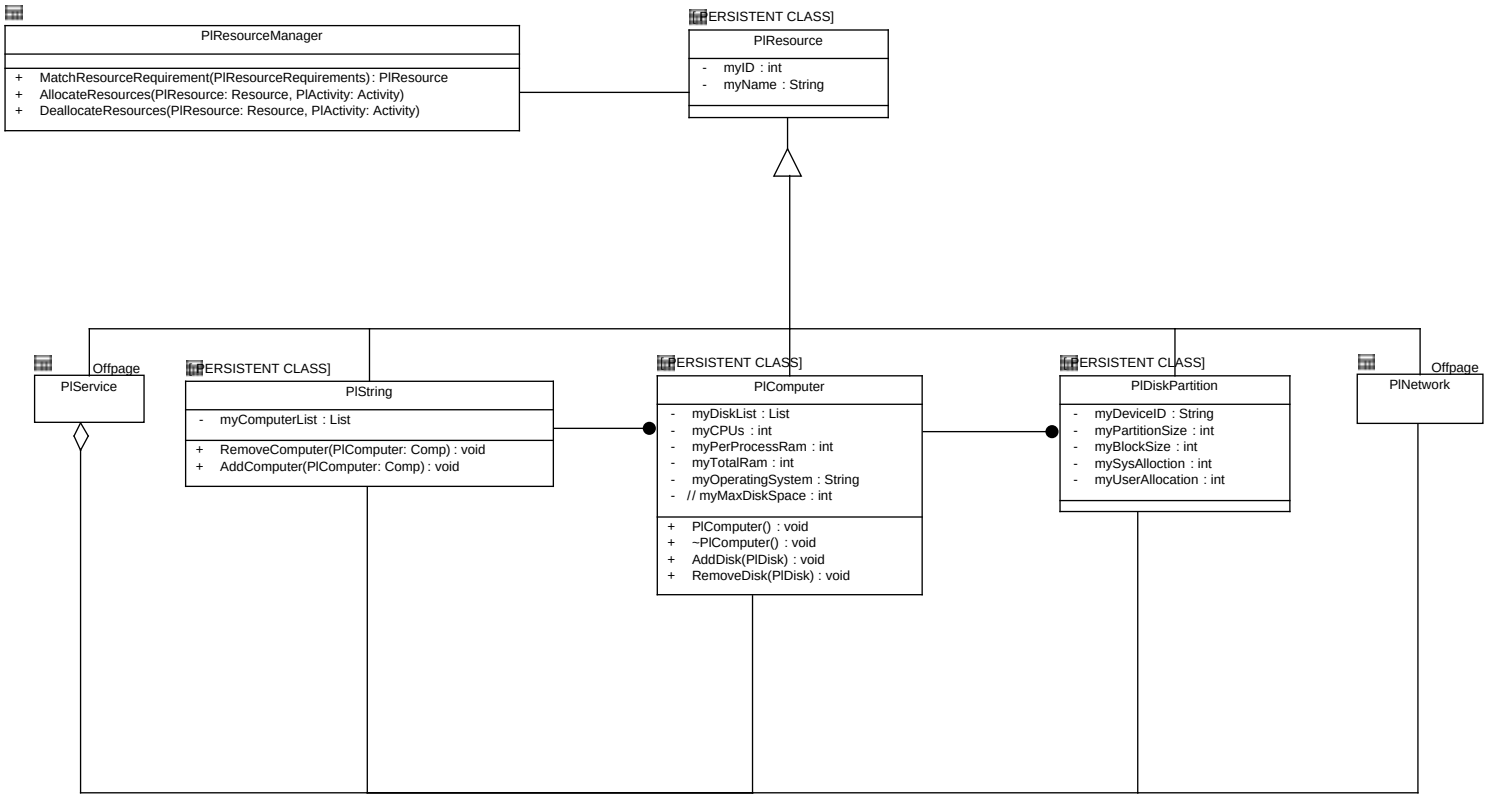



***Figure 4.2-8. Resource Management View***

### 4.2.9 Plan Activation View

This view (4.2-9) again shows more detail for the Planning Workbench. It explicitly shows the Data Processing subsystem interface.

The diagram introduces the following key classes:

- PlActivePlan: This class is the specialization of the PlPlan class and contains the methods to manage the activation, canceling, and statesman of a plan.
- PlPlanB: This class represents an abstraction for a production plan. The class describes the metadata that will be stored for a plan within the PDPS database. The operations shown are an abstraction for those used within the planning framework.
- PlActivity: This class describes an item within a plan. The activity class is a base class within a specialization hierarchy describing the different activities which occur in the production plan.
- PlPDASRecords: The individual records that describe when a data granule that is being produced at a DAAC is predicted to be completed.
- PlPDASMetaData: Describes the Planning Data Availability Schedule (PDAS) records
- PlDPRB: This class describes an individual run of a PGE.
- PlGroundEvent:This class describes a Ground Event which is recorded in the PDPS database. A Ground Event marks the allocation of resources to some non-production task such as maintenance.

The interactions of these classes are described in the following scenario:

- 4.4.9 Plan Activation
- 4.4.11 Canceling a Plan
- 4.4.10 Statusing a Plan

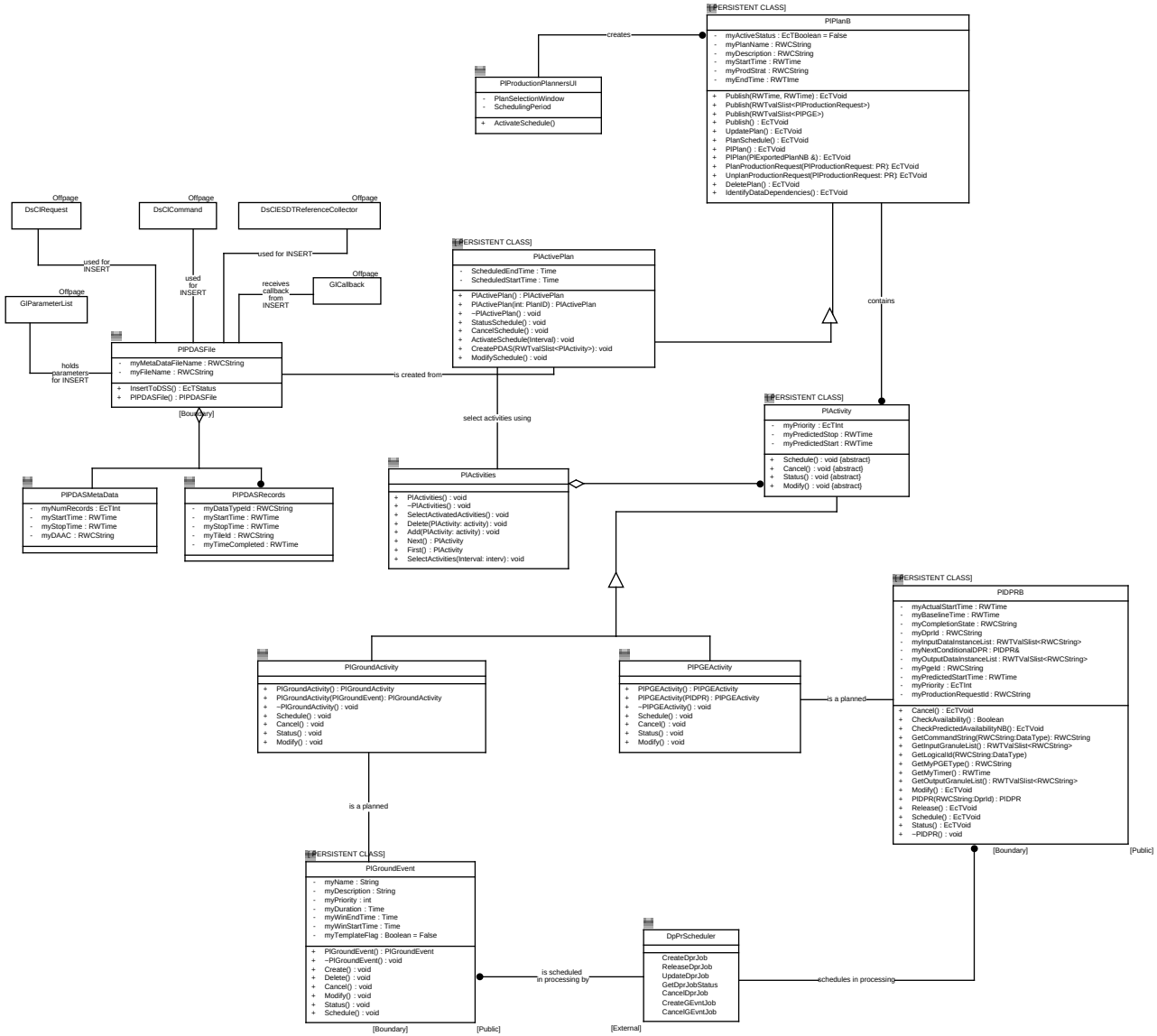



***Figure 4.2-9. Plan Activation View***

### 4.2.10 Subscription Manager View

This view (4.2-10) describes the classes used in the management of notification that a subscription has been fulfilled. Note: in the implementation of Production Rules, the Subscription Manager usese a number of object introduced in Section 4.2.1.

This view introduces the following key classes:

- PlSubscriptionManager: This class contains the main application methods associated with the subscription manager application.
- PlSubMsgCb: Specialization class to provide the subscription manager callbacks to the EcMpMsgCb in order to handle the callback chores.
- EcMpMsgCb: This class encapsulates the communication for message passing used for subscription notifications.
- EcPfManagedServer: This class contains the external interfaces required by the Managed process framework in order to activate the appropriate lifecycle services.
- EcUrUR: This class is constructed to provide a Universal Reference (UR) of the Data Server subscription notification.
- DsClESDTReference: This class provides the subscription manager with services by which to access the metadata associated with a Universal Reference.
- PlDataAvailabilityTimes: This aggregate class is composed of records with predictions of data availability from other sources. It is created from either an FOS Detailed Activity Schedule or from a Planning Data Availability Schedule received from another DAAC.
- PlDASDifferent: This class compares newly received FOS Detailed Activity Schedules and Planning Data Availability Schedules to the previous one of the same type.
- PlDASDelta: This class defines the deltas for each data type.

The interactions of these classes are described within the following scenarios:

- 4.4.13Subscription Notification
- 4.4.14Subscription Notification with Spatial Based Input
- 4.4.15Subscription Notification with Alternate Inputs
- 4.4.17Limited Automatic Replan based on a New DAS

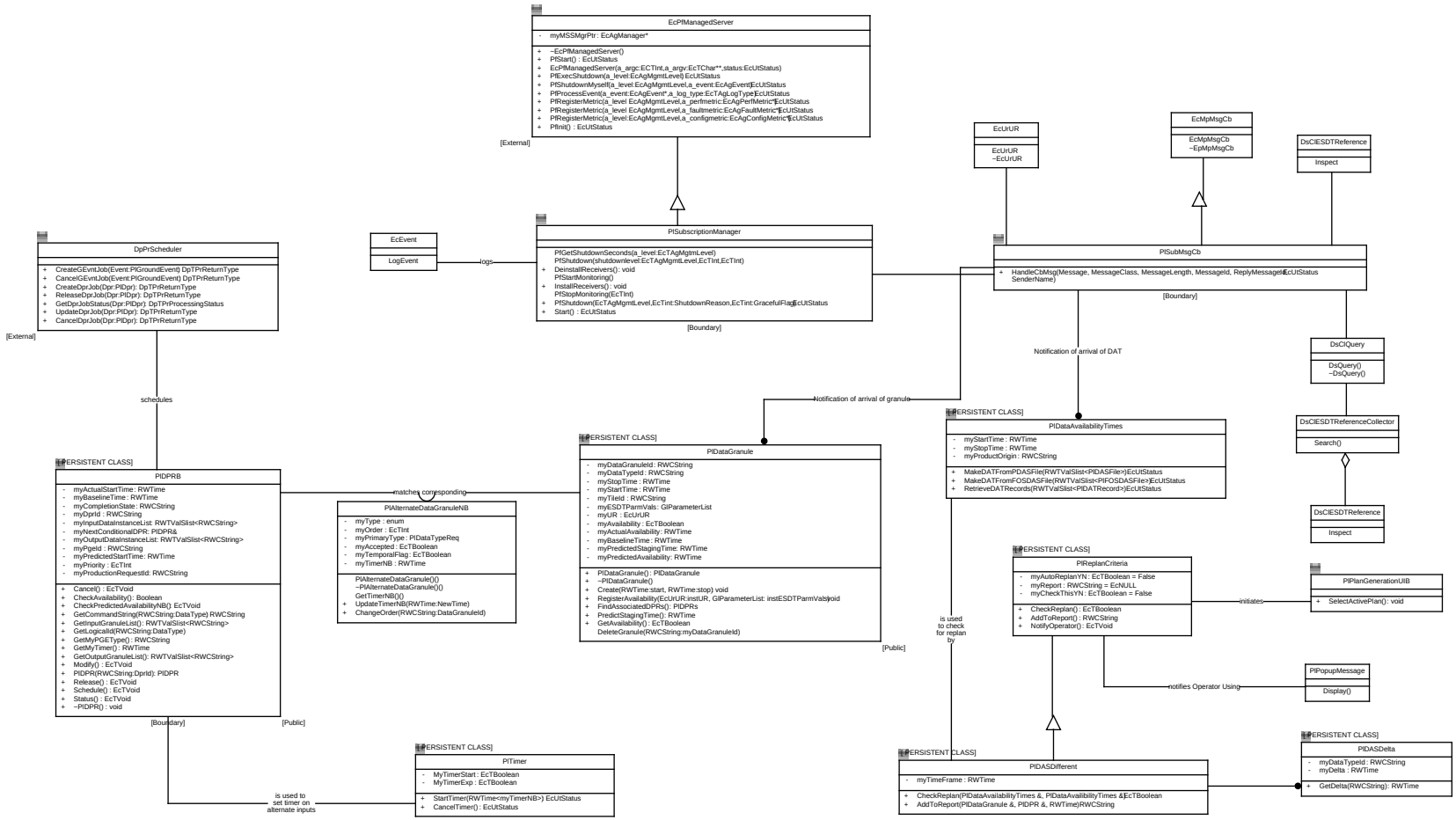



***Figure 4.2-10. Subscription Manager View***

### 4.2.11 Data Activity Times View

This view (4.2.11) shows the structure and methods of a DataActivityTimes object and the precursor objects from which it constructs itself.

The key classes introduced within this model are:

- PlDATRecord: This class defines the component records of a PlDataAvailabilityTimes schedule. It contains the predicted time range and associated information for the arrival of each data granule.
- PlDataSchedules: This class is used for retrieving either a FOS DAS or a PDAS from the data server.
- PlPDASFile: This class represents the file containing the Planning Data Availability Schedule.
- PlPDASMetaData: This class contains the metadata information associated with the PDAS to be retrieved from the data server.
- PlPDASRecords: This class contains records associated with the PDAS to be retrieved from the data server.
- PlFOSDASFile: This class represents the file that contains the FOS Detailed Activity Record.
- PlDASNB: This class contains the metadata information associated with the FOS DAS to be retrieved from the data server.
- PlPEDASRecordNB: This class contains records associated with the FOS DAS to be retrieved from the data server.
- PlPEDASModeRecordNB: This class contains the instrument mode records associated with the FOS DAS to be retrieved from the data server.
- PlInstrumentModes: This aggregate class is composed of PlInstModeRecords. It is created when a FOS Detailed Activity Schedule is received.
- PlInstModeRecords: This class defines the component records of a PlInstrumentModes Schedule. It contains the predicted time range associated with the specified mode for a particular instrument.
- PlDataTransferHistory: This class defines the methods for calculating the empirically-derived estimate of the time it takes for a granule to be processed by either EDOS or another DAAC until the time it is available on the data server.
- PlSourceToDsHistoryNB: This class contains historical data which is used to compute a moving average of the delay times of each dataType subscribed to.

The interactions of these classes are described within the following scenario:

- 4.4.16 Data Availability Times

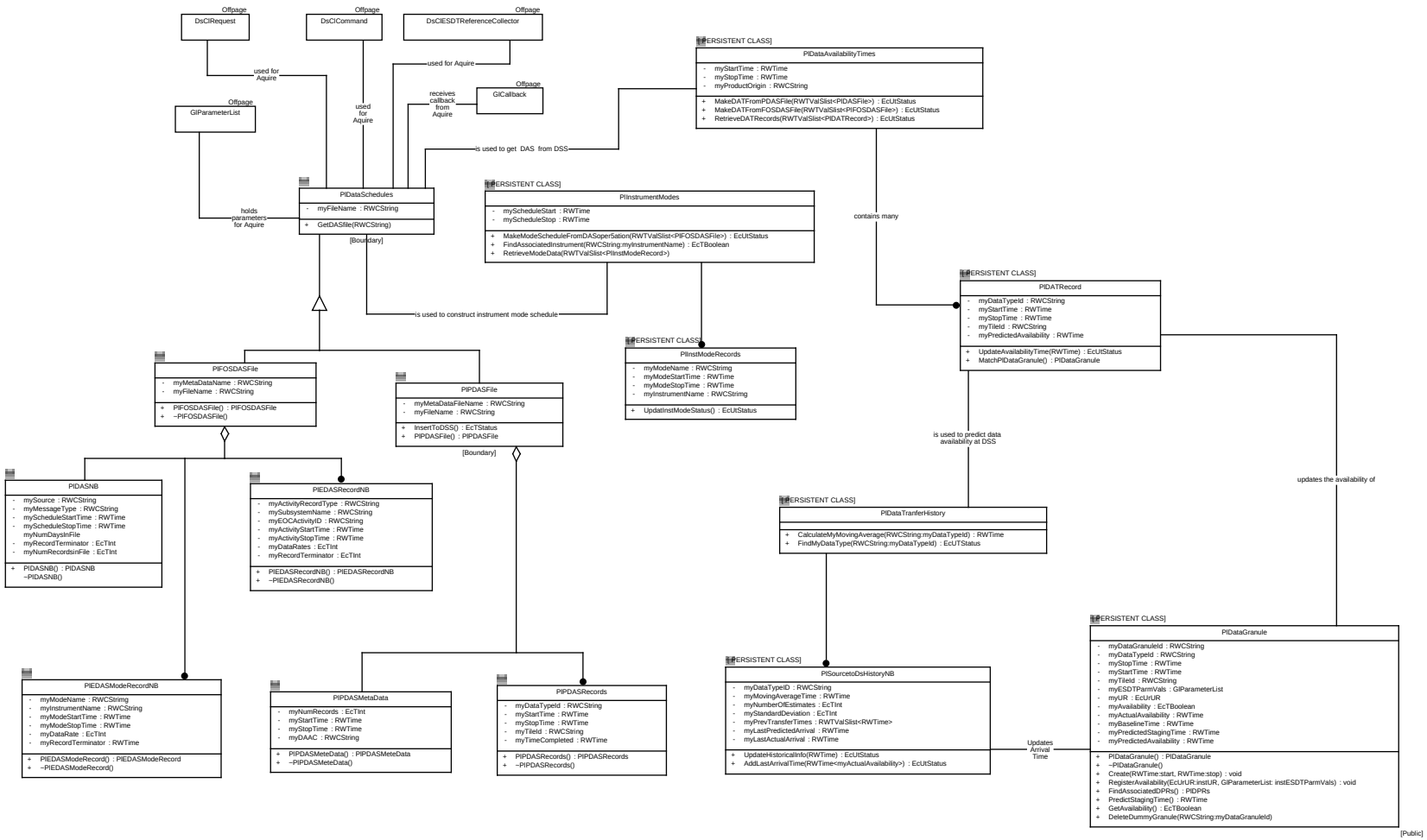



***Figure 4.2-11. Data Activity Times View***

### 4.2.12 DBMS Proxy Agent View

This view (Figure 4.2-12) describes the classes used in the management of the lifecycle services for the PDPS DBMS. This application is almost a template reuse of the MSS services for Proxy Agents. For further details of the MSS agents and proxy agents refer to Release A CSMS Management Subsystem Design Specification for the ECS Project.

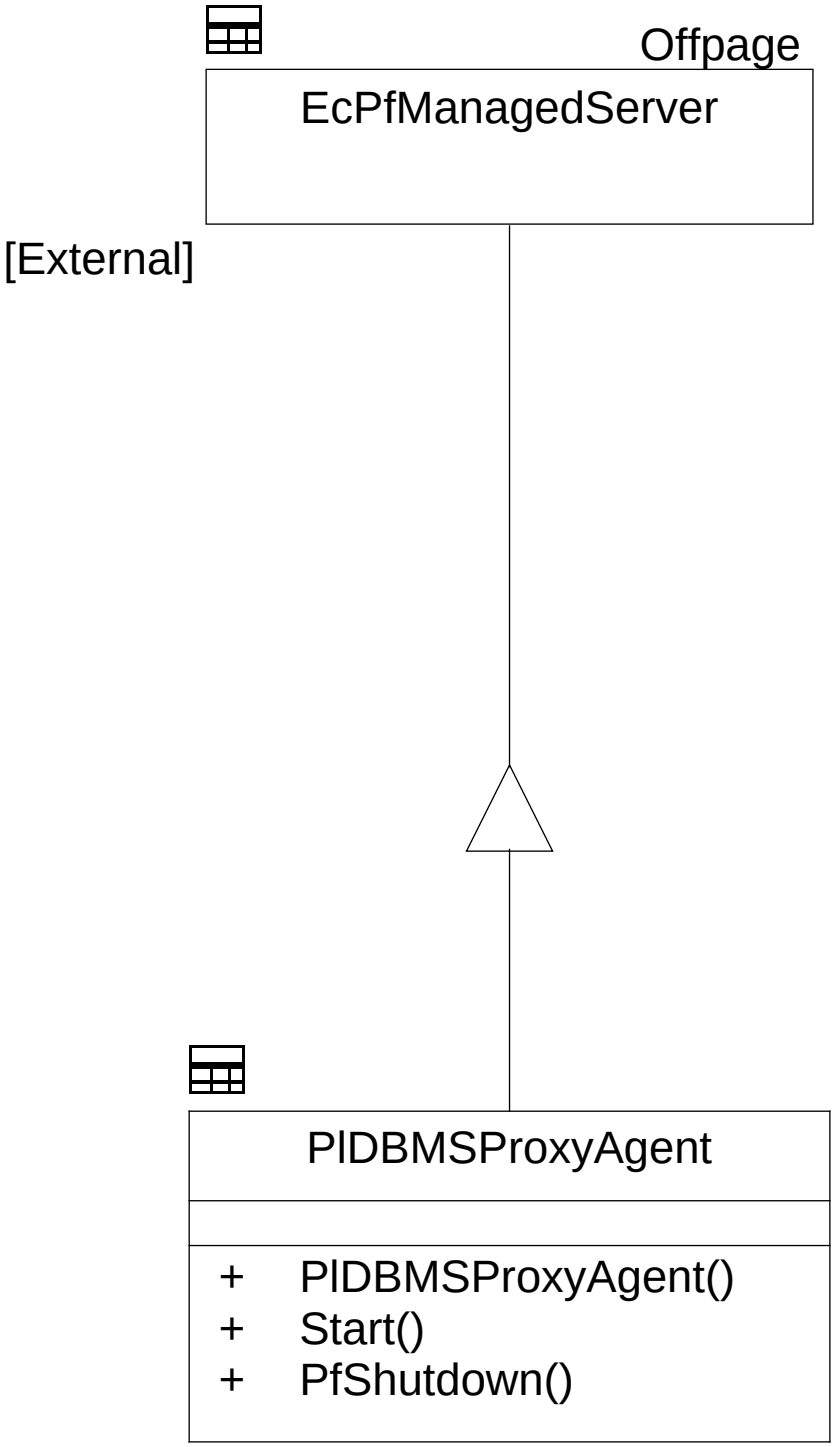



***Figure 4.2-12. DBMS Proxy Agent View***

### 4.2.13 On-Demand Manager View

The On-Demand Manager view (Figure 4.2-13) describes the classes which are used by the planning subsystem to check an On-Demand Production Request and ensure that it is eligible for submission to the processing subsystem.

The key classes introduced within this model are:

- PlOnDemandManagerNB: This class is the manager for all On-Demand Product Requests providing the capability to make modifications, provide status and cancel production requests as necessary.
- PlOnDemandPRNB: This class is the specialization class which holds additional/modified attributes and the operations needed for On-Demand Production Requests beyond those found in the PlProductionRequestB and it's subclasses.
- PlRescUseThreshNB: This class is a table containing the resource usage thresholds for on-demand production requests and the operation which compares the resource requirements.
- EcRequest: This is the global class used to maintain the current status of a request to make that status available to any user needing to know the current status any given time.
- EcEvent: This object provides the event logging interface to the management system subsystem.
- EcPfManagedServer: This class is the framework class for Managed Server Processes. This class also provides the interface to the management system subsystem lifecycle services.
- EcMpMsgCb: This class provides the callback notification for communications between servers.
- PlSubscriptionSubmitIF: This class is an abstraction for the user interface to the subscription submission application. The interface will be developed with a suitable GUI building tool.
- PlReplanCriteria: This class is an abstract base class for the criteria for replan notification.
- PlOnDemandExceed: This class checks each On-Demand Production Request against a number of limited automatic replan resource thresholds to determine if the operator should be notified that a replan should be considered.
- PlOnDemandReplanValues: This class maintains the resource usage thresholds used for determining if a replan notification is necessary - one per resource type .
- PlPlanGenerationUIB: This class is an abstraction for the user interface to the planning workbench application.

The interactions of these classes are described within the following scenarios:

- 4.4.19 On-Demand Product Request Scenario (Processed)
- 4.4.21 On-Demand Product Request Scenario (Delayed)
- Limited Automatic Replan based on an On-Demand PR

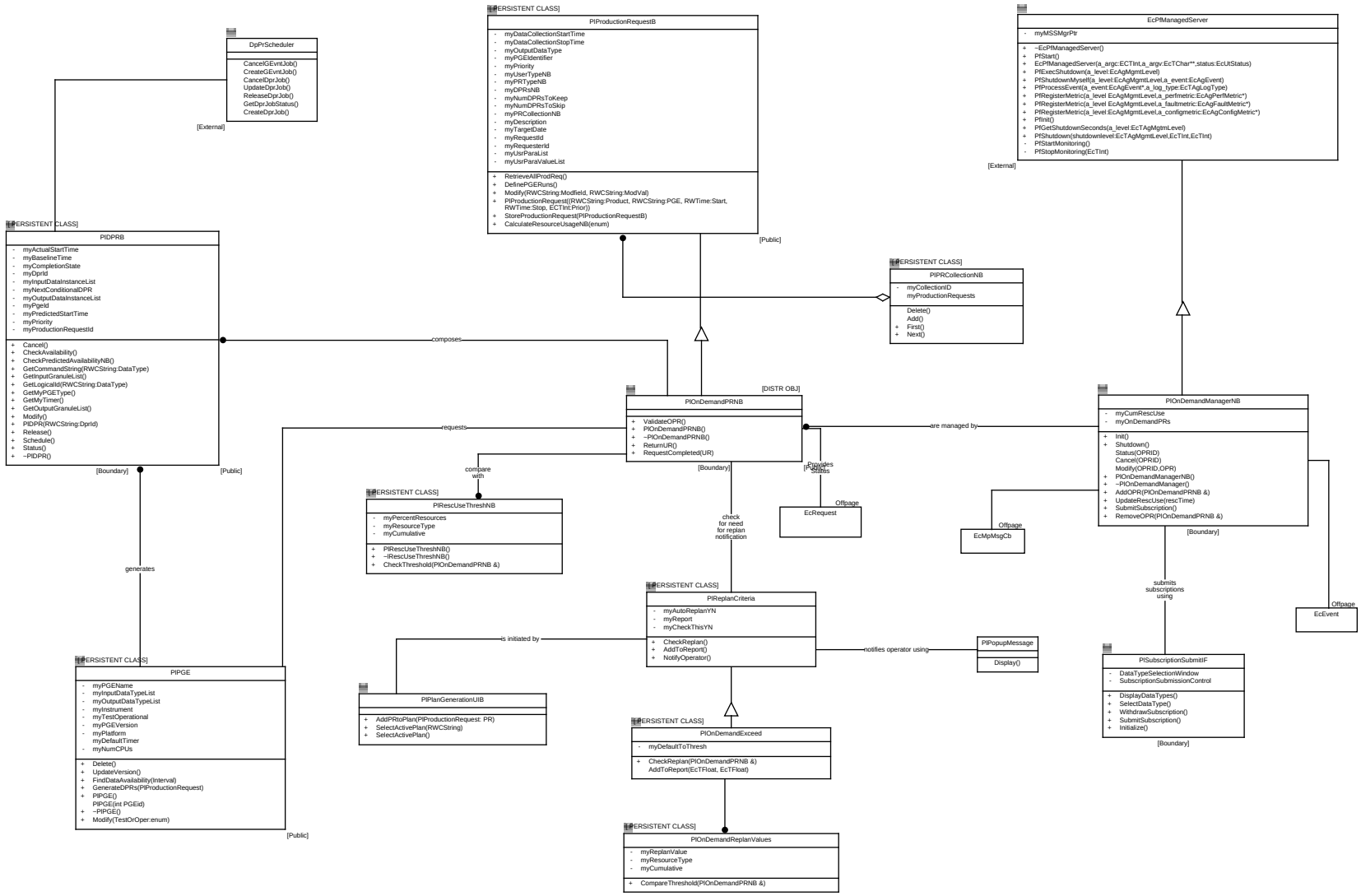



***Figure 4.2-13. On-Demand Manager View***

## 4.3 Production Planning Class Descriptions

### 4.3.1 DmLmClRequestServer Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The DmLmClRequestServer class has associations with the following classes:
Class: PlPlanningWorkbenchUI QUERIESforplansfromotherDAACsusing

### 4.3.2 DpPrScheduler Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
DpPrScheduler provides operations to manage science software on a DPR level.

**Attributes:**

None

**Operations:**

**CancelDprJob**
Arguments:

**CancelDprJob**
Arguments:Dpr:PlDpr
Return Type:DpTPrReturnType
Privilege:Public

**CancelDprJob**
Arguments:

**CancelGEvntJob**
Arguments:Event:PlGroundEvent
Return Type:DpTPrReturnType
Privilege:Public

**CancelGEvntJob**
Arguments:

**CreateDprJob**
Arguments:

**CreateDprJob**
Arguments:Dpr:PlDpr
Return Type:DpTPrReturnType
Privilege:Public

**CreateDprJob**
Arguments:

**CreateGEvntJob**
Arguments:Event:PlGroundEvent
Return Type:DpTPrReturnType
Privilege:Public

**CreateGEvntJob**
Arguments:

**GetDprJobStatus**
Arguments:

**GetDprJobStatus**
Arguments:Dpr:PlDpr
Return Type:DpTPrProcessingStatus
Privilege:Public

**GetDprJobStatus**
Arguments:

**ReleaseDprJob**
Arguments:Dpr:PlDpr
Return Type:DpTPrReturnType
Privilege:Public

**ReleaseDprJob**

Arguments:

**UpdateDprJob**
Arguments:

**UpdateDprJob**
Arguments:Dpr:PlDpr
Return Type:DpTPrReturnType
Privilege:Public

**UpdateDprJob**
Arguments:

**Associations:**

The DpPrScheduler class has associations with the following classes:
Class: PlDPRB
Class: PlGroundEvent isscheduledinprocessingby
Class: PlDPRB schedules
Class: PlDPRB schedulesinprocessing

### 4.3.3 DsClCommand Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The DsClCommand class has associations with the following classes:
Class: PlPlanningWorkbenchUI ACQUIREsplansusing
Class: PlDataSchedules usedforAquire
Class: PlPDASFile usedforINSERT
Class: PlPublishedPlan usedtoINSERTandACQUIREplans
Class: DsClRequest usestoperformINSERTandACQUIREs

### 4.3.4 DsClESDTReference Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

**Inspect**
Arguments:

**Associations:**

The DsClESDTReference class has associations with the following classes:
Class: PlSubMsgCb
DsClESDTReferenceCollector (Aggregation)

### 4.3.5 DsClESDTReferenceCollector Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

**Search**
Arguments:

**Associations:**

The DsClESDTReferenceCollector class has associations with the following classes:
Class: DsClQuery
Class: PlPublishedPlan usedforACQUIREs
Class: PlDataSchedules usedforAquire
Class: PlPDASFile usedforINSERT
Class: PlPlanningWorkbenchUI usedforQUERIESandACQUIRES

### 4.3.6 DsClQuery Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

**DsQuery**
Arguments:

**~DsQuery**
Arguments:

**Associations:**

The DsClQuery class has associations with the following classes:
Class: DsClESDTReferenceCollector
Class: PlDataTypeB
Class: PlSubMsgCb

### 4.3.7 DsClRequest Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The DsClRequest class has associations with the following classes:
Class: PlDataSchedules usedforAquire
Class: PlPDASFile usedforINSERT
Class: DsClCommand usestoperformINSERTandACQUIREs

### 4.3.8 DsClSubscription Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

**DsClSubscription**
Arguments:Advertisement&, userinfo, DsClSubsriptionCollector&, CallbackQ

**Submit**
Arguments:

**Withdraw**
Arguments:

**Associations:**

The DsClSubscription class has associations with the following classes:
Class: PlDataTypeB
Class: PlSubscriptionSubmitIF creates

### 4.3.9 EcEvent Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
Event logging interface object

**Attributes:**

None

**Operations:**

**LogEvent**
Arguments:

**Associations:**

The EcEvent class has associations with the following classes:
Class: PlOnDemandManagerNB
Class: PlSubscriptionManager logs

### 4.3.10 EcMpMsgCb Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
Message Callback object

**Attributes:**

None

**Operations:**

**EcMpMsgCb**
Arguments:

**~EpMpMsgCb**
Arguments:

**Associations:**

The EcMpMsgCb class has associations with the following classes:
Class: PlOnDemandManagerNB

### 4.3.11 EcPfClient Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:

**Attributes:**

None

**Operations:**

None

**Associations:**

The EcPfClient class has associations with the following classes:
None

### 4.3.12 EcPfManagedServer Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class is the framework class for Managed Server Processes. This class defines the method Process Event which handles the events generated by the Managed Server Processes. This class is also connected to the MSS EcAgManager class as required by the MSS desing. The Managed Server class will povide methods to inform the EcAgManager to start and stop monitoring, to inform the EcAgManager the number of shutdown seconds required for the application, program, and process, to register metrics with the EcAgManager. The Managed Server class also receives requests from the EcAgManager class to suspend, resume, and shutdown. The method, PfShutdownMyself, is provided by the Managed Server class to the application may request a shutdown of itself.

**Attributes:**

**myMSSMgrPtr** - Pointer to the EcAgManger object. Used to call methods contained in the EcAgManager class.
Data Type:EcAgManager*
Privilege:Private
Default Value:

**Operations:**

**EcPfManagedServer** - Constructor
Arguments:a_argc:ECTInt,a_argv:EcTChar**,status:EcUtStatus
Return Type:Void
Privilege:Public

**PfExecShutdown** - Calls theServer Shutdown() and PfShutdown method with the management level(application, program, process).
Arguments:a_level:EcAgMgmtLevel
Return Type:EcUtStatus
Privilege:Public

**PfGetShutdownSeconds** - Needs to be overriden by application, the number of seconds requires to shutdown the application
Arguments:a_level:EcTAgMgtmLevel
Return Type:EcTInt
Privilege:Public

**PfInit** - Calls the PfGenServerInit(inherited) method, instantiates the EcAgManager Object and registers it with the GSO
Arguments:
Return Type:EcUtStatus
Privilege:Public

**PfProcessEvent** - Calls the ProcessEvent method of the EcAgManager object
Arguments:a_event:EcAgEvent*,a_log_type:EcTAgLogType
Return Type:EcUtStatus
Privilege:Public

**PfRegisterMetric** - Calls the RegisterMetric method of the EcAgManager object.
Arguments:a_level EcAgMgmtLevel,a_faultmetric:EcAgFaultMetric*
Return Type:EcUtStatus
Privilege:Public

**PfRegisterMetric** - Calls the RegisterMetric method of the EcAgManager object.
Arguments:a_level:EcAgMgmtLevel,a_configmetric:EcAgConfigMetric*
Return Type:EcUtStatus
Privilege:Public

**PfRegisterMetric** - Calls the RegisterMetric method of the EcAgManger object.
Arguments:a_level EcAgMgmtLevel,a_perfmetric:EcAgPerfMetric*
Return Type:EcUtStatus
Privilege:Public

**PfRegisterMetric** - Calls the RegisterMetric method of the EcAgManager object.
Arguments:a_level EcAgMgmtLevel,a_faultmetric:EcAgFaultMetric*

Return Type:EcUtStatus
Privilege:Public

**PfRegisterMetric** - Calls the RegisterMetric method of the EcAgManager object.
Arguments:a_level:EcAgMgmtLevel,a_configmetric:EcAgConfigMetric*
Return Type:EcUtStatus
Privilege:Public

**PfShutdown** - Needs to be overriden by application to perform specific shutdown appropriate for application
Arguments:shutdownlevel:EcTAgMgmtLevel,EcTInt,EcTInt
Return Type:EcUtStatus
Privilege:Public

**PfShutdownMyself** - Calls StopMonitoring. Calls PfShutdown method with the management level(application,program,process). Calls the DCEServer Shutdown method.
Arguments:a_level:EcAgMgmtLevel,a_event:EcAgEvent
Return Type:EcUtStatus
Privilege:Public

**PfStart** - Calls PfStartMonitoring and DCEServer listen methods
Arguments:
Return Type:EcUtStatus
Privilege:Public

**PfStartMonitoring** - Calls StartMonitoring() method of the EcAgManager Object
Arguments:
Return Type:EcUtStatus
Privilege:Private

**PfStopMonitoring** - Calls StopMonitoring() method of the EcAgManager Object
Arguments:EcTInt
Return Type:EcUtStatus
Privilege:Private

**~EcPfManagedServer** - Destructor. Deletes the EcAgManager object.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The EcPfManagedServer class has associations with the following classes:
None

### 4.3.13 EcRequest Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
Global status passing object for anyone knowledgeable of this request to check the status of the request.

**Attributes:**

None

**Operations:**

None

**Associations:**

The EcRequest class has associations with the following classes:
Class: PlOnDemandPRNB ProvidesStatus

### 4.3.14 EcUrUR Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

**EcUrUR**
Arguments:

**~EcUrUR**
Arguments:

**Associations:**

The EcUrUR class has associations with the following classes:
Class: PlSubMsgCb

### 4.3.15 GlCallback Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The GlCallback class has associations with the following classes:
Class: PlDataSchedules receivescallbackfromAquire
Class: PlPDASFile receivescallbackfromINSERT
Class: PlPublishedPlan usedforcallbacksfromINSERTand
Class: PlPlanningWorkbenchUI usestogetcallbackfromqueryandacquires

### 4.3.16 GlParameterList Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The GlParameterList class has associations with the following classes:
Class: PlDataSchedules holdsparametersforAquire
Class: PlPDASFile holdsparametersforINSERT
Class: PlPublishedPlan isusedwhenconstructingINSERTandACQUIREcommands
Class: PlPlanningWorkbenchUI useswhenconstructingACQUIREs

### 4.3.17 IoAdProduct Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The IoAdProduct class has associations with the following classes:
Class: IoAdService has
Class: IoAdProvider isprovidedby
IoAdProductList (Aggregation)

### 4.3.18 IoAdProductList Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The IoAdProductList class has associations with the following classes:
Class: IoAdProductSearchCommand uses

### 4.3.19 IoAdProductSearchCommand Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The IoAdProductSearchCommand class has associations with the following classes:
Class: PlDataTypeB isusedtointerfacetotheAdvertisingServiceby
Class: IoAdProductList uses

### 4.3.20 IoAdProvider Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The IoAdProvider class has associations with the following classes:
Class: IoAdProduct isprovidedby
Class: IoAdService provides

### 4.3.21 IoAdService Class

Parent Class:Not Applicable

**Attributes:**

None

**Operations:**

None

**Associations:**

The IoAdService class has associations with the following classes:
Class: IoAdProduct has
Class: IoAdProvider provides

### 4.3.22 PlActivePlan Class

Parent Class:PlPlanB
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is the specialization of the PlPlan class and contains the methods to manage the activation, cancelling, and statusing of a plan

**Attributes:**

**ScheduledEndTime -** The end date and time for the portion of the plan entered into the data processing subsystem scheduler.
Data Type:Time
Privilege:Private
Default Value:

**ScheduledStartTime -** The start date and time for the portion of the plan entered into the data processing subsystem scheduler.
Data Type:Time
Privilege:Private
Default Value:

**Operations:**

**ActivateSchedule -** Schedule the activities within the specified interval in the plan with the data processing scheduler.
Arguments:Interval
Return Type:void
Privilege:Public
PDL: {
// Create an ordered list of activities which are within the plan
// and lie within the interval specified
// Iterate through the list, and invoke the schedule method for
// each activity
}

**CancelSchedule -** Cancel the activities that have been scheduled from the plan within the data processing scheduler

Arguments:
Return Type:void
Privilege:Public
PDL: {
// Create an ordered list of activities which are within the plan
// and have been scheduled within the data processing scheduler
// Iterate through the list, and invoke the cancel method for
// each activity
}

**CreatePDAS -** Creates a PDPS Data Availability Schedule for the current active plan and stores this file to the Data Server.
Arguments:RWTvalSlist<PlActivity>
Return Type:void
Privilege:Public
PDL: {
Open file for metadata
  Write out metadata parameters
  Close file
Open file for PDAS records
For each activity in entire plan
  get DPR id
  For each output data in myOutputDataInstanceList whos DataType indicates
    that this data is used at a remote DAAC
Write out a record with the data type,
  start and stop times of the data granule,
  plus the time this data granule should be produced
  (myPredictedStartTime + the time the PGE nominally runs)
Close file
}

**ModifySchedule -** Modify the activities that have been scheduled from the plan within the data processing scheduler. The attributes of the activity that may be modified are priority and the anticipated start / stop times of the activity used to set the alarms in the data processing scheduler. Any greater modification should be managed by canceling the activity and scheduling a new activity.
Arguments:
Return Type:void
Privilege:Public
PDL:{
// Create an ordered list of activities which are within the plan
// and have been scheduled within the data processing scheduler

// Iterate through the list, and invoke the modify method for
// each activity

}

**PlActivePlan** - Creator method for the active plan
Arguments:
Return Type:PlActivePlan
Privilege:Public

**PlActivePlan** - Recreates the active plan from the DBMS given an identifier for that plan.
Arguments:int: PlanID
Return Type:PlActivePlan
Privilege:Public

**StatusSchedule** - Status the plan against the schedule active in the data processing scheduler.
Arguments:
Return Type:void
Privilege:Public
PDL:{
// Create an ordered list of activities which are within the plan
// and have been scheduled within the data processing scheduler

// Iterate through the list, and invoke the status method for
// each activity -- this is reflected within the activity object
// and DBMS

// Update the plan accordingly
}

**~PlActivePlan** - Destructor method for the active plan.
Arguments:
Return Type:void
Privilege:Public

**Associations:**

The PlActivePlan class has associations with the following classes:
Class: PlPDASFile iscreatedfrom
Class: PlActivities selectactivitiesusing

### 4.3.23 PlActivities Class

Parent Class:Not Applicable
Public:No
Distributed Object:No

Purpose and Description:
This class is a container class for activities in the PDPS database. This class may be implemented by a suitable Rogue Wave template class.

**Attributes:**

None

**Operations:**

**Add** - Add an activity to the collection
Arguments:PlActivity: activity
Return Type:void
Privilege:Public

**Delete** - Delete an activity from the collection.
Arguments:PlActivity: activity
Return Type:void
Privilege:Public

**First** - Returns the first activity within the collection.
Arguments:
Return Type:PlActivity
Privilege:Public

**Next** - Returns the next activity within the collection (or NULL if no more activities).
Arguments:
Return Type:PlActivity
Privilege:Public

**PlActivities** - Constructor for the container class.
Arguments:
Return Type:void
Privilege:Public

**SelectActivatedActivities** - Select those activities within the plan that have been activated, i.e. those scheduled within the data processing scheduler.
Arguments:
Return Type:void
Privilege:Public

**SelectActivities** - Builds the collection for all activities within the plan that lies within the time interval specified.
Arguments:Interval: interv
Return Type:void

Privilege:Public

**~PlActivities -** Destructor method.
Arguments:
Return Type:void
Privilege:Public

**Associations:**

The PlActivities class has associations with the following classes:
Class: PlPlanB iscontainedina
Class: PlActivePlan selectactivitiesusing

### 4.3.24 PlActivity Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class describes an item within a plan. The activity class is a base class within a specialization heirachy descrining the different activities which occur in the production plan.

**Attributes:**

**myActivityId -** Database key for activity
Data Type:RWCString
Privilege:Private
Default Value:

**myPredictedStart -** The predicted start time of an activity
Data Type:RWTime
Privilege:Private
Default Value:

**myPredictedStop -** The predicted end time of an activity.
Data Type:RWTime
Privilege:Private
Default Value:

**myPriority -** Priority of the activity within the plan.
Data Type:EcTInt
Privilege:Private

Default Value:

**Operations:**

**Cancel** - Abstract operation for the cancelling of an activity within the Data Processing subsystem.
Arguments:
Return Type:void
Privilege:Public
This is an abstract operation

**Modify** - Abstract operation for the modification of an activity within the Data Processing subsystem.
Arguments:
Return Type:void
Privilege:Public
This is an abstract operation

**PlActivity** - Default Constructor
Arguments:
Return Type:void
Privilege:Public

**PlActivity** - constructs a new activity for a plan - this is used when importing a plan from a remote DAAC
Arguments:RCWString, RWTime, RWTime, EcTInt
Return Type:Void
Privilege:Public

**Schedule** - Abstract operation for the scheduling of an activity within the Data Processing subsystem.
Arguments:
Return Type:void
Privilege:Public
This is an abstract operation

**Status** - Abstract operation for the statusing of an activity from the Data Processing subsystem.
Arguments:
Return Type:void
Privilege:Public
This is an abstract operation

**Associations:**

The PlActivity class has associations with the following classes:
Class: PlPlanB contains
Class: PlRescUseThreshNB placeholdersforon-demandjobscreatedusing
PlActivities (Aggregation)

### 4.3.25 PlAlternateDataGranuleNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:

**Attributes:**

**myAccepted** - Indicates whether the data granule, when available, has passed any metadata checks. Defaults to TRUE for data granules that do not undergo metadata checks.
Data Type:EcTBoolean
Privilege:Private
Default Value:

**myOrder** - The relative priority of an alternate input data granule.
Data Type:EcTInt
Privilege:Private
Default Value:

**myPrimaryType** - The data type of the primary alternate input associated with this data granule.
Data Type:PlDataTypeReq
Privilege:Private
Default Value:

**myTemporalFlag** - Indicates whether this alternate data granule (if TRUE) can be replaced with the most recent data granule of the same type, if one can't be found in the current production request time frame.
Data Type:EcTBoolean
Privilege:Private
Default Value:

**myTimerNB** - The amount of time the Subscription Manager will wait for an alternate input data granule to arrive.
Data Type:RWTime
Privilege:Private
Default Value:

**myType -** Indicates the type of alternate input: required, primary, or backup.
Data Type:enum
Privilege:Private
Default Value:

**Operations:**

**ChangeOrder -** Changes the priority order of an alternate input data granule. The relative order for the associated alternate data granules for the DPR is retained.
Arguments:RWCString:DataGranuleId
Return Type:Void
Privilege:Public

**GetTimerNB**
Arguments:

**PlAlternateDataGranule**
Arguments:

**UpdateTimerNB -** Updates the amount of time the Subscription Manager will wait for an alternate input data granule to arrive.
Arguments:RWTime:NewTime
Return Type:Void
Privilege:Public

**~PlAlternateDataGranule**
Arguments:

**Associations:**
The PlAlternateDataGranuleNB class has associations with the following classes:
None

### 4.3.26 PlAlternateNB Class

Parent Class:PlDataTypeReq
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class defines an alternate input to a PGE. It is used for PGE that has multiple inputs, and allows a secondary choice if on (or more) or its primary inputs are not available.

**Attributes:**

**myDefaultOrder** - Default priority of an alternate input data type.
Data Type:EcTInt
Privilege:Private
Default Value:

**myDefaultTimer** - The default amount of time that the Subscription Manager will wait for an alternate input data type to arrive.
Data Type:RWTime
Privilege:Private
Default Value:

**myPrimary** - This points to the primary input for which this input is an alternate.
Data Type:PlDataTypeReq
Privilege:Private
Default Value:

**myTemporalFlag** - Indicates (if TRUE) whether the most recent alternate input data granule can be used instead of the primary data granule of the same data type, if a primary data granule can't be found for the production request time frame.
Data Type:EcTBoolean
Privilege:Private
Default Value:

**myWaitFor** - This attribute indicates whether or not Planning should wait for this alternate input after the alternate inputs timer has expired. If the value is set to false, then Planning will allow the PGE to be executed without this input.
Data Type:EcTBoolean
Privilege:Private
Default Value:

**Operations:**

**PlAlternate** - This is the constructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**~PlAlternate** - This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlAlternateNB class has associations with the following classes:
None

### 4.3.27 PlCluster Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class defines the Cluster of tiles used by the PGE. A cluster is set up to allow a group of tiles to be scheduled at one time that use the same or similar geographic inputs.

**Attributes:**

**myLastOrbit** - The number of the orbit that contains the last data granule in this cluster.
Data Type:EcTInt
Privilege:Private
Default Value:

**myNumTiles** - The number of tiles for this cluster.
Data Type:EctInt
Privilege:Private
Default Value:

**myPredictedAvailability** - The predicted availability time of all data granules in this cluster.
Data Type:RWTime
Privilege:Private
Default Value:

**Operations:**

**PlCluster** - This is the constructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**PlCluster** - This constructor takes in a LastOrbit, a PredictedAvail, and a NumTiles and fills in the attributes when the class is created.
Arguments:EctInt Tiles, RWTime PredAvail, EctInt LastOrbit
Return Type:Void
Privilege:Public

**ReturnLastOrbit** - This returns the value in myLastOrbit.
Arguments:
Return Type:EctInt
Privilege:Public

**ReturnNumTiles** - This method returns myNumTiles.
Arguments:
Return Type:EctInt
Privilege:Public

**ReturnPredictedAvail** - This method returns the value in myPredictedAvailability.
Arguments:
Return Type:RWTime
Privilege:Public

**Update** - This method updates the specified Attribute with the specified Value. If the attribute does not exist, or the value is out of range a "bad" return code is returned.
Arguments:RWCString Attribute, RWCString Value
Return Type:EctInt
Privilege:Public

**~PlCluster** - This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlCluster class has associations with the following classes:
Class: PlTile
Class: PlTileScheduledNB

### 4.3.28 PlComputer Class

Parent Class:PlResource
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class describes the computers which are part of the resource configuration for the production system.

**Attributes:**

/ **myMaxDiskSpace** - Derived max disk space
Data Type:int
Privilege:Private
Default Value:
This is a Derived Attribute.

**myCPUs** - The number of CPUs within the computer
Data Type:int
Privilege:Private
Default Value:

**myDiskList** - Describes the disks associated with the computer.
Data Type:List
Privilege:Private
Default Value:

**myOperatingSystem** - The operating system name and version
Data Type:String
Privilege:Private
Default Value:

**myPerProcessRam** - The operating system's allocation of ram per process
Data Type:int
Privilege:Private
Default Value:

**myTotalRam** - The total Ram for the computer
Data Type:int
Privilege:Private
Default Value:

**Operations:**

**AddDisk** - Add a disk to the list
Arguments:PlDisk
Return Type:void
Privilege:Public

**PlComputer** - Class constructor
Arguments:
Return Type:void
Privilege:Public

**RemoveDisk** - remove a disk from the list
Arguments:PlDisk

Return Type:void
Privilege:Public

**~PlComputer** - Class destructor
Arguments:
Return Type:void
Privilege:Public

**Associations:**

The PlComputer class has associations with the following classes:
Class: PlDiskPartition
Class: PlString
PlService (Aggregation)

### 4.3.29 PlDASDelta Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
A list of the deltas for each data type.

**Attributes:**

**myDataTypeId** - Used to match this delta with a particular data type
Data Type:RWCString
Privilege:Private
Default Value:

**myDelta** - This is the maximum amount of time between when a Data Granule was predicted to arrive in the Planning Data Availability Schedule (PDAS) and the previous PDAS with the time frame specified before a limited automatic replan should be triggered.
Data Type:RWTime
Privilege:Private
Default Value:

**Operations:**

**GetDelta** - This operation returns the delta for a particular data type
Arguments:RWCString
Return Type:RWTime

Privilege:Public

**Associations:**

The PlDASDelta class has associations with the following classes:
Class: PlDASDifferent

### 4.3.30 PlDASDifferent Class

Parent Class:PlReplanCriteria
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
Newer Annotation: One of the replan criteria - this object is used to compare a newly-received DAS or PDAS to the previous one of the same type.

**Attributes:**

**myTimeFrame** - This is the time frame (from the time the PDAS or FOS DAS was received) within the predicted times of the data granules should be checked.
Data Type:RWTime
Privilege:Private
Default Value:

**Operations:**

**AddToReport** - This operation adds information about any data that is now predicted to arrive later than expected and the impacts of this late data on DPRs waiting for that data to a report that will be displayed so the operator can determine whether a replan should be done.
Arguments:PlDataGranule &, PlDPR &, RWTime
Return Type:RWCString
Privilege:Public
PDL: //Create a string
// with Data type, start time, stop time
// original predicted availability (From RWTime parameter)
// new DAS predicted availability (From PlDataGranule)
// DPR/PGE waiting for this data
// Predicted start time of this DPR
//Return this string

**CheckReplan** - Compares the data availability times from a new Planning Data Availability Schedule or FOS Detailed Activity Schedule with the previous version to see

if any data granules expected within my TimeFrame are now predicted to come myDelta later than expected.
Arguments:PlDataAvailabilityTimes &, PlDataAvailibilityTimes &
Return Type:EcTBoolean
Privilege:Public
PDL: //For first DAT passed in
//For each record in the DAT

```
//    If Availability time is within myTimeFrame
//        Search for matching record in second DAT
//
//        If PredictedAvailability from first DAT
//        is later than time in second DAT by
//        a greater amount than myDelta for this data type,
//
//            Search database for PlDataGranule that matches this record.
//            If one exists,
//                Search database for DPRs that are in the current
//                active plan that use this data,
//                and check that the Predicted start time of these DPRs
//
//            If any DPR is predicted to start before the new predicted
//            availability of the data granule
//                if first failure, put general information about PDPS DAS or
//                FOS DAS in myReport
//
//                call PlDASDifferent::AddToReport
//                        (PlDataGranule &, PlDPR &, RWTime)
//                where RWTime is the original predicted time from DAT
// end FOR
//If any failures, call PlOnDemandExceed::NotifyOperator
```

**Associations:**

The PlDASDifferent class has associations with the following classes:
Class: PlDASDelta
Class: PlDataAvailabilityTimes isusedtocheckforreplanby

### 4.3.31 PlDASNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:False
Purpose and Description:

This class contains the meta data information found in an FOS Detailed Actiity Schedule.

**Attributes:**

**myMessageType** - This attribute identifies the type of message being transmitted. (i.e. DAS)
Data Type:RWCString
Privilege:Private
Default Value:

**myNumDaysInFile**

**myNumRecordsinFile** - Identifies the number of records contained in the file.
Data Type:EcTInt
Privilege:Private
Default Value:

**myRecordTerminator** - Identifies the end of the Detailed Activity Schedules header record.
Data Type:EcTInt
Privilege:Private
Default Value:

**myScheduleStartTime** - Define the earliest activity start time contained in the Detailed Activity Schedule.
Data Type:RWTime
Privilege:Private
Default Value:

**myScheduleStopTime** - Defines the start time of the latest activity found in the Detailed Activity Schedule.
Data Type:RWTime
Privilege:Private
Default Value:

**mySource** - This attribute identifies the sender of the message (i.e. EOC)
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**PlDASNB** - Constructor
Arguments:
Return Type:PlDASNB

Privilege:Public

**~PlDASNB** - Destructor
Arguments:
Return Type:Void
Privilege:

**Associations:**
The PlDASNB class has associations with the following classes:
PlFOSDASFile (Aggregation)

### 4.3.32 PlDATRecord Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
Contains the predicted availability for a particular data granule received from source outside of the local DAAC.

**Attributes:**

**myDataTypeId** - Matches the myDataTypeID from PlDataGranule and PlDataType
Data Type:RWCString
Privilege:Private
Default Value:

**myPredictedAvailability** - The time that this data granule is predicted to be available at the local DAAC
Data Type:RWTime
Privilege:Private
Default Value:

**myStartTime** - The start time of the data - matches PlDataGranule::myStartTime
Data Type:RWTime
Privilege:Private
Default Value:

**myStopTime** - The stop time of the data - matches PlDataGranule::myStopTime
Data Type:RWTime
Privilege:Private
Default Value:

**myTileId** - For data granules that are geographical tiles instead of time continuous data, this attribute will note which tile within the start time and stop time of the orbit set (an orbit set covers the entire earth once) this data granule matches. Tile attributes, including the latitude and longitude of the coordinates that bound this tile, are found in the PGE Profile for PGEs that produce this data type. For data granules that are not tiles, this attribute would be set to zero.
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**MatchPlDataGranule** - Returns any data granule from the PDPS database that matches this PLDATRecord.
Arguments:
Return Type:PlDataGranule
Privilege:Public

**UpdateAvailabilityTime** - This operation will update the availability time of the data type to reflect the actual availability on the data server.
Arguments:RWTime
Return Type:EcUtStatus
Privilege:Public
PDL:{
Update Availabilitiy will use information passed in from PlDATRecords
regarding the completion time of the data type, and call
CalculateMyMoving Average to compute this transfer time.
call PlDataTransferHistory::CalculateMyMovingAverage()
Based on new moving average for transfer of data to data server and the
predicted completion time calculate Predicted Availability
PlDATRecord:myPredictedAvailability =
PlSourceToDataDsHistoryNB:myMovingAverageTime + CompletionTime .
}

**Associations:**

The PlDATRecord class has associations with the following classes:
Class: PlDataAvailabilityTimes
Class: PlDataAvailabilityTimes containsmany
Class: PlDataTranferHistory isusedtopredictdataavailabilityatDSS
Class: PlDataTypeB matches
Class: PlDataGranule updatesavailabilityof
Class: PlDataGranule updatestheavailabilityof

### 4.3.33 PlDBMSProxyAgent Class

Parent Class:EcPfManagedServer
Public:No
Distributed Object:No
Purpose and Description:
This class describes the main body for the proxy agent required to manage the lifecycle services of the PDPS DBMS. The proxy agent will be an almost template re-use of the MSS capabilities.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**PfShutdown** - Shut down Sybase SQL server containing the PDPS database
Arguments:
Return Type:Void
Privilege:Public

**PlDBMSProxyAgent** - class constructor function
Arguments:
Return Type:Void
Privilege:Public

**Start** - starts the Sybase SQL server containing the PDPS database
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlDBMSProxyAgent class has associations with the following classes:
None

### 4.3.34 PlDPRB Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:

This class describes an individual run of a PGE.

**Attributes:**

**myActualStartTime -** The actual time that this DPR began processing in DPS
Data Type:RWTime
Privilege:Private
Default Value:

**myBaselineTime -** The time the DPR is predicted to begin in that latest baseline plan.
Data Type:RWTime
Privilege:Private
Default Value:

**myCompletionState -** Status indicator describing active status of data processing request
Data Type:RWCString
Privilege:Private
Default Value:

**myDprId -** Unique identifier for the DPR instance
Data Type:RWCString
Privilege:Private
Default Value:

**myInputDataInstanceList -** List describing all the input files required within for the PGE
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**myNextConditionalDPR -** For PGEs with mode-based activations, a reference to the next DPR associated with this one. For normal PGEs, this will be null for the first DPR in the PR list (PlProductionRequest::myDPRs).
Data Type:PlDPR&
Privilege:Private
Default Value:

**myOutputDataInstanceList -** List describing all the output files to be produced by the PGE
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**myPgeId -** The id of the PGE that is being executed in this DPR
Data Type:RWCString
Privilege:Private

Default Value:

**myPredictedStartTime**
Data Type:RWTime
Privilege:Private
Default Value:

**myPriority** - Priority for the data processing request is inherited from the production reqeust, but may be modified individually
Data Type:EcTInt
Privilege:Private
Default Value:

**myProductionRequestId** - The id of the Production Request of which this DPR is a part.
Data Type:RWCString
Privilege:Private
Default Value:

**myStartTime** -

**myStopTime** -

**myUR** - The Universal Reference generated by Data Server for the product generated by this processing request.

**Operations:**

**Cancel** - Cancel the DPR within the data processing job scheduler
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL:{
// calls the cancel DPR job method of the DpPrJobScheduler
}

**CheckAvailability** - Checks to see if all the data dependies for a DPR have been fulfilled. Returns true or false.
Arguments:
Return Type:Boolean
Privilege:Public
PDL:{
Get the input granules list for the list of granules needed for Dpr.
PlDPRB::GetInputGranuleList()
Iterate through the input data list to check the availability of the granule

```
For (index = 1; index <= TotalInList; index++)
{
  Call Get Availability to determine if granule is available
  PlDataGranule::GetAvailability
  if (GranuleExists = false)
  {
    If not check to see if the granule is a required granule
    if (PlAlternateDataGranule:myType = Required)
    {
      If the granule is required, then CheckAvailability fails
      return failed
    }
  }
  else
  {
    Granule exists
    if (PlAlternateDataGranule:myType = Required)
    {
      set RequiredAvailable to True to indicate that the required data
      is available
    }
    Check if the granule passed the QA check
    if (PlAlternateDataGranule:myAccepted = True)
    {
      If granule passed and it is a primary input
      if (PlAlternateDataGranule:myType = Primary)
      {
        increment the count of primaries that are available. TotPrim ++
      }
      If granule passed and it is not a primary input
      if (PlAlternateDataGranule:myType = Backup)
      {
        increment the count of backups that are available. TotBackup ++
      }
    }
  }
} loop
If the required and all the primary are availably release job
if (TotPrim = TotPrimaryRequired and RequiredAvailable)
{
  Return True
  Cancel timer if it has been set
  if (TimerIsSet)
  {
    PlTimer:CancelTimer
```

```
    }
  }
  If the required and not all the primary is available set timer, or if the temporal
  flag is set, indicating that a newer version of the same type is pending, set timer
  if (TotPrim < TotPrimaryRequired .or. (TotPrim = TotPrimaryRequired .and.
    PlAlternateDataGranule:myTemporalFlag = true)  and RequiredAvailable)
  {
    Check if the timer is set and set if necessary
    if (PlTimer:myTimerStart = false)
    {
      Return False
      Get wait time and start timer
      PlDpr:: GetMyTimer
      PlTimer::StartTimer()
    }
  }
}
```

**CheckPredictedAvailabilityNB -** Determines the predicted availability of the input data granules
Arguments:
Return Type:EcTVoid
Privilege:Public

**GetCommandString -** Operation to format the command string that is sent to Data Server for a Query.
Arguments:RWCString:DataType
Return Type:RWCString
Privilege:Public

**GetInputGranuleList -** Read the list of required data granules from the PDPS Database for the PGE to be used.
Arguments:
Return Type:RWTValSlist<RWCString>
Privilege:Public

**GetLogicalId -** Operation to used to get the logical Id of the Data Type used for this production request
Arguments:RWCString:DataType
Return Type:Void
Privilege:Public

**GetMyPGEType -** This operation will get the PGE Type in order to determine whether or not a PGE is a tiling PGE.
Arguments:

Return Type:RWCString
Privilege:Public
PDL:
{
  Instantiate an instance of PlPge to find the pge type associated with PGE
}

**GetMyTimer -** Returns the length of time a timer should wait for the primary input data after receiving the required inputs before using the backups.
Arguments:
Return Type:RWTime
Privilege:Public
PDL:{
   gets the wait time associated with the data
   GETPlAlternateDataGranuleNB:myTimerNB
}

**GetOutputGranuleList -** Read the list of date granules that the Pge will generate from the PDPS Database.
Arguments:
Return Type:RWTValSlist<RWCString>
Privilege:Public

**Modify -** Modify the DPR within the data processing job scheduler. The attributes of the DPR that may have been modified are priority and the anticipated start / stop times of the DPR used to set the alarms in the data processing scheduler.
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL:{
// calls the modify DPR job method of the DpPrJobScheduler
}

**PlDPR -** Constructor method
Arguments:

**PlDPR -** Constructor
Arguments:RWCString:DprId
Return Type:PlDPR
Privilege:Public

**Release -** Releases the DPR within the data processing job scheduler. The DPR is released when the data dependencies of that DPR have been fulfilled.
Arguments:
Return Type:EcTVoid

Privilege:Public
PDL:{
// calls the release DPR job method of the DpPrJobScheduler
}

**Schedule** - schedule the DPR within the data processing job scheduler
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL:{
// calls the create DPR job method of the DpPrJobScheduler
}

**Status** - status the DPR within the data processing job scheduler
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL:{
// calls the status DPR job method of the DpPrJobScheduler
}

**~PlDPR** - Destructor method
Arguments:
Return Type:void
Privilege:Public

**Associations:**
The PlDPRB class has associations with the following classes:
Class: DpPrScheduler
Class: PlPGE
Class: PlProdStratNB basesprioritieson
Class: PlOnDemandPRNB composes
Class: PlPGE generates
Class: PlDataGranule input
Class: PlPGEActivity isaplanned
Class: PlProductionRequestB isbrokenupintoindividual
Class: PlProductionRequestB ismadeupof
Class: PlResourceRequirement isrequiredby
Class: PlTimer isusedtosettimeronalternateinputs
Class: PlDataGranule matchescorresponding
Class: PlDataGranule output
Class: DpPrScheduler schedules
Class: DpPrScheduler schedulesinprocessing
Class: PlDataGranule uses

PlDPRs (Aggregation)

### 4.3.35 PlDPRs Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Purpose and Description:
This is a collector class for the PlDPR class, and contains methods to select Data Processing Requests from the PDPS database and to iterate through them. This class may be implemented by a suitable Rogue Wave template class.

**Attributes:**

None

**Operations:**

**Add** - Add a DPR to the collection
Arguments:PlDPR: dpr
Return Type:void
Privilege:Public

**Delete** - Delete an DPR from the collection.
Arguments:PlDPR: dpr
Return Type:void
Privilege:Public

**First** - Returns the first DPR within the collection
Arguments:
Return Type:PlDPR
Privilege:Public

**Next** - Returns the next DPR within the collection (or NULL if no more activities).
Arguments:
Return Type:PlDPR
Privilege:Public

**PlDPRs** - Constructor method
Arguments:
Return Type:PlDPRs
Privilege:Public

**SelectDPRs** - Builds the collection for all DPRs within the plan that lies within the time

interval specified
Arguments:Interval
Return Type:void
Privilege:Public

**~PlDPRs** - Destructor method
Arguments:
Return Type:void
Privilege:Public

**Associations:**

The PlDPRs class has associations with the following classes:
None

### 4.3.36 PlDataAvailabilityTimes Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This object represents the PDPS Data Availability Schedule or the FOS Detailed Activity Schedule which was used to produce these data availability times records.

**Attributes:**

**myProductOrigin** - Where these data availability time records come from (example - DAAC name or EDOS)
Data Type:RWCString
Privilege:Private
Default Value:

**myStartTime** - The start time of the FOS DAS or PDPS DAS used to create these Data Availability Time Records
Data Type:RWTime
Privilege:Private
Default Value:

**myStopTime** - The stop time of the FOS DAS or PDPS DAS used to create these Data Availability Time Records
Data Type:RWTime
Privilege:Private

Default Value:

**Operations:**

**MakeDATFromFOSDASFile -** This operation will create a Data Availability Time Schedule from a PlFOSDASFile received from the Data Server.
Arguments:RWTValSlist<PlFOSDASFile>
Return Type:EcUtStatus
Privilege:Public
PDL:{

```
Using the information obtained from a PlFOSDASFile from FOS,
generate a DAT Schedule of the data needed at the local DAAC.
This function will use empirical information contained in
PlDataTransferHistory.
Call PlDataAvailabilityTimes::PlDataAvailabilityTimes constructor method
Get PlDataSchedules:PlFOSDASFiles
Get PlDASNB:myMetaDataName
Populate
PlDataAvailabilityTimes:myStartTime,PlDataAvailabilityTimes:myStopTime,
and PlDataAvailabilityTimes:myproductOrigin with meta data values from
PlDASNB.

for all PlEDASRecordNB
{
  Determine the volumn of data being collected
  Get PlEDASRecordNB:myDataRates
  Get PlEDASRecordNB:myActivityStartTime
  Get PlEDASRecordNB:myActivityStopTime
  The Data rate over the elapsed time will provide us with an estimate
  of the data type being collected during that time.  Other information
  received from FOS may help us to more accurately predict arrival time
  in the future.
  Search DataBase for DataTypeBeingCollected to see if this data
  type is subscribed to by DAAC
 if (DataTypeBeingCollected is found)
 {
    Call PlPDATRecord constructor for record
    populate PlDATRecord:myDataTypeId with DataTypeBeingCollected,
    PlDATRecord:myStartTime,PlDATRecord:myStopTime,
    PlDATRecord:myTileId (if tiling), with
    values from  PlEDASRecordNB
    call PlDATRecord::UpdateAvailabilityTime() to get
    transfer time.
 }
} loop
```

**MakeDATFromPDASFile -** This operation will create a Data Availability Time Schedule from a PlPDASFile received from the Data Server.
Arguments:RWTValSlist<PlDASFile>
Return Type:EcUtStatus
Privilege:Public
PDL:{

```
  Upon notification of the arrival of a PlPDASFile from another DAAC, this
  operation generates a DAT Schedule for the data needed at the local DAAC.
  This function will use the empirical information contained in
  PlDataTransferHistory to predict the actual arrival at the data server.
  Call PlDataAvailabilityTimes::PlDataAvailabilityTimes constructor method
  Get PlPDASFile:myMetaDataFileName
  Populate
  PlDataAvailabilityTimes:myStartTime,PlDataAvailabilityTimes:myStopTime,
  and PlDataAvailabilityTimes:myProductOrigin with meta data values from
  PlPDASMetaData.
  Get PlPDASFile:myFileName
  for all PlPDASRecords
  {
    Search DataBase for PlPDASRecords:myDataTypeId to see if this data
    type is subscribed to by DAAC
    if (PlPDASRecords:myDataTypeId is found)
    {
      Call PlPDATRecord constructor and populate DAT record
      PlDATRecord:myDataTypeId,PlDATRecord:myStartTime,
      PlDATRecord:myStopTime, PlDATRecord:myTileId, with values from
      PlPDASRecords
      Get PlDASRecords:myTimeCompleted
      call PlDATRecord::UpdateAvailabilityTime() to get
      transfer time.
    }
  } loop
}
```

**RetrieveDATRecords -** This operation retrieves all the DAT records for a particular PDPS DAS or FOS DAS.
Arguments:RWTValSlist<PlDATRecord>
Return Type:EcUtStatus
Privilege:Public

**RetrieveDATRecords -** This operation retrieves all the DAT records for a particular PDPS DAS or FOS DAS.
Arguments:RWTvalSlist<PlDATRecord>

**RetrieveDATRecords -** This operation retrieves all the DAT records for a particular PDPS DAS or FOS DAS.
Arguments:RWTValSlist<PlDATRecord>
Return Type:EcUtStatus
Privilege:Public

**Associations:**

The PlDataAvailabilityTimes class has associations with the following classes:
Class: PlSubMsgCb NotificationofarrivalofDAT
Class: PlDATRecord
Class: PlDATRecord containsmany
Class: PlDASDifferent isusedtocheckforreplanby
Class: PlReplanCriteria isusedtocheckforreplanby
Class: PlDataSchedules isusedtogetDASfromDSS

### 4.3.37 PlDataDependencies Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
Used to store information about the data dependencies of an imported plan.

**Attributes:**

**myDataTypeId -** The data type of this data - matches PlDataType::myDataTypeId
Data Type:RWCString
Privilege:Private
Default Value:

**myStartTime -** The start time and date of the data
Data Type:RWTime
Privilege:Private
Default Value:

**myStopTime -** The stop time and date of the data
Data Type:RWTime
Privilege:Private
Default Value:

**myTileId -** For data granules that are geographical tiles instead of time continuous data, this attribute will note which tile within the start time and stop time of the orbit set (an orbit set covers the entire earth once) this data granule matches. Tile attributes, including the

latitude and longitude of the coordinates that bound this tile, are found in the PGE Profile for PGEs that produce this data type. For data granules that are not tiles, this attribute would be set to zero.
Data Type:RWCString
Privilege:Private
Default Value:

**myTypeDependency -** Indicates whether this data is needed as input for this activity (with an 'I') or is output for this activity ('O') and needed as input by another activity
Data Type:EcTChar
Privilege:Private
Default Value:

**Operations:**

None

**Associations:**

The PlDataDependencies class has associations with the following classes:
Class: PlImportedActivity mayhave

### 4.3.38 PlDataGranule Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class describes individual instances or granules of data types.

**Attributes:**

**myActualAvailability -** Date and time that data was made available to subscription manager.
Data Type:RWTime
Privilege:Private
Default Value:

**myAvailability -** Flag to indicate availability of data
Data Type:EcTBoolean
Privilege:Private
Default Value:

**myBaselineTime -** The availability time of the data granule according to the baselined active plan agreed to amongst DAACs requiring the data granule.
Data Type:RWTime
Privilege:Private
Default Value:

**myDataGranuleId -** ID of data granule.
Data Type:RWCString
Privilege:Private
Default Value:

**myDataTypeId -** The identifier of the data type of the data granule.
Data Type:RWCString
Privilege:Private
Default Value:

**myESDTParmVals -** Selected metadata fields associated to the data type required to determine suitability in the production (such as quality info or geophysical attributes)
Data Type:GlParameterList
Privilege:Private
Default Value:

**myPredictedAvailability -** Predicted time at which the data will be available in ECS, used to determine PGE schedule estimates.
Data Type:RWTime
Privilege:Private
Default Value:

**myPredictedStagingTime -** This is an ROUGH estimate of the amount of time that this file will take to stage, based on the size of the file, the age of the file (determines whether its in deep storage or on a local disk) and whether the file was remote or local.
Data Type:RWTime
Privilege:Private
Default Value:

**myStartTime -** The start time and date of the data
Data Type:RWTime
Privilege:Private
Default Value:

**myStopTime -** The stop time and date of the data
Data Type:RWTime
Privilege:Private
Default Value:

**myTileId** - For data granules that are geographical tiles instead of time continuous data, this attribute will note which tile within the start time and stop time of the orbit set (an orbit set covers the entire earth once) this data granule matches. Tile attributes, including the latitude and longitude of the coordinates that bound this tile, are found in the PGE Profile for PGEs that produce this data type. For data granules that are not tiles, this attribute would be set to zero.
Data Type:RWCString
Privilege:Private
Default Value:

**myUR** - Universal Reference by which to reference the granule within the data server.
Data Type:EcUrUR
Privilege:Private
Default Value:

**Operations:**

**Create** - Create a entry in the DBMS if this is a unique instance of the granule.
Arguments:RWTime:start, RWTime:stop
Return Type:void
Privilege:Public

**DeleteDummyGranule** - This operation is used for PGEs that produce spatial tiles to delete that dummy data granule that represents the last data granule which will be collected (or processed for higher level products, assuming the data granules are produced in pretty much the same order that they are collected). The Subscription Manager uses the notification that this data has arrived to begin to query the Data Server for all the actual data granules that are needed as input to produce this tile.
Arguments:RWCString:myDataGranuleId
Return Type:Void
Privilege:Public

**DeleteGranule** - This operation will delete the predicted granule created by the production request editor for tiling.
Arguments:RWCString:myDataGranuleId

**FindAssociatedDPRs** - Method to determine the data processing requests associated to the data granule
Arguments:
Return Type:PlDPRs
Privilege:Public

**GetAvailability** - Returns whether the data item is available within the ECS or not.
Arguments:
Return Type:EcTBoolean

Privilege:Public

**PlDataGranule** - Constructor method
Arguments:
Return Type:PlDataGranule
Privilege:Public

**PredictStagingTime** - Computes the value stored in myPredictedStagingTime.
Arguments:
Return Type:RWTime
Privilege:Public

**RegisterAvailability** - Method to register that a data instance that was predicted has arrived, and to record the UR and metadata associated to that data.
Arguments:EcUrUR:instUR, GlParameterList: instESDTParmVals
Return Type:void
Privilege:Public

**~PlDataGranule** - Destructor method
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlDataGranule class has associations with the following classes:
Class: PlSubMsgCb Notificationofarrivalofgranule
Class: PlSourcetoDsHistoryNB UpdatesArrivalTime
Class: PlDPRB input
Class: PlDPRB matchescorresponding
Class: PlDPRB output
Class: PlDataTypeB populates
Class: PlFile simulatesstroageof
Class: PlDATRecord updatesavailabilityof
Class: PlDATRecord updatestheavailabilityof
Class: PlDPRB uses

### 4.3.39 PlDataScheduled Class

Parent Class:PlPGE
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:

This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined from some regular time period, such as an hour, a day, a week, a month etc.

**Attributes:**

**myInputDataId -** ID of the input data based on the arrival of which the PGE is scheduled to run
Data Type:int
Privilege:Private
Default Value:

**myNoOfInstancesPerPgeRun -** ID of the input data based on the arrival of which the PGE is scheduled to run
Data Type:int
Privilege:Private
Default Value:

**myPrimaryDataSource -** Identifies the source of the data
Data Type:String
Privilege:Private
Default Value:

**Operations:**

**GenerateDPRs -** Generate the data processing request that fulfill the production request
Arguments:PlProductionRequest
Return Type:void
Privilege:Public
PDL:{

```
// Select the primary input granules that are within the period of
// the production request

// Generate a DPR for each myNoOfInstancesPerPgeRun granules
// of input data

// Invoke the select data method for the DPR
}
```

**Associations:**

The PlDataScheduled class has associations with the following classes:
None

### 4.3.40 PlDataSchedules Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This Class interfaces with the data Server to aquire Detailed Activity Schedules from FOS and Data Availability Schedules from other DAACs.

**Attributes:**

**myFileName**
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**GetDASfile -** This operation is used to get a DAS file from the Data Server.
Arguments:RWCString
Return Type:Void
Privilege:Public

**Associations:**

The PlDataSchedules class has associations with the following classes:
Class: GlParameterList holdsparametersforAquire
Class: PlInstrumentModes isusedtoconstructinstrumentmodeschedule
Class: PlDataAvailabilityTimes isusedtogetDASfromDSS
Class: GlCallback receivescallbackfromAquire
Class: DsClCommand usedforAquire
Class: DsClESDTReferenceCollector usedforAquire
Class: DsClRequest usedforAquire

### 4.3.41 PlDataSource Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is the base class that provides the methods for predicting when external data will arrive within the ECS from SDPF or NOAA for example. The Data Sources are specialized

to describe the different ways that predictions are obtained.

**Attributes:**

**myPredictedMethod** - Describes the method by which the data availability prediction occurs - e.g., routine arrival, arrival at scheduled times, FOS based prediction.
Data Type:enum
Privilege:Private
Default Value:

**mySupplierName** - Identifies the supplier of the data.
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**PlDataSource** - default constructor
Arguments:
Return Type:Void
Privilege:Public

**PlDataSource** - overloaded constructor from data type id
Arguments:const RWCString &
Return Type:Void
Privilege:Public

**PredictArrivals** - Predicts data instance arrivals for a given period, pure virtual operation.
Arguments:const PlTime &, const PlTime &
Return Type:EcTVoid
Privilege:Public
This is an abstract operation

**~PlDataSource** - destructor method
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlDataSource class has associations with the following classes:
Class: PlDataTypeB checks
Class: PlDataSourceFactory createdby
Class: PlSubscriptionSubmitIF submitssubscriptionsusing

### 4.3.42 PlDataSourceFactory Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:

**Attributes:**

None

**Operations:**

**Create -** Constructs a DataSource object based upon its id.
Arguments:const RWCString &
Return Type:PlDataSource *
Privilege:Public

**PlDataSourceFactory**
Arguments:
Return Type:Void
Privilege:Public

**~PlDataSourceFactory**
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlDataSourceFactory class has associations with the following classes:
Class: PlDataSource createdby

### 4.3.43 PlDataTranferHistory Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class will computes a moving average of the time it takes for the data to arrive from EDOS or another DAAC to the Data Server.

**Attributes:**

None

**Operations:**

**CalculateMyMovingAverage** - This operation will calculate the moving average of the transfer time of a data type from the source to the data server. The standard deviation will also be calculated to provide the acceptable delta for the predicted availability of the arrival of data.

```
Arguments:RWCString:myDataTypeId
Return Type:RWTime
Privilege:Public
PDL:{
   Find record in database associated with data type
   PlDataTranferHistory::FindMyDataType()
   Calculate the last transfer time of data type to the data server to
   based on last predicted and last actual time.
   LatestTransferTime = ABS(myLastActual - myLastpredicted)
   if (ABS(myPreviousTransferTime - LatestTranferTime) >
     PlSourceToDsHistoryNB:myStandardDeviation)
   {
     if latest transfer time is outside of the standard deviation, this
     time should not be used in computing a new moving average.
     Set PlDataTranferHistory:myMovingAverageFlag == False
     return;
   }
   Compute moving average and standard deviation
   for (index = 1; index <=
   PlSourceToDsHistoryNB:myNumberOfEstimates;index++)
   {
     Sum prior transfer times to calculate average
     Total = Total + PlSourceToDsHistoryNB:myPreviousTransferTimes[index]
     Square these times to calculate the standard deviation
     SumOftransfersSqrd = SumOfTransfersSqrd +
     (PlSourceToDsHistoryNB:myPreviousTransferTimes[index])**2
   }
   Average = Total / PlSourceToDsHistoryNB:myNumberOfEstimates
   StandardDeviation = SQRT ( (SumOfTransfersSqrd -
   (PlSourceToDsHistoryNB:myNumberOfEstimates*(Average)**2)))/
   (PlSourceToDsHistoryNB:myNumberOfEstimates - 1)
   Update persistent information pertaining to data.
   PlSourceToDsHistoryNB:UpdateHistoricalinfo
}
```

**FindMyDataType -** This operation will find the transfer history record associated with the Data Availability Times record received.
Arguments:RWCString:myDataTypeId
Return Type:EcUTStatus
Privilege:Public
PDL:{
Parse through list of PlSourceToDsHistoryRecordNB records to find Data Type associated with this prediction.
Get PlSourceToDsHistoryNB:myDataTypeId
}

**Associations:**

The PlDataTranferHistory class has associations with the following classes:
Class: PlSourcetoDsHistoryNB
Class: PlDATRecord isusedtopredictdataavailabilityatDSS

### 4.3.44 PlDataTypeB Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class describes a data type known to the planning subsystem. This is a description of an input or output type, distinct to a granule or instance of the data type. The class is an abstraction or proxy that describes one of the Data Server ESDTs. The class captures data and operations that are required to subscribe and receive notification from the Data Server when a new instance of the Data Type arrives.

**Attributes:**

**myArchiveCenter -** DAAC where Data Type is archived.
Data Type:RWCString
Privilege:Private
Default Value:

**myCatalogueCatagory -** Catalogue catagory indicates whether the Data Type is valid for a production request, which would indicate it is generated from a PGE at a site, otherwise the Data Type is some intermediate file, or input file received from another site.

**myDServURString -** Universal Reference to identify Data Server providing services (retreive, insert, inspect) for the Data Type.
Data Type:RWCString

Privilege:Private
Default Value:

**myDataTypeId** - This is the identified of the Data Type.
Data Type:RWCString
Privilege:Private
Default Value:

**myDescription** - This attribute provides a text description for the Data Type
Data Type:RWCString
Privilege:Private
Default Value:

**myDynamicFlag** - Indicates whether the Data Type is dynamic or static. Examples of dynamic are L0, L1 data sets etc. with a frequest update time. Examples of static are calibration files which only change with a new version of a PGE.
Data Type:EcTBoolean
Privilege:Private
Default Value:

**myESDTParmList** - A parameter list used within the inspect to the Data Server, to retreive the metadata associated to a newly arrived instance of the Data Type.
Data Type:GlParameterList
Privilege:Private
Default Value:

**myInstrumentName** - Instrument name associated with PGE
Data Type:RWCString
Privilege:Private
Default Value:

**myName** - Attribute describes the ESDT Name for the data set within the Data Server.
Data Type:RWCString
Privilege:Private
Default Value:

**myNominalSize** - Nominal size of the data type
Data Type:EcTFloat
Privilege:Private
Default Value:

**myProcessingCenter** - This is the name of the DAAC that produces this data type.
Data Type:RWCString
Privilege:Private
Default Value:

**myProvider** - the DAAC that is maintaining the data
Data Type:RWCString
Privilege:Private
Default Value:

**myQASubscription** - Captures whether a subscription has been set up for the QA of this data type.
Data Type:EcTBoolean
Privilege:Private
Default Value:

**mySatelliteName** - Satellite name associated to PGE
Data Type:RWCString
Privilege:Private
Default Value:

**myService** - The service (usually ACQUIRE) that serves the data
Data Type:RWCString
Privilege:Private
Default Value:

**mySpatialFlag** - If this flag is set, then the data granules for this type require spatial coordinates to define them as well as start and stop time of the data. This flag will identify data that has been placed into tiles.
Data Type:EcTBoolean
Privilege:Private
Default Value:EcDFalse

**mySubscriptionFlag** - Set to true if Planning currently has a subscription on this data type with the Data Server or Ingest
Data Type:EcTBoolean
Privilege:Private
Default Value:EcDFalse

**myUsedByCenter** - List of DAACs which use this Data Type as input for their processing. Used for InterDAAC planning.
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**Operations:**

**FindDataAvailability** - Ensures that the Data Granules that are required to fulfill a production request are captured in the PDPS database

```
Arguments:RWTime Start,RWTime Stop
Return Type:EcTVoid
Privilege:Public
PDL://
////find any existing data granules
//
//LatestTime = start of time frame
//
////if locally produced data type, query PDPS DB
//
//if myProcessingCenter == myDAAC {
//  query PDPS DB for any data granules in the time frame
//  call CreateDataGranules (LatestTime)
//}
//
//Call DSS Inspect and Query (see DID 313 scenario primitive for
//Data Server Inspect and Query) to get all datagranules for time period
//between LatestTime and end of time frame
//
//call CreateDataGranules (LatestTime)
//
//// check data availability times (DATs) for remote DAAC data granules
//
//if myProcessingCenter != myDAAC {
//  call QueryDATRecords (LatestTime, end of time frame, granuleList)
//  call CreateDataGranules (LatestTime)
//}
//
//// Submit subscriptions for future data
//for time > LatestTime && time < end of time frame {
//  if mySubscriptionFlag != TRUE
//    submit subscription (call PlSubscriptionSubmitIF::SubmitSubscription)
//  predict arrivals using PlRoutineArrival
//
//}
//
//CreateDataGranules (RWTime &LatestTime)
//{
//
//  for each datagranule found {
//    PlDataGranule()
//    PlAlternateDataGranule()
//
//    // perform metadata check
//    call PlMetaDataChecks::CheckforCondition
```

```
//    if (PlAlternateDataGranule::myAccepted)   // passed check
//      insert data granule in DB
//      // keep track of latest granule
//      if PlDataGranule.StopTime > LatestTime
//        LatestTime = PlDataGranule.StopTime
//
//      // if primary alternate input, set temporal flag to
//      // indicate to subscription mananger to check for most recent
//      // granule of the same type as primary, if primary not available
//      if (PlDataTypeReq::myType != Required) {
//        if (PlAlternate.myTemporalFlag == TRUE
//            & PlAlternateDataGranule.myPrimaryType == Null)
//          set PlAlternateDataGranule.myTemporalFlag == TRUE
//
//  }
//}
//
```

**GetAdService -** Used to get the advertisement title for a service for this data type. This advertisement title will be used when submitting a subscription.
Arguments:EcUtStatus &
Return Type:IoAdService *
Privilege:Public

**InspectDataArrival -** Creates a UR from the notification message received and uses the UR to extract the requried metadata (ESDT parameter list) from the Data Server.
Arguments:
Return Type:EcTVoid
Privilege:Public

**InstallReceiver -** Install a receiving queue in which to receive notification of subscription from the Data Server.
Arguments:
Return Type:EcTVoid
Privilege:Public

**MatchDataArrival -** Match the data arrival with the predictions within Data Granule table.
Arguments:
Return Type:EcTVoid
Privilege:Public

**PlDataType -** This is the default constructor for the class.
Arguments:
Return Type:PlDataType
Privilege:Public

**PlDataType -** This constructor takes in the ID of the Data Type to be created.
Arguments:RWCString:DataTypeId
Return Type:PlDataType
Privilege:Public

**PlDataType -** This is the default constructor for the class.
Arguments:
Return Type:PlDataType
Privilege:Public

**PlDataType -** This constructor takes in the ID of the Data Type to be created.
Arguments:RWCString:DataTypeId
Return Type:PlDataType
Privilege:Public

**QueryDATRecords -** This operation returns a list of DAT records for this data type for the specified interval.
Arguments:RWTime: StartTime, RWTime: StopTime, RWTlistSval<PlDATRecords>
Return Type:EcUtStatus
Privilege:Public

**RegisterDataArrival -** Procedure to be called on notification of data arrival, this a control procedure that manages the inspection of the data, comparison to the data instances, and then subsequent handling of associated DPRs to the arrived data.
Arguments:
Return Type:EcTVoid
Privilege:Public

**Submit -** Used to submit a subscription for this data type
Arguments:
Return Type:EcUtStatus
Privilege:Private

**Withdraw -** Used to withdraw a subscription for this data type
Arguments:
Return Type:EcUtStatus
Privilege:Public

**~PlDataType -** This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlDataTypeB class has associations with the following classes:
Class: DsClQuery
Class: DsClSubscription
Class: PlDataSource checks
Class: PlPGE input
Class: IoAdProductSearchCommand isusedtointerfacetotheAdvertisingServiceby
Class: PlDATRecord matches
Class: PlPGE output
Class: PlDataGranule populates
Class: PlSubscriptionSubmitIF updates
PlDataTypeCatalogue (Aggregation)
PlPGEProfile (Aggregation)

### 4.3.45 PlDataTypeCatalogue Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class describes a collection that lists the data types that may be generated by the production system. The class is constructed from the PlDataTypes class using the catalogue catagory attribute from that class.

**Attributes:**

None

**Operations:**

**CatalogueRequest** - Request to retrieve all data types the ECS site is capable of generating.
Arguments:
Return Type:catalogue:char*
Privilege:Public

**RetrieveDefPGE** - Retrieves the default PGE associated with a selected product.
Arguments:RWCString:ProductType
Return Type:RWCString:DefPGEName
Privilege:Public

**RetrieveDefProduct** - Retrieves the default product associated with a PGE that has been selected.
Arguments:RWCString:PGEID
Return Type:RWCString:DefProductName

Privilege:Public

**Associations:**

The PlDataTypeCatalogue class has associations with the following classes:
Class: PlProductionRequestUI storesrequestsin

### 4.3.46 PlDataTypeReq Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class contains the data that associates a PGE to an input data type.

**Attributes:**

**myCommandString -** This string is used to construct the acquire command for the data type, used to stage the data from the data server.
Data Type:RWCString
Privilege:Private
Default Value:

**myDataTypeId -** This is the identified of the Data Type that is required.
Data Type:RWCString
Privilege:Private
Default Value:

**myDataTypeReq -** This attribute specifies the data type requirement of a PGE. The string specifies the select statement to be applied to the Data Granule table, to determine the input Data Granules for a given Data Processing Request.
Data Type:RWCString
Privilege:Private
Default Value:

**myInspectString -** This attribute stores the command string sent to the Data Server to inspect fields of the metadata.
Data Type:RWCString
Privilege:Private
Default Value:

**myLogicalID -** The logical id relates to the PGE input identifier for a particular product type. This attribute is required within the SDP toolkit process control interface.

Data Type:EcTInt
Privilege:Private
Default Value:

**myNumNeeded** - This represents the number of inputs required by the PGE associated with PlDataTypeReq.
Data Type:EcTInt
Privilege:Private
Default Value:

**myPgeId** - This is the identifier of the PGE that requires this data.
Data Type:RWCString
Privilege:Private
Default Value:

**myQAThreshold** - This attribute describes the quality threshold to be applied to the data granules to "approve" their suitability for production before releasing a scheduled PGE. The threshold is specified in terms of the ESDT parameter list of the data type.
Data Type:RWCString
Privilege:Private
Default Value:

**myType** - This represents the type of input: Required, primary, or backup.
Data Type:enum
Privilege:Private
Default Value:

**Operations:**

**PlDataTypeReq** - Default constructor for the class
Arguments:
Return Type:EctVoid
Privilege:Public

**PlDataTypeReq** - This constructor takes in the PGE ID and Data Type ID of the Data Type Required class to be created.
Arguments:RWCString:PgeId, RWCString:DataTypeId
Return Type:EctVoid
Privilege:Public

**PlDataTypeReq** - Default constructor for the class
Arguments:
Return Type:EctVoid
Privilege:Public

**PlDataTypeReq -** This constructor takes in the PGE ID and Data Type ID of the Data Type Required class to be created.
Arguments:RWCString:PgeId, RWCString:DataTypeId
Return Type:EctVoid
Privilege:Public

**~PlDataTypeReq** - This is the destructor for the class.
Arguments:
Return Type:EctVoid
Privilege:Public

**Associations:**

The PlDataTypeReq class has associations with the following classes:
Class: PlMetaDataChecks
Class: PlMetaDataChecks performs

### 4.3.47 PlDiskPartition Class

Parent Class:PlResource
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class describes the disk resources for data production

**Attributes:**

**myBlockSize -** the block size for the partition
Data Type:int
Privilege:Private
Default Value:

**myDeviceID** - specifies the UNIX device id for the partition
Data Type:String
Privilege:Private
Default Value:

**myPartitionSize -** the total partition size
Data Type:int
Privilege:Private
Default Value:

**mySysAlloction** - the size of allocation of the partition for system usage
Data Type:int
Privilege:Private
Default Value:

**myUserAllocation** - the size of allocation of the partition for production usage
Data Type:int
Privilege:Private
Default Value:

**Operations:**

All Operations inherited from parent class

**Associations:**

The PlDiskPartition class has associations with the following classes:
Class: PlComputer
PlService (Aggregation)

### 4.3.48 PlEDASModeRecordNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:False
Purpose and Description:
This Class contains information found in a FOS Detailed Activity Schedule, Mode Record.

**Attributes:**

**myDataRate** - The average rate at which data is being stored in the buffer during the mode.
Data Type:EcTInt
Privilege:Private
Default Value:

**myInstrumentName** - Identifies the instrument name with which the mode is associated with.
Data Type:RWCString
Privilege:Private
Default Value:

**myModeName** - Mode name as defined in the DFCD for the EOS AM-1 Project Data Base.

Data Type:RWCStrimg
Privilege:Private
Default Value:

**myModeStartTime -** The specified start time of the mode.
Data Type:RWTime
Privilege:Private
Default Value:

**myModeStopTime -** The specified stop time of the mode.
Data Type:RWTime
Privilege:Private
Default Value:

**myRecordTerminator -** Identified the end of the mode record.
Data Type:RWTime
Privilege:Private
Default Value:

**Operations:**

**PlEDASModeRecord -** Constructor
Arguments:
Return Type:PlEDASModeRecord
Privilege:Public

**~PlEDASModeRecord -** Destructor
Arguments:
Return Type:Void
Privilege:Public

**Associations:**
The PlEDASModeRecordNB class has associations with the following classes:
PlFOSDASFile (Aggregation)

### 4.3.49 PlEDASRecordNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:False
Purpose and Description:
This class contains information found in FOS Detailed Activity Schedule, Activity

Records.

**Attributes:**

**myActivityRecordType** - Indicates the type of scheduling used for the activity (i.e. absolute time or event based)
Data Type:RWCString
Privilege:Private
Default Value:

**myActivityStartTime** - Specifies the Start time of the activity.
Data Type:RWTime
Privilege:Private
Default Value:

**myActivityStopTime** - Specifies the end time of the activity.
Data Type:RWTime
Privilege:Private
Default Value:

**myDataRates** - Specifies the average rate at which data is being stored.
Data Type:EcTInt
Privilege:Private
Default Value:

**myEOCActivityID** - An integer value that uniquely identifies the activity.
Data Type:RWCString
Privilege:Private
Default Value:

**myRecordTerminator** - Identifies the end of an activity record.
Data Type:EcTInt
Privilege:Private
Default Value:

**mySubsystemName** - Identifies the instrument/subsystem name with which the activity is scheduled.
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**PlEDASRecordNB** - Constructor
Arguments:

Return Type:PlEDASRecordNB
Privilege:Public

**~PlEDASRecordNB -** Destructor
Arguments:
Return Type:Void
Privilege:Public

**Associations:**
The PlEDASRecordNB class has associations with the following classes:
PlFOSDASFile (Aggregation)

### 4.3.50 PlEntryScreenNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
An abstraction of a GUI entry screen.

**Attributes:**

None

**Operations:**

**Add**
Arguments:
Return Type:Void
Privilege:Public

**ConfirmCreate**
Arguments:
Return Type:Void
Privilege:Public

**ConfirmDelete**
Arguments:
Return Type:Void
Privilege:Public

**ConfirmModify**
Arguments:
Return Type:Void

Privilege:Public

**Copy**
Arguments:
Return Type:Void
Privilege:Public

**Cut**
Arguments:
Return Type:Void
Privilege:Public

**Delete**
Arguments:
Return Type:Void
Privilege:Public

**Modify**
Arguments:
Return Type:Void
Privilege:Public

**Paste**
Arguments:
Return Type:Void
Privilege:Public

**Select**
Arguments:
Return Type:Void
Privilege:Public

**Associations:**
The PlEntryScreenNB class has associations with the following classes:
PlPlanningWorkbenchUI (Aggregation)

### 4.3.51 PlExportedPlanNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This file is an exported version of a plan. It would contain information such as the start

and stop times, priority and inter-DAAC data dependencies of the jobs in a plan.

**Attributes:**

None

**Operations:**

**Export -** Converts a local representation of a plan into a file that can be stored to the data server.
Arguments:RWTvalSlist<PlActivities>
Return Type:Void
Privilege:Public
PDL: {

```
For plan to be exported
Open output file
  Get list of activities in plan
  For each activity,
    Write name, predicted start & stop time and priority to file
    Identify input and output data granules
    For each data granule
      if data type is needed at or processed at a remote DAAC
        write this data type to file as well, with I for input, O for output
}
```

**Import -** Reads in an exported plan from a remote DAAC and creates a local representation of it.
Arguments:
Return Type:Void
Privilege:Public
PDL: {

```
For file received from data server
Create new plan
Open input file
 Read first name, predicted start & stop time and priority
    While not EOF
        If new Activity
            Assign to new PlActivity::
            PlActivity(name, start, stop, priority)
        If data type,
            create a PlDataDependencies -
         Read next line
}
```

**Associations:**
The PlExportedPlanNB class has associations with the following classes:
PlPublishedPlan (Aggregation)

### 4.3.52 PlFOSDASFile Class

Parent Class:PlDataSchedules
Public:No
Distributed Object:No
Purpose and Description:
This file contains FOS Detailed Activity Schedules received from the Data Server.

**Attributes:**

**myFileName** - Name of the data file for the FOS DAS
Data Type:RWCString
Privilege:Private
Default Value:

**myMetaDataName** - Name of the metadata file for the FOS DAS
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**PlFOSDASFile** - Constructor
Arguments:
Return Type:PlFOSDASFile
Privilege:Public

**~PlFOSDASFile** - Destructor
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlFOSDASFile class has associations with the following classes:
None

### 4.3.53 PlFile Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
Used by the planning algorithm to allocate space to the filesystem for any files used by a DPR in a plan.

**Attributes:**

**myAllocationTime -** Allocation Time of File.
Data Type:
Privilege:Private
Default Value:

**mySize -** Nominal size of PlDataGranule.
Data Type:
Privilege:Private
Default Value:

**Operations:**

None

**Associations:**

The PlFile class has associations with the following classes:
Class: PlDataGranule simulatesstroageof

### 4.3.54 PlGroundActivity Class

Parent Class:PlActivity
Public:No
Distributed Object:No
Purpose and Description:
This class is a generalization of the PlActivity class. The class describes a Ground Event within the plan.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**Cancel** - Cancel the activity
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the cancel method of the Ground Event
}

**Modify** - Modify the activity
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the modify method of the Ground Event
}

**PlGroundActivity** - Constructor method
Arguments:
Return Type:PlGroundActivity
Privilege:Public

**PlGroundActivity** - Create an activity from the Ground Event object
Arguments:PlGroundEvent
Return Type:PlGroundActivity
Privilege:Public

**Schedule** - Schedule the activity
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the schedule method of the Ground Event
}

**Status** - Status the activity
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the status method of the Ground Event
}

**~PlGroundActivity** - Destructor method
Arguments:

Return Type:void
Privilege:Public

**Associations:**

The PlGroundActivity class has associations with the following classes:
Class: PlGroundEvent isaplanned
Class: PlGroundEvent whenplannedis

### 4.3.55 PlGroundEvent Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class describes a Ground Event which is recorded in the PDPS database. A Ground Event marks the allocation of resources to some none-production task such as maintainance.

**Attributes:**

**myDescription -** This attribute provides storage for a text description of a ground event
Data Type:String
Privilege:Private
Default Value:
This is a Constant.

**myDuration -** This attribute describes the duration of the Ground Event
Data Type:Time
Privilege:Private
Default Value:
This is a Constant.

**myName -** Event name entered by Resource Manager for this ground event.
Data Type:String
Privilege:Private
Default Value:

**myPriority -** This attribute describes the priority of the Ground Event.
Data Type:int
Privilege:Private
Default Value:
This is a Constant.

**myTemplateFlag -** This attribute describes whether the ground event is to be saved as a template description of a ground events. Ground Events with this attribute True will be retained in the PDPS database. Those with this attribute as False will automatically be deleted a month after the completion date of the Ground Event.
Data Type:Boolean
Privilege:Private
Default Value:False
This is a Constant.
Contraints:
Non Persisent Flag:

**myWinEndTime -** This attribute describes the end time of the window of opportunity for when the Ground Event may be planned
Data Type:Time
Privilege:Private
Default Value:
This is a Constant.
Contraints:
Non Persisent Flag:

**myWinStartTime -** This attribute describes the start time of the window of opportunity for when the Ground Event may be planned
Data Type:Time
Privilege:Private
Default Value:
This is a Constant.
Contraints:
Non Persisent Flag:

**Operations:**

**Cancel -** Cancel the Ground Event within the data processing job scheduler
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the cancel Ground Event method of the DpPrJobScheduler
}

**Create -** Create an ground event in the PDPS database
Arguments:
Return Type:void
Privilege:Public

**Delete** - Delete the ground event from the PDPS database
Arguments:
Return Type:void
Privilege:Public

**Modify** - Modify the Ground Event within the data processing job scheduler. The attributes of the Ground Event that may have been modified are priority and the anticipated start / stop times of the DPR used to set the alarms in the data processing scheduler.
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the modify Ground Event job method of the DpPrJobScheduler
}

**PlGroundEvent** - Constructor method
Arguments:
Return Type:PlGroundEvent
Privilege:Public

**Schedule** - Schedule the Ground Event within the data processing job scheduler
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the create Ground Event job method of the DpPrJobScheduler
}

**Status** - Status the Ground Event within the data processing job scheduler
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the status Ground Event job method of the DpPrJobScheduler
}

**~PlGroundEvent** - Destructor method
Arguments:
Return Type:void
Privilege:Public

**Associations:**

The PlGroundEvent class has associations with the following classes:
Class: PlResourceRequirement hasa

Class: PlGroundActivity isaplanned
Class: DpPrScheduler isscheduledinprocessingby
Class: PlGroundActivity whenplannedis
PlGroundEvents (Aggregation)

### 4.3.56 PlGroundEventAllocation Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
A link class that determines whether or not this ground event needs exclusive use of this resource

**Attributes:**

**myExclusiveOrShared** - Determines whether this ground event requires the exclusive use of a resource, or if it can share the resource with other activities.
Data Type:enum
Privilege:Private
Default Value:

**Operations:**

None

**Associations:**

The PlGroundEventAllocation class has associations with the following classes:
None

### 4.3.57 PlGroundEventExecutable Class

Parent Class:PlGroundEvent
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
Any ground events that can executed via a UNIX script will be automatically handled through resource planning using this object

**Attributes:**

**myActualStartTime** - This is the actual time that this Ground Event's script began executing.
Data Type:RWTime
Privilege:Private
Default Value:

**myActualStopTime** - This is the actual time that this Ground Event's script finished executing.
Data Type:RWTime
Privilege:Private
Default Value:

**myScriptName** - This is the actual name of the script that executes this ground event, complete with the name of the directory on the host machine where this script can be found
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**Schedule** - This operation overrides the parent "Schedule" operation and allows an actual script to be scheduled to run, instead of just creating a place holder in the Production Queue
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlGroundEventExecutable class has associations with the following classes:
None

### 4.3.58 PlGroundEvents Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This is a collector class for the PlGroundEvent class, and contains methods to select Ground Events from the PDPS database and to iterate through them. This class may be implemented by a suitable Rogue Wave template class

**Attributes:**

None

**Operations:**

**Add -** Add a Ground Event to the collection
Arguments:PlGroundEvent: event
Return Type:void
Privilege:Public

**Delete -** Delete a Ground Event from the collection
Arguments:PlGroundEvent: event
Return Type:void
Privilege:Public

**First -** Returns the first Ground Event within the collection.
Arguments:
Return Type:PlGroundEvent
Privilege:Public

**Next -** Returns the next Ground Event within the collection (or NULL if no more activities).
Arguments:
Return Type:PlGroundEvent
Privilege:Public

**SelectEvents -** Builds the collection for all Ground Events within the PDPS database that lie within the time interval specified.
Arguments:Interval
Return Type:void
Privilege:Public

**Associations:**

The PlGroundEvents class has associations with the following classes:
None

### 4.3.59 PlImportedActivity Class

Parent Class:PlActivity
Public:No
Distributed Object:No
Purpose and Description:
This object is a specialization of the class PlActivity and represents activities that have been

derived from an imported plan.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

All Operations inherited from parent class

**Associations:**

The PlImportedActivity class has associations with the following classes:
Class: PlDataDependencies mayhave

### 4.3.60 PlInstModeRecords Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class represents the various modes related to each instrument during the period of the Instrument Mode Schedule.

**Attributes:**

**myInstrumentName -** This attribute specifies the instruments name.
Data Type:RWCStrimg
Privilege:Private
Default Value:

**myModeName -** Mode name, derived from the PlFOSDASFile as defined in the DFCD for the EOS AM-1 Project Data Base.
Data Type:RWCStrimg
Privilege:Private
Default Value:

**myModeStartTime -** The specified start time of the mode.
Data Type:RWTime
Privilege:Private
Default Value:

**myModeStopTime -** The specified end time of the mode.

Data Type:RWTime
Privilege:Private
Default Value:

**Operations:**

**UpdatInstModeStatus -** this operation updates the mode status for an instrument based on scheduling information received.
Arguments:
Return Type:EcUtStatus
Privilege:Public
PDL:{
Assign values based on mode information receive from FOS DAS.
PlInstModeRecords:myModeName
PlInstModeRecords:myModeStartTime
PlInstModeRecords:myModeStopTime
}

**Associations:**

The PlInstModeRecords class has associations with the following classes:
Class: PlInstrumentModes

### 4.3.61 PlInstrumentModes Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This object represents the PDPS Instrument Mode Schedule based on FOS Detailed Activity Schedule, mode information.

**Attributes:**

**myScheduleStart -** The start time of the FOS Detailed Activity Schedule that was used to create the Instrument Mode Schedule.
Data Type:RWTime
Privilege:Private
Default Value:

**myScheduleStop -** The end time of the FOS Detailed Activity Schedule that was used to create the Instrument Mode Schedule.
Data Type:RWTime

Privilege:Private
Default Value:

**Operations:**

**FindAssociatedInstrument -** This operation will find the instrument associated with a particular mode in the instrument mode schedule.
Arguments:RWCString:myInstrumentName
Return Type:EcTBoolean
Privilege:Public
PDL:{

```
 Parse through list of PlInstModeRecords to find associated instrument
 Get PlInstModeRecord:myInstrumentName
if ( found)
{
   return Instrument found flag of false
}
else
{
   call PlInstModeRecords:UpdateInstModeStatus
   to update mode information in list
}
}
```

**MakeModeScheduleFromDASoper5ation**
Arguments:RWTValSlist<PlFOSDASFile>
Return Type:EcUtStatus
Privilege:Public

**RetrieveModeData -** This operation will get the mode information from for a particular instrument.
Arguments:RWTValSlist<PlInstModeRecord>
Return Type:Void
Privilege:Public

**Associations:**

The PlInstrumentModes class has associations with the following classes:
Class: PlInstModeRecords
Class: PlPGECollection
Class: PlPGEProfile
Class: PlDataSchedules isusedtoconstructinstrumentmodeschedule

### 4.3.62 PlMetaDataChecks Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class provides a metadata field and its corresponding value to be checked against the actual metadata of the specified input data granule when deciding if a particular input should be used, or a PGE should be executed.

**Attributes:**

**myDataTypeId** - This is the identifier of the data type for which the defined metadata values should be checked.
Data Type:RWCString
Privilege:Private
Default Value:

**myMetaDataOper** - This is the operation (=, >, <) for the comparision of the metadata field.
Data Type:RWCString
Privilege:Private
Default Value:

**myMetaDataParmName** - This is the name of the metadata parameter to be compared.
Data Type:RWCString
Privilege:Private
Default Value:

**myMetaDataType** - This is the type (int, string, etc...) of the metadata parameter to be compared.
Data Type:RWCString
Privilege:Private
Default Value:

**myMetaDataValue** - This is the value for the metadata parameter to be compared with the actual metadata value.
Data Type:RWCString
Privilege:Private
Default Value:

**myPgeId** - This is the identifier of the PGE for which the metadata values are to be checked.
Data Type:RWCString

Privilege:Private
Default Value:

**Operations:**

**CheckForCondition -** This method checks the metadata for the specified Data Granule for the metadata field and value.
Arguments:RWCString, PlDataGranule &, EcTBoolean &
Return Type:Void
Privilege:Public

**CompareActualWithCondition -** This method compares the String metadata attribute value with the actual file's metadata.
Arguments:RWCString &, RWCString &, RWCString &, EcUtStatus &
Return Type:Void
Privilege:Public

**CompareActualWithCondition -** This method compares the Int metadata attribute value with the actual file's metadata.
Arguments:EcTInt, EcTInt, RWCString &, EcUtStatus &
Return Type:Void
Privilege:Public

**CompareActualWithCondition -** This method compares the Float metadata attribute value with the value of the actual file's metadata.
Arguments:EcTFloat, EcTFloat, RWCString &, EcUtStatus &
Return Type:Void
Privilege:Public

**CompareActualWithCondition -** This method compares the String metadata attribute value with the actual file's metadata.
Arguments:RWCString &, RWCString &, RWCString &, EcUtStatus &
Return Type:Void
Privilege:Public

**CompareActualWithCondition -** This method compares the Int metadata attribute value with the actual file's metadata.
Arguments:EcTInt, EcTInt, RWCString &, EcUtStatus &
Return Type:Void
Privilege:Public

**Associations:**

The PlMetaDataChecks class has associations with the following classes:
Class: PlDataTypeReq

Class: PlDataTypeReq performs

### 4.3.63 PlNetwork Class

Parent Class:PlResource
Public:No
Distributed Object:No
Purpose and Description:
Simple specialization of the PlResource class to describe a Network object against which a Ground Event may be allocated.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

All Operations inherited from parent class

**Associations:**

The PlNetwork class has associations with the following classes:
PlService (Aggregation)

### 4.3.64 PlOnDemandExceed Class

Parent Class:PlReplanCriteria
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
Checks each On-demand Production Request against a number of limit automatic replan resource thresholds to determine if the operator should be notified that a replan should be considered.

**Attributes:**

**myDefaultToThresh** - Indicates whether the Limited Automatic Replan should be triggered at the same levels as the On Demand Thresholds set up for deferring an OPR. This way every time an OPR is deferred, the operator will be asked if a replan should be done.
Data Type:EcTBoolean
Privilege:Private

Default Value:False

**Operations:**

**AddToReport -** Creates a string that can be added to myReport that identifies what resource threshold was exceeded, what the current limit is, and what the OPR uses.
Arguments:EcTFloat, EcTFloat, EcTInt
Return Type:RWCString
Privilege:Public
PDL: {
For On Demand Production Request resource value that exceeded threshold
 Create a string that contains type of threshold exceeded,
 current threshold, and OPRs threshold amount
 Return this string
}

**AddToReport**
Arguments:EcTFloat, EcTFloat

**CheckReplan -** This operation checks if an On-demand Production Request exceeds the resource thresholds set for limit automatic replans. If any do, then the operator is notified that a replan should be considered.
Arguments:PlOnDemandPRNB &
Return Type:EcTBoolean
Privilege:Public
PDL: {

```
For each resource type
  Call OnDemandReplanValues::CompareThreshod(OPR &)
  if fails -
    if first failure, put general information about OPR in myReport
    call OnDemandExceed::AddToReport
          (OPR::CalculateResourceUsage(resource type),
               OnDemandReplanValues::myReplanValue)
end For
If any failures, call PlOnDemandExceed::NotifyOperator
}
```

**Associations:**

The PlOnDemandExceed class has associations with the following classes:
Class: PlOnDemandReplanValues

### 4.3.65 PlOnDemandManagerNB Class

Parent Class:EcPfManagedServer
Public:No
Distributed Object:No
Purpose and Description:
This class is the manager for all On-Demand production requests providing status, modification, cancellation and housekeeping services.

**Attributes:**

**myCumRescUse** - This attribute contains the cumulative usage of a resouce for On-Demand Productions by the resource type for a Production period.
Data Type:RWTime
Privilege:Private
Default Value:

**myOnDemandPRs** - List of On Demand Production Request currently active.
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**Operations:**

**AddOPR** - This operation adds the current request to the list of requests maintained by the manager.
Arguments:PlOnDemandPRNB &
Return Type:void
Privilege:Public

**Cancel** - Operation to cancel the execution of the production associated with the request identified by the parameter passed to the operation.
Arguments:OPRID

**Init** - Initializes the application by creating the lifecycle service callbacks (shutdown, etc.) and setting up the communications for receiving subscriptions.
Arguments:
Return Type:void
Privilege:Public

**Modify** - Operation which will provide the capability to modify a production request based on the parameters passed to the operation which identify the request and provide the new requirements for the production request.
Arguments:OPRID,OPR

**PlOnDemandManagerNB -** Constructor method
Arguments:
Return Type:PlOnDemandManagerNB
Privilege:Public

**RemoveOPR -** This operation is called when the notification is received that the product has been generated and this operation then removes the production request from the list of requests the manager maintains.
Arguments:PlOnDemandPRNB &
Return Type:void
Privilege:Public

**Shutdown -** Performs a graceful shutdown of the application by deinstalling receivers and closing communications.
Arguments:
Return Type:void
Privilege:Public

**Status -** Operation which when called will provide the current status of the production request associated with the parameter passed to the operation.
Arguments:OPRID

**SubmitSubscription -** This operation is called by ReturnUR in PlOnDemandPRNB to create and submit a subscription to be notified of the insertion into the Data Server of an instance of the product generated by the OnDemand request.
Arguments:
Return Type:void
Privilege:Public

**UpdateRescUse -** This operation adds the resource usage time to the attribute in the manager that is tracking the total resource usage for On Demand requests.
Arguments:rescTime
Return Type:void
Privilege:Public

**~PlOnDemandManager -** Destructor method
Arguments:
Return Type:void
Privilege:Public

**Associations:**
The PlOnDemandManagerNB class has associations with the following classes:
Class: EcEvent
Class: EcMpMsgCb
Class: PlOnDemandPRNB aremanagedby

Class: PlSubscriptionSubmitIF submitssubscriptionsusing

### 4.3.66 PlOnDemandPRNB Class

Parent Class:PlProductionRequestB
Public:Yes
Distributed Object:Yes
Purpose and Description:
This class is the specialization class which holds additional/modified attributes and operations needed for On-Demand production requests.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**PlOnDemandPRNB** - Constructor method
Arguments:
Return Type:PlOnDemandPRNB
Privilege:Public

**RequestCompleted** - Operation to accept UR of a completed product and associate with a given Production Request, then remove it from the list of requests kept by the Manager and return the UR to Data Server via the Callback routine.
Arguments:UR
Return Type:void
Privilege:Public

**ReturnUR** - This operation is called by the Data Server to initiate action to retrieve the UR for the product being generated by this request. It calls the SubmitSubscription routine in PlOnDemandManagerNB, and then after the subscription is returned and the cleanup is accomplished it returns the UR of the product.
Arguments:
Return Type:EcUrUR
Privilege:Public

**ValidateOPR** - Operation to determine if thresholds are exceeded and if data is available to generate product. If conditions are satisfactory then request is submitted to processing.
Arguments:
Return Type:String
Privilege:Public
PDL:
PlProductionRequestB : PlDefinePgeRuns();

PlProductionRequestB : PlStoreProductionRequest(); // Call to PlProductionRequestB operation to store the information pertaining to this request in the PDPS database.

PlOnDemandManagerNB : AddOPR(PlOnDemandPRNB &); // Call to PlOnDemand ManagerNB operation to add current PR to list of PRs kept in myOnDemandPRs attribute.

PlDPRB : Schedule(); // Calls DpPrScheduler operation to create the DPR info in the PDPS Database.

PlRescUseThreshNB(); // Constructor to create instance of PlRescUseThreshNB and populate the attributes with information from the DB.

if (PlRescUseThreshNB : CheckThreshold(PlOnDemandPRNB &) // Compare resource requirements from PGE Profile against authorized useage limits for On Demand Production.

then

PlOnDemandManagerNB : UpdateRescUse(rescTime); //Call to PlOnDemandManagerNB operation to add current production request resource usage to myCumRescUse attribute to maintain a running total.

if (PlDPRB : CheckAvailability()) // Call to PlDPRB operation to check if the necessary data is available at the Data Server.

then

PlDPRB : Release(); // Calls DpPrScheduler operation to release the job to the Autosys Job Box to start production.

status = "Submitted";

else

PlSubscriptionSubmitIF : SubmitSubscription(); // Call to PlSubscriptionSubmitIF operation to create a subscription to be notified upon arrival at Data Server of data.

status = "Delayed";

else

status = "Exceeded"; // Resource thresholds exceeded requires operator intervention.

PlOnDemandExceed : CheckReplan(PlOnDemandPRNB &); Check to see if a replan notification is necessary.

return(status); // Return the text string indicating current status of the production request, EcRequest object would provide additional information.

**~PlOnDemandPRNB** - Destructor method.
Arguments:
Return Type:void
Privilege:Public

**Associations:**
The PlOnDemandPRNB class has associations with the following classes:
Class: EcRequest ProvidesStatus
Class: PlOnDemandManagerNB aremanagedby
Class: PlReplanCriteria checkforneedforreplannotification
Class: PlRescUseThreshNB comparewith
Class: PlDPRB composes
Class: PlPGE requests

### 4.3.67 PlOnDemandReplanValues Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
The resource usage thresholds for replan notification - one per resource type (CPU, disk space, etc)

**Attributes:**

**myCumulative** - If true, indicates that this is a culmutive threshold - that is, give a replan notification if the summation of OPRs received during this time period exceed this threshold
Data Type:EcTBoolean
Privilege:Private
Default Value:

**myReplanValue** - The threshold for this resource above which causes a replan notification
Data Type:EcTInt
Privilege:Private
Default Value:

**myResourceType** - Indicates which resource type this value should be assocation with.
Data Type:enum
Privilege:Private
Default Value:

**Operations:**

**CompareThreshold -** Compares the resource usage of a Production Request with the threshold set for replan notification for this resource type.
Arguments:PlOnDemandPRNB &
Return Type:EcTBoolean
Privilege:Public
PDL: {
if OPR::CalculateResourceUsage(resource type) > myReplan Value,
return FAILURE
}

**Associations:**

The PlOnDemandReplanValues class has associations with the following classes:
Class: PlOnDemandExceed

### 4.3.68 PlOrbitModelNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is a table used by PlOrbitScheduled for calculating the estimated times for an orbit.

**Attributes:**

**myOrbitNum -** This attribute is the orbit number.
Data Type:EctInt
Privilege:Private
Default Value:

**myPlatform -** This attribute is the satelite which will be providing the data.
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**ConvertOrbitNumtoTime -** Converts an orbit number into its corresponding time.
Arguments:EcTInt:OrbitNum,RWTime:OrbitTime

Return Type:Void
Privilege:Public

**PlOrbitModelNB -** This is the default constructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**PlOrbitModelNB -** This constructor for the class takes in the Platform and Orbitnum, and places the values into the appropriate attributes when an object is created.
Arguments:Platform:RWCString, OrbitNum:EcTInt
Return Type:Void
Privilege:Public

**~PlOrbitModelNB -** This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**
The PlOrbitModelNB class has associations with the following classes:
Class: PlOrbitScheduledNB references

### 4.3.69 PlOrbitScheduledNB Class

Parent Class:PlPGE
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is a table used by the PGEs scheduled for the orbit to generate the DPRs and their input data.

**Attributes:**

**myEphemeris -** This attribute is the ephemeris data files that are used in determining orbit start and stop times and to syncronize orbits, if necessary.
Data Type:RWTValList<RWCString>
Privilege:Private
Default Value:

**myOrbitLength -** This attribute is the duration of an orbit.
Data Type:RWTime
Privilege:Private

Default Value:

**Operations:**

**GenerateDPRs -** This method generates production requests based on the attributes of this class and the associated OrbitModel.
Arguments:PlProductionRequest
Return Type:Void
Privilege:Public

**PlOrbitScheduledNB -** Constructor
Arguments:
Return Type:Void
Privilege:Public

**~PlOrbitScheduledNB -** Destructor
Arguments:
Return Type:Void
Privilege:Public

**Associations:**
The PlOrbitScheduledNB class has associations with the following classes:
Class: PlOrbitModelNB references

### 4.3.70 PlOtherTypes Class

Parent Class:PlPGE
Public:No
Distributed Object:No
Purpose and Description:
This is a place holder class allowing for other types of PGE scheduling.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

All Operations inherited from parent class

**Associations:**

The PlOtherTypes class has associations with the following classes:
None

### 4.3.71 PlOutputYield Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
Specifies the recipe to describe the output data granules for a PGE.

**Attributes:**

**myCommandString -** This string is used to construct the insert command for the data type, used to destage the data to the data server.
Data Type:RWCString
Privilege:Private
Default Value:

**myDataTypeId -** This is the identifier for the Data Type for which this is the output yield.
Data Type:RWCString
Privilege:Private
Default Value:

**myLogicalID -** The logical id relates to the PGE output identifier for a particular product type. This attribute is required within the SDP toolkit process control interface.
Data Type:EcTInt
Privilege:Private
Default Value:

**myPgeId -** This is the identifier of the PGE that this Output Yield is for.
Data Type:RWCString
Privilege:Private
Default Value:

**myYield -** Describes the number of data granules produced from the Data Processing Request. These are assumed to be evenly distributed in time accross the acquisition processing time of the DPR.
Data Type:EcTFloat
Privilege:Private
Default Value:

**Operations:**

**PlOutputYield** - This is the default constructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**PlOutputYield** - This constructor takes in the PGE ID and Data Type ID for the created output yield object.
Arguments:RWCString:PgeId, RWCString:DataTypeId
Return Type:Void
Privilege:Public

**PlOutputYield** - This is the default constructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**PlOutputYield** - This constructor takes in the PGE ID and Data Type ID for the created output yield object.
Arguments:RWCString:PgeId, RWCString:DataTypeId
Return Type:Void
Privilege:Public

**~PlOutputYield** - This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlOutputYield class has associations with the following classes:
None

### 4.3.72 PlPDASFile Class

Parent Class:PlDataSchedules
Public:No
Distributed Object:No
Purpose and Description:
This object is a representation of a file containing data availability times from remote DAACs that is stored in the data server

**Attributes:**

**myFileName -** Name of the PDAS file
Data Type:RWCString
Privilege:Private
Default Value:

**myMetaDataFileName -** Name of the metadata file for the PDAS
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**InsertToDSS -** This operation inserts the PDPS Data Availability Schedule file to the Data Server.
Arguments:
Return Type:EcTStatus
Privilege:Public
PDL: {
Build parameter list for insert (this should be fairly constant for plans)
submit request for insert.
}

**PlPDASFile**
Arguments:
Return Type:PlPDASFile
Privilege:Public

**Associations:**

The PlPDASFile class has associations with the following classes:
Class: GlParameterList holdsparametersforINSERT
Class: PlActivePlan iscreatedfrom
Class: GlCallback receivescallbackfromINSERT
Class: DsClCommand usedforINSERT
Class: DsClESDTReferenceCollector usedforINSERT
Class: DsClRequest usedforINSERT

### 4.3.73 PlPDASMetaData Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:

The metadata stored with a PDPS Data Availability Schedule on the Data Server

**Attributes:**

**myDAAC -** The name of the DAAC that is produced this PDAS
Data Type:RWCString
Privilege:Private
Default Value:

**myNumRecords -** Number of data availability records in this PDPS Data Availability Schedule
Data Type:EcTInt
Privilege:Private
Default Value:

**myStartTime -** The start of the timeframe for this PDAS
Data Type:RWTime
Privilege:Private
Default Value:

**myStopTime -** The end of the timeframe for this PDAS
Data Type:RWTime
Privilege:Private
Default Value:

**Operations:**

**PlPDASMeteData -** Constructor
Arguments:
Return Type:PlPDASMeteData
Privilege:Public

**~PlPDASMeteData -** Destructor
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlPDASMetaData class has associations with the following classes:
PlPDASFile (Aggregation)

### 4.3.74 PlPDASRecords Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
Contains the information about when data will be produced.

**Attributes:**

**myDataTypeId** - Matches myDataTypeId on PlDataType and PlDataGranule
Data Type:RWCString
Privilege:Private
Default Value:

**myStartTime** - Matches myStartTime in PlDataGranule - the start time of the data.
Data Type:RWTime
Privilege:Private
Default Value:

**myStopTime** - Matches myStopTime in PlDataGranule - the stop time of the data.
Data Type:RWTime
Privilege:Private
Default Value:

**myTileId**
Data Type:RWCString
Privilege:Private
Default Value:

**myTimeCompleted** - This is the time the data will finished being produced at the DAAC.
Data Type:RWTime
Privilege:Private
Default Value:

**Operations:**

**PlPDASRecords** - Constructor
Arguments:
Return Type:PlPDASRecords
Privilege:Public

**~PlPDASRecords** - Destructor
Arguments:
Return Type:Void

Privilege:Public

**Associations:**

The PlPDASRecords class has associations with the following classes:
PlPDASFile (Aggregation)

### 4.3.75 PlPGE Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This is the base class within a generalization heirachy that describes PGEs. The class defines abstract operations required for the planning subsystem to work out when a PGE needs to be scheduled. As well as containing the key attributes defining the PGE.

**Attributes:**

**myDefaultTimer** - This is the default timer value for the PGE for use in Alternative Input selection.

**myInputDataTypeList** - List of input data types needed by the PGE
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**myInstrument** - Instrument for which the PGE is appropriate
Data Type:RWCString
Privilege:Private
Default Value:

**myNumCPUs** - This is the number of CPUs that the PGE requires. It should be 1 for non-parallel processing PGEs.
Data Type:EcTInt
Privilege:Private
Default Value:

**myOutputDataTypeList** - List of output data types needed by the PGE
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**myPGEName -** Name of the PGE
Data Type:RWCString
Privilege:Private
Default Value:

**myPGEVersion -** Version number of the PGE
Data Type:RWCString
Privilege:Private
Default Value:

**myPgeId -** This is the unique identifier of the PGE.
Data Type:RWCString
Privilege:Private
Default Value:

**myPlatform -** Platform for which the PGE is appropriate, may be a list
Data Type:RWCString
Privilege:Private
Default Value:

**myTestOperational -** Indicates if the PGE's status is test or operational
Data Type:enum
Privilege:Private
Default Value:

**myType -** This is the type of the PGE.
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**Delete -** Delete the PGE from the PDPS database
Arguments:
Return Type:void
Privilege:Public

**FindDataAvailability -** This is used to find what input data is available for this PGE.
Arguments:Interval
Return Type:void
Privilege:Public
PDL:{
// Iterates over the input data type list invokes the
// Find Data Availability method for each input type
}

**GenerateDPRs -** Generate the Data Processing Requests to fulfill the Production Request.
Arguments:PlProductionRequest
Return Type:void
Privilege:Public
This is an abstract operation
PDL://

```
//// do not generate DPR if metadata checks have failed for all inputs
// if (number of metadata check failures > PlDataTypeReq.myNumNeeded) {
//   set PlDPR.myCompletionState to "failed"
//   return
// }
//
// determine time period unit (e.g. hour, month, orbit set)
//
// while scheduleTime < stopTime
//
//   handle intermittent execution activations
//
//   if (PlProductionRequest.myNumDPRsToKeep > 0) {
//     // construct DPR if this occurrence is to be kept
//     if (keepCount < myNumDPRsToKeep) {
//       skipCount = 0
//       PlDPR // constructor
//       keepCount++
//     }
//     else if (skipCount < myNumDPRsToSkip) {
//       keepCount = 0
//       if (skipCount < myNumDPRsToSkip) {
//         skipCount++
//         break
//       }
//     }
//   }
//   else // DPR is always generated each time period
//     PlDPR // constructor
//   }
//
//   get all input and outpout data granules and insert into PDPS DB
//
//   bump scheduleTime to beginning of next time period
//
// } // end while
```

**Modify -** This is used to modify the attributes of this class.

Arguments:TestOrOper:enum
Return Type:void
Privilege:Public

**PlPGE** - Constructor method
Arguments:
Return Type:PlPGE
Privilege:Public

**PlPGE** - Construct the PGE from the PDPS database
Arguments:int PGEid

**PlPGE** - Constructor method
Arguments:
Return Type:PlPGE
Privilege:Public

**PlPGE** - This constructor takes the PGE ID of the object to be constructed.
Arguments:RWCString:PGEid
Return Type:PlPGE
Privilege:Public

**PlPGE** - Constructor method
Arguments:
Return Type:PlPGE
Privilege:Public

**PlPGE** - This constructor takes the PGE ID of the object to be constructed.
Arguments:RWCString:PGEid
Return Type:PlPGE
Privilege:Public

**UpdateVersion** - Update the version of the PGE
Arguments:
Return Type:void
Privilege:Public

**~PlPGE** - Destructor method
Arguments:
Return Type:void
Privilege:Public

**Associations:**

The PlPGE class has associations with the following classes:

Class: PlDPRB
Class: PlPgeFactory createdby
Class: PlPgeFactory creates
Class: PlDPRB generates
Class: PlDataTypeB input
Class: PlResourceRequirement isreferencedby
Class: PlUserParameters isreferencedby
Class: PlDataTypeB output
Class: PlOnDemandPRNB requests
Class: PlProductionRequestB requests
PlPGEProfile (Aggregation)

### 4.3.76 PlPGEActivity Class

Parent Class:PlActivity
Public:No
Distributed Object:No
Purpose and Description:
This class is a generalization of the PlActivity class. The class describes a Data Processing Request - a run of a PGE - within the plan.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**Cancel** - Cancel the activity
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the cancel method of the Data Processing Request
}

**Modify** - Modify the activity
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the modify method of the Data Processing Request
}

**PlPGEActivity** - Constructor method
Arguments:
Return Type:PlPGEActivity
Privilege:Public

**PlPGEActivity** - Constructs an activity within the plan for the Data Processing Request
Arguments:PlDPR
Return Type:PlPGEActivity
Privilege:Public

**Schedule** - Schedule the activity
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the schedule method of the Data Processing Request
}

**Status** - Modify the activity
Arguments:
Return Type:void
Privilege:Public
PDL:{
// calls the modify method of the Data Processing Request
}

**~PlPGEActivity** - Destructor method
Arguments:
Return Type:void
Privilege:Public

**Associations:**

The PlPGEActivity class has associations with the following classes:
Class: PlProdStratNB determinespriorityof
Class: PlDPRB isaplanned

### 4.3.77 PlPGECollection Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True

Purpose and Description:

**Attributes:**

None

**Operations:**

None

**Associations:**

The PlPGECollection class has associations with the following classes:
Class: PlInstrumentModes

### 4.3.78 PlPGEPriorityNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is part of the production strategies which relates a particular PGE to a particular priority. It can be used to determine the priority of jobs useing that PGE.

**Attributes:**

**myPGE -** This attribute is part of the production strategy that relates a PGE to a particular priority.
Data Type:RWCString
Privilege:Private
Default Value:

**myPriority -** This attribute is part of the production strategy that relates a priority to a particular PGE.
Data Type:EcTInt
Privilege:Private
Default Value:

**Operations:**

**GetPriority -** This retrieves the priority associated with a PGE for a DPR.
Arguments:PlDPR &
Return Type:Void

Privilege:Public
PDL: {
Search database table for PRType = PlDPR::GetPR::GetPRType()
return PRIORITY from table associated with that PR Type
if no priority in table for this PR type, return default priority
}

**Associations:**
The PlPGEPriorityNB class has associations with the following classes:
PlProdStratNB (Aggregation)

### 4.3.79 PlPGEProfile Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Purpose and Description:
This class describes the collection of information that describes a PGE to the Planning subsystem.

**Attributes:**

None

**Operations:**

**$PlPGEProfile -** Constructor function
Arguments:...
Return Type:void
Privilege:Public

**DeletePGEProfile -** Deletes a given PGE profile
Arguments:PGEProfID:int
Return Type:void
Privilege:Public

**ModifyPGEProfile -** Modifies a given PGE profile
Arguments:PGEProfId:int, ModType:enum, ModField:String, ModValue:String
Return Type:void
Privilege:Public

**RetrievePGEProfile -** Retrieves a given PGE profile
Arguments:PlPGEProfID
Return Type:void

Privilege:Public

**~PlPGEProfile -** Destructor function
Arguments:...
Return Type:void
Privilege:Public

**Associations:**

The PlPGEProfile class has associations with the following classes:
Class: PlInstrumentModes

### 4.3.80 PlPRCollectionNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is a list of the production requests either in processing or awaiting processing.

**Attributes:**

**myCollectionID -** Uniquely identifies this production request collection.
Data Type:RWCString
Privilege:Private
Default Value:

**myDataCollectionStartTime -** The start date/time of the collection.
Data Type:RWTime
Privilege:Private
Default Value:

**myDataCollectionStopTime -** The stop date/time of the collection.
Data Type:RWTime
Privilege:Private
Default Value:

**myOriginationTime -** The time the collection request was entered.
Data Type:RWTime
Privilege:Private
Default Value:

**myPriority -** The priority of the collection.

Data Type:EcTInt
Privilege:Private
Default Value:

**myProductionRequestList** - The list of production requests associated with this collection.
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**myProductionRequests** - This attribute is an instance of a production request.

**myRequesterID** - The ID of the requester entering the collection.
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**Add** - Adds a production request to the collection.
Arguments:

**Add** - Adds a production request to the collection.
Arguments:RWCString
Return Type:Void
Privilege:Public

**Delete** - Deletes a production request from the collection.
Arguments:

**Delete** - Deletes a production request from a collection.
Arguments:RWCString
Return Type:Void
Privilege:Public

**First** - Retrieves the first production request in a collection.
Arguments:
Return Type:RWCString
Privilege:Public

**Next** - Retrieves the next production request in the collection.
Arguments:
Return Type:RWCString
Privilege:Public

**Associations:**
The PlPRCollectionNB class has associations with the following classes:
Class: PlProductionRequestUI displays

### 4.3.81 PlPRPriorityNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class describes a table in the PDPS database that matches the types of production requests (on-demand, standard and reprocessing) and the priority assigned to requests of that type.

**Attributes:**

**myPriority -** This attribute is part of the production strategy that relates a priority to a particular production request type.
Data Type:EcTInt
Privilege:Private
Default Value:

**myType -** This attribute is the type of production request (example: standard, reprocessing, on-demand)
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**GetPriority -** This retrieves the priority associated with a PGE for a DPR.
Arguments:PlDPR &
Return Type:EcTInt
Privilege:Public
PDL: {
Search database table for PRType = PlDPR::GetPR::GetPRType()
return PRIORITY from table associated with that PR Type
if no priority in table for this PR type, return default priority
}

**Associations:**
The PlPRPriorityNB class has associations with the following classes:

PlProdStratNB (Aggregation)

### 4.3.82 PlPerformance Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class describes the performance statistcs of a PGE. These performance statistics are established at AI&T. The class also contains attributes to describe the statistics updated from the Data Processing subsystem.

**Attributes:**

**myElapsedTime -** Elapsed time for the PGE during AI&T
Data Type:RWTime
Privilege:Private
Default Value:

**myMaxMemoryUse -** Maximum memory required by the PGE during AI&T
Data Type:EcTFloat
Privilege:Private
Default Value:

**myNoOfBlockInOper -** Number of input blocks for the PGE during AI&T
Data Type:EcTInt
Privilege:Private
Default Value:

**myNoOfBlockOutOper -** Number of output blocks for the PGE during AI&T
Data Type:EcTInt
Privilege:Private
Default Value:

**myNoOfPageFaults -** Number of PGE page faults for the PGE during AI&T
Data Type:EcTInt
Privilege:Private
Default Value:

**myNoOfSwaps -** Number of PGE swaps for the PGE during AI&T
Data Type:EcTInt
Privilege:Private
Default Value:

**myPGECPUTime -** CPU required for the PGE during AI&T
Data Type:RWTime
Privilege:Private
Default Value:

**myPgeId -** This is the unique identifier of the PGE.
Data Type:RWCString
Privilege:Private
Default Value:

**myRunElapsedTime -** Elapsed time for the PGE during production
Data Type:RWTime
Privilege:Private
Default Value:

**myRunMaxMemoryUse -** Maximum memory required by the PGE during production
Data Type:EcTFloat
Privilege:Private
Default Value:

**myRunNoOfBlockInOper -** Number of input blocks for the PGE during production
Data Type:EcTInt
Privilege:Private
Default Value:

**myRunNoOfBlockOutOper -** The number of output blocks used by the PGE during production.
Data Type:EcTInt
Privilege:Private
Default Value:

**myRunNoOfPageFaults -** Number of PGE page faults for the PGE during production
Data Type:EcTInt
Privilege:Private
Default Value:

**myRunNoOfSwaps -** Number of PGE swaps for the PGE during production
Data Type:EcTInt
Privilege:Private
Default Value:

**myRunPGUCPUTime -** CPU required for the PGE during production
Data Type:RWTime
Privilege:Private

Default Value:

**myRunSharedMemoryUse -** Shared memory required by the PGE during production
Data Type:EcTFloat
Privilege:Private
Default Value:

**mySharedMemoryUse -** Shared memory required by the PGE during AI&T
Data Type:EcTFloat
Privilege:Private
Default Value:

**Operations:**

**PlPerformance -** This is the default constructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**PlPerformance -** This is the constructor for the class that takes Pgeid as an argument.
Arguments:RWCString:PgeId
Return Type:Void
Privilege:Public

**UpdateRunTimePerfPar -** This routine updates the runtime performance parameters. It is called after a PGE has been run to record the latest performance information.
Arguments:EcTInt:ParId, RWString:ParVal
Return Type:EcTVoid
Privilege:Public

**~PlPerformance -** This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlPerformance class has associations with the following classes:
Class: PlResourceRequirement
PlPGEProfile (Aggregation)

### 4.3.83 PlPgeFactory Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class is used to create PGE Profile objects. It can create the classes that make up a PGE Profile reguardless of the type of PGE.

**Attributes:**

None

**Operations:**

**Create -** This method takes in the address of a PGE Profile and creates an object of PlPGE.
Arguments:RWCString&
Return Type:PlPge*
Privilege:Public

**PlPgeFactory -** This is the constructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**~PlPgeFactory -** This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlPgeFactory class has associations with the following classes:
Class: PlPGE createdby
Class: PlPGE creates

### 4.3.84 PlPlanASCIIReportFile Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class represents an ascii readable externalized report format of the plan. The precise format of this report is TBD. The report will provide a summary view of the tasks planned

(no. of particular product types generated within the plan period) as well as a prediction of the generation time for each product.

**Attributes:**

None

**Operations:**

**Write -** Converts a local representation of the activities in a plan to an ASCII file that can be read by humans
Arguments:RWTvalSlist<PlActivities>
Return Type:Void
Privilege:Public
PDL: {
Open output file
Create summary information on activies and write them out
For list of activities
  Write out description
}

**Associations:**

The PlPlanASCIIReportFile class has associations with the following classes:
PlPublishedPlan (Aggregation)

### 4.3.85 PlPlanB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class represents an abstraction for a production plan. The class describes the metadata that will be stored for a plan within the PDPS database. The operations shown are an abstraction for those used within the planning framework.

**Attributes:**

**myActiveStatus -** Indicator to identify an active plan
Data Type:EcTBoolean
Privilege:Private
Default Value:False

**myDescription -** Descriptive text for operator comments describing conditions under which the plan was generated.
Data Type:RWCString
Privilege:Private
Default Value:

**myEndTime -** End time for the plan
Data Type:RWTIme
Privilege:Private
Default Value:

**myPlanName -** Descriptive name for plan to facilitate plan selection.
Data Type:RWCString
Privilege:Private
Default Value:

**myProdStrat -** The production strategy used to determine priorities for activities for this plan.
Data Type:RWCString
Privilege:Private
Default Value:

**myStartTime -** Start time for plan
Data Type:RWTime
Privilege:Private
Default Value:

**Operations:**

**DeletePlan -** Deletes the plan from the PDPS database along with the associated activities
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

**IdentifyDataDependencies -** Identifies the data dependencies between displayed plans from different DAACs.
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL: {
For the current Plan
If Imported Plan

```
   For each PlImportedActivity
     For each PlDataDependency
       For each plan in PlTimeLineDisplay::GetPlansDisplayed
       excluding the current plan
         If PlDataDependency = PlDataDependency, update display
   If Local Plan
   For each PlPGEActivity
     For each data granule in PlDPR::GetInputGranuleList() and
       PlDPR::GetOutputGranuleList()
     with a DataType that is needed by a remote DAAC or is
     produced at a remote DAAC
         If PlDataDependency = PlDataDependency, update display
}
```

**PlPlan -** Default Constructor
Arguments:
Return Type:EcTVoid
Privilege:Public

**PlPlan -** This operation is an abstraction for bringing an exported plan acquired from the data server into the Planning Workbench.
Arguments:PlExportedPlanNB &
Return Type:EcTVoid
Privilege:Public
PDL: {
  Open Plan Metadata File
  Read in parameters
  Construct plan object using these parameters
}

**PlPlan -** Default Constructor
Arguments:
Return Type:EcTVoid
Privilege:Public

**PlPlan -** This operation is an abstraction for bringing an exported plan acquired from the data server into the Planning Workbench.
Arguments:PlExportedPlanNB &
Return Type:EcTVoid
Privilege:Public
PDL: {
  Open Plan Metadata File
  Read in parameters
  Construct plan object using these parameters
}

**PlPlan -** Default Constructor
Arguments:
Return Type:EcTVoid
Privilege:Public

**PlPlan -** This operation is an abstraction for bringing an exported plan acquired from the data server into the Planning Workbench.
Arguments:PlExportedPlanNB &
Return Type:EcTVoid
Privilege:Public
PDL: {
  Open Plan Metadata File
  Read in parameters
  Construct plan object using these parameters
}

**PlanProductionRequest -** Adds the Data Processing Requests associated to a Production Plan. This involves creating an activity within the plan to describe the DPR, and allocating the resources for the DPR.
Arguments:PlProductionRequest: PR
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

**PlanSchedule -** This operation is an abstraction that describes the planning process by which activities are associated to resources in order to set out a time-line for the plan.
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

**Publish -** Initiates the activities involved in publishing a subset of a plan based on Production Requests
Arguments:RWTvalSlist<PlProductionRequest>
Return Type:Void
Privilege:Public
PDL: {
  For the current Plan
    Create list of activities by comparing each PlActivity
      If PlActivity::GetDPR::GetProductionRequest in subset PRs
    Create metadata file
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
        PlPlanBinaryReportFile::Write(List of activities in plan)
        PlExportedPlan::Export(List of activities in plan)

```
    Call PlPublishedPlan::InsertToDSS()
}
```

**Publish -** Initiates the activities involved n publishing a subset of a plan based on PGEs
Arguments:RWTvalSlist<PlPGE>
Return Type:Void
Privilege:Public

```
PDL: {
    Create metadata file
    Create list of activities by comparing each PlActivity
      If PlActivity::GetDPR::GetPGE in subset PGEs
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
         PlPlanBinaryReportFile::Write(List of activities in plan)
         PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}
```

**Publish -** Initiates the activities involved in publishing a subset of a plan based on Production Requests
Arguments:RWTvalSlist<PlProductionRequest>
Return Type:Void
Privilege:Public

```
PDL: {
  For the current Plan
    Create list of activities by comparing each PlActivity
      If PlActivity::GetDPR::GetProductionRequest in subset PRs
    Create metadata file
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
         PlPlanBinaryReportFile::Write(List of activities in plan)
         PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}
```

**Publish -** Initiates the activities involved n publishing a subset of a plan based on PGEs
Arguments:RWTvalSlist<PlPGE>
Return Type:Void
Privilege:Public

```
PDL: {
    Create metadata file
    Create list of activities by comparing each PlActivity
      If PlActivity::GetDPR::GetPGE in subset PGEs
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
         PlPlanBinaryReportFile::Write(List of activities in plan)
         PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
```

```
}
```

**Publish -** Initiates the activities involved in publishing a plan.
Arguments:
Return Type:EcTVoid
Privilege:Public

```
PDL: {
  For current Plan
    Create list of activities in plan
    Create metadata file
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
          PlPlanBinaryReportFile::Write(List of activities in plan)
          PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}
```

**Publish -** Initiates the activities involved in publishing a subset of a plan based on time.
Arguments:RWTime, RWTime
Return Type:EcTVoid
Privilege:Public

```
PDL: {
  For the current Plan
    Create list of activities by comparing each PlActivity
      If activity falls within the timeframe (inclusive)
    Create metadata file
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
          PlPlanBinaryReportFile::Write(List of activities in plan)
          PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}
```

**Publish -** Initiates the activities involved in publishing a plan.
Arguments:
Return Type:EcTVoid
Privilege:Public

```
PDL: {
  For current Plan
    Create list of activities in plan
    Create metadata file
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
          PlPlanBinaryReportFile::Write(List of activities in plan)
          PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}
```

**Publish -** Initiates the activities involved in publishing a subset of a plan based on time.
Arguments:RWTime, RWTime
Return Type:EcTVoid
Privilege:Public
PDL: {

```
  For the current Plan
    Create list of activities by comparing each PlActivity
      If activity falls within the timeframe (inclusive)
    Create metadata file
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
         PlPlanBinaryReportFile::Write(List of activities in plan)
         PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}
```

**Publish -** Initiates the activities involved in publishing a plan.
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL: {

```
  For current Plan
    Create list of activities in plan
    Create metadata file
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
         PlPlanBinaryReportFile::Write(List of activities in plan)
         PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}
```

**Publish -** Initiates the activities involved in publishing a subset of a plan based on time.
Arguments:RWTime, RWTime
Return Type:EcTVoid
Privilege:Public
PDL: {

```
  For the current Plan
    Create list of activities by comparing each PlActivity
      If activity falls within the timeframe (inclusive)
    Create metadata file
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
         PlPlanBinaryReportFile::Write(List of activities in plan)
         PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}
```

**Publish -** Initiates the activities involved in publishing a subset of a plan based on

Production Requests
Arguments:RWTvalSlist<PlProductionRequest>
Return Type:Void
Privilege:Public
PDL: {
  For the current Plan
    Create list of activities by comparing each PlActivity
      If PlActivity::GetDPR::GetProductionRequest in subset PRs
    Create metadata file
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
        PlPlanBinaryReportFile::Write(List of activities in plan)
        PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}

**Publish -** Initiates the activities involved n publishing a subset of a plan based on PGEs
Arguments:RWTvalSlist<PlPGE>
Return Type:Void
Privilege:Public
PDL: {
    Create metadata file
    Create list of activities by comparing each PlActivity
      If PlActivity::GetDPR::GetPGE in subset PGEs
    Call PlPlanASCIIReportFile::Write(List of activities in plan)
        PlPlanBinaryReportFile::Write(List of activities in plan)
        PlExportedPlan::Export(List of activities in plan)
    Call PlPublishedPlan::InsertToDSS()
}

**UnplanProductionRequest -** Removes the activities associated with a particular production request from the plan.
Arguments:PlProductionRequest: PR
Return Type:EcTVoid
Privilege:Public
PDL: No PDL

**UpdatePlan -** Updates the plan from the status of the activities in the PDPS database.
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL: //Create an ordered list of the Ground Events that are in the plan
//Verify whether any new ground events have been added
//If so alert operator, with option of including event
//Regenerate the plan

**Associations:**

The PlPlanB class has associations with the following classes:
Class: PlPlanningWorkbenchUI
Class: PlRescUseThreshNB constructsplaceholdersforOn-demandPRsusingthecumulative
Class: PlActivity contains
Class: PlProductionPlannersUI creates
Class: PlActivities iscontainedina
Class: PlProdStratNB iscreatedusing
Class: PlPublishedPlan ispublishedusing
Class: PlResourceManager usestomanageresources

### 4.3.86 PlPlanBinaryReportFile Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class represents an electronic parsable report format of the plan. The precise format of this report is TBD. The report will provide a summary view of the tasks planned (no. of particular product types generated within the plan period) as well as a prediction of the generation time for each product.

**Attributes:**

None

**Operations:**

**Write -** Converts the local representation of a plan into a file that is readable by other computer programs
Arguments:RWTValSlist<PlActivities>
Return Type:Void
Privilege:Public
PDL: {
Open output file
Create summary information on activies and write them out
For list of activities
  Write out description in machine readable format
}

**Associations:**

The PlPlanBinaryReportFile class has associations with the following classes:
PlPublishedPlan (Aggregation)

### 4.3.87 PlPlanGenerationUIB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class is an abstraction for the user interface to the planning workbench application. The class shows the functions available to the production scheduler in support of generating a plan

**Attributes:**

None

**Operations:**

**AddPRtoPlan -** Add the Data Processing Requests associated to a Production Request to a plan
Arguments:PlProductionRequest: PR
Return Type:void
Privilege:Public

**DelPRfromPlan -** Delete the Data Processing Requests associated to a Production Request from a plan
Arguments:PlProductionRequest: PR
Return Type:void
Privilege:Public

**NewPlan -** Generate a new plan
Arguments:
Return Type:void
Privilege:Public

**SelectActivePlan -** This operation is called when the Limited Automatic Replan brings up the Production Planning Workbench automatically. It loads in the current active plan and replans it based on the current state of the system.
Arguments:
Return Type:void
Privilege:Public

**SelectActivePlan -** This operation is called when the Limited Automatic Replan brings up

the Production Planning Workbench automatically. It loads in the current active plan and replans it based on the current state of the system - in addition, it takes as a parameter the PR id (myRequestId on PlProductionRequest), determines the OPR this id relates to and adds this production request to the plan before replanning.
Arguments:RWCString
Return Type:void
Privilege:Public

**SelectActivePlan** - This operation is called when the Limited Automatic Replan brings up the Production Planning Workbench automatically. It loads in the current active plan and replans it based on the current state of the system.
Arguments:
Return Type:void
Privilege:Public

**SelectPlan** - Select a plan for modification or activation
Arguments:
Return Type:void
Privilege:Public

**SelectProdStrat** - Abstract GUI operation which allows an operator to select a production strategy for a plan.
Arguments:RWCString: ProdStrat

**Associations:**
The PlPlanGenerationUIB class has associations with the following classes:
Class: PlReplanCriteria initiates
Class: PlReplanCriteria isinitiatedby
Class: PlProdStratNB selects
Class: PlProductionRequestB selectsforplanning
PlPlanningWorkbenchUI (Aggregation)

### 4.3.88 PlPlanMetadataFile Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class represents the metadata that will be used to describe the plan within the Document Data Server. The format of this report will be a Parameter=Value list line separated.

**Attributes:**

**myDAAC** - The site at which the plan was produced
Data Type:String
Privilege:Private
Default Value:

**myDataSets** - Describes the data sets which are planned for production within the plan.
Data Type:String
Privilege:Private
Default Value:

**myDescription** - Captures the operators description of the purpose of the plan (for example 30 day forecast)
Data Type:String
Privilege:Private
Default Value:

**myEndDay** - The end date of the plan
Data Type:Time
Privilege:Private
Default Value:

**myForecast** - The number if days time period which the plan covers
Data Type:int
Privilege:Private
Default Value:

**myStartTime** - The start date of the plan
Data Type:Time
Privilege:Private
Default Value:

**Operations:**

None

**Associations:**

The PlPlanMetadataFile class has associations with the following classes:
PlPublishedPlan (Aggregation)

### 4.3.89 PlPlanningWorkbenchUI Class

Parent Class:EcPfClient
Public:No
Distributed Object:No
Purpose and Description:
This class is an abstraction for the user interface to the planning workbench application. The interface will be developed with a suitable GUI builder tool.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**AddPlanNB** - This operation ACQUIRES a plan from the data server.
Arguments:GlUR &
Return Type:Void
Privilege:Public
PDL: {
Creates a ESDTReferenceCollector
Creates a DsClCommand for the ACQUIRE
Tells Data Server to push the Exported Plan on the local disk
}

**CreateTargetDateReport** - This operation creates a target date or "goal" report - that is, it compares the target completion dates for each PR with the predicted time of completion of all the DPRs for that PR. In addition, it compares the predicted completion time for each DPR that will create data granules needed at remote DAACs (as identified by a non-NULL value in PlDataType::myUsedByCenter) with the baseline completion time.
Arguments:
Return Type:Void
Privilege:Public
PDL:{

```
For current local Plan
First compare PR goals with whats in plan
  For each PR in plan
    For each DPR in PR, get the Activity
      If PlActivity::GetStartTime +
         PlPerformance(PlDPR::GetPGEID)::GetElaspedTime
      is later than PR::myCompletionGoal
      add information about this PR/DPR to report
Next compare baseline start times with what's in plan
  For each PR in plan, get the PGEid
    If any DataType produced by that PGEid is needed at remote DAAC
      For each DPR in PR, get the Activity
```

```
        If PlActivity::GetStartTime is later than
          PlDPR::GetBaselineTime
          add information about this PR/DPR to report
  Call  DisplayPopUp() and display report in it.
}
```

**DisplayInspectResultsNB -** This operation is called by the callback for the INSPECT that was performed on all the URs returned from the query on the plans in the data server. It takes the results from the inspect and displays them so the operator can select which plan to retrieve.

Arguments:
Return Type:Void
Privilege:Public
PDL: {

```
Display each UR with the metadata from the plan it represents as returned
   by the INSPECT submitted by DisplayQueryResults
}
```

**DisplayPopUp -** Abstract operation for displaying a pop up window
Arguments:

**DisplayQueryResultsNB -** This operation is called by callback for the query submitted by the PlanQueryNB. It submits an INSPECT for all the URs retrieved by the query.

Arguments:
Return Type:Void
Privilege:Public
PDL: {

```
For each UR returned from PlanQuery
  Do an inspect on the metadata for the plan from the Data Server
}
```

**PlanQueryNB -** This operation uses the selection criteria entered in a GUI screen to create a query to be submitted to the DIM.

Arguments:RWCString
Return Type:Void
Privilege:Public
PDL: {

```
If first time called, construct DmLmClRequestServer object with
       myUser = Planning Workbench,
       myServer will be stored in a configuration file
Start a new search request
GUI interface has query criteria for plans filled in by user
   This criteria is used to construct ESQL
Call DmLmClRequestServer::SendQuery(ESQL query)
}
```

**ProductionSchedulerCommands** - This is an abstraction of the GUI operations intended for the production scheduler to use.
Arguments:
Return Type:Void
Privilege:Public

**SelectEntryScreen** - This is an abstraction of a GUI menu that allows the operator to select which entry screen he/she wants to use.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlPlanningWorkbenchUI class has associations with the following classes:
Class: DsClCommand ACQUIREsplansusing
Class: PlPlanB
Class: PlPlanningWorkbenchUI
Class: DmLmClRequestServer QUERIESforplansfromotherDAACsusing
Class: DsClESDTReferenceCollector usedforQUERIESandACQUIRES
Class: GlCallback usestogetcallbackfromqueryandacquires
Class: GlParameterList useswhenconstructingACQUIREs

### 4.3.90 PlPopupMessage Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This objects is an abstraction of a display of a pop-up window

**Attributes:**

None

**Operations:**

**Display** - An abstraction of the GUI operations necessary to bring up a pop-up window and display messages to the operator.
Arguments:

**Associations:**

The PlPopupMessage class has associations with the following classes:
Class: PlReplanCriteria notifiesOperatorUsing
Class: PlReplanCriteria notifiesoperatorusing

### 4.3.91 PlProdStratNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class describes the production strategies used by the DAAC to determine data processing request priorities.

**Attributes:**

**myAgingDelta -** This is an amount added to the calculated priority of an activity if the DPR was already selected to be in a previous active plan, but has not yet been completed. This value allows an operator to "bump up" the priorities of older jobs.
Data Type:EcTInt
Privilege:Private
Default Value:

**myInterDAACDelta -** This is an amount added to the calculated priority of an activity if the DPR produces data that is needed at a remote DAAC.
Data Type:EcTInt
Privilege:Private
Default Value:

**myPGEWeight -** This attribute is the weight given to the priority based on the PGE.
Data Type:EcTFloat
Privilege:Private
Default Value:

**myPRTypeWeight -** This attribute is the weight given to the priority based on the production request type.
Data Type:EcTFloat
Privilege:Private
Default Value:

**myProdStratId -** Primary key for the database table represented by this object
Data Type:RWCString
Privilege:Private

Default Value:

**myUserSelectedPriorityWeight -** This attribute is the weight given to the priority entered by the user for the Production Request
Data Type:EcTFloat
Privilege:Private
Default Value:

**myUserWeight -** This attribute is the priority given a production request based on the user.
Data Type:EcTFloat
Privilege:Private
Default Value:

**Operations:**

**AssignPriorityToActivity -** This method determines the priority for a DPR based on the production strategies for a particular plan.
Arguments:PlActivity &
Return Type:Void
Privilege:Public
PDL: {

```
For DPR related to Activity
    priority = PlPGEPriority::GetPriority(PlDPR &) * myPgeWeight
             + PlUserPriority::GetPriority(PlDPR &) * myPRTypeWeight
             + PlPRPriority::GetPriority(PlDPR &) * myUserWeight
             + PlProductionRequest::GetPriority * myUserSelectedPriorityWeight
If  DPR::myPredictedStartTime is before current time,
    priority = priority + myAgingDelta
If any DataType produced by that PGEid is needed at remote DAAC
     priority = priority + myInterDAACDelta
}
```

**Associations:**
The PlProdStratNB class has associations with the following classes:
Class: PlDPRB basesprioritieson
Class: PlPGEActivity determinespriorityof
Class: PlProdStratUINB displays
Class: PlPlanB iscreatedusing
Class: PlPlanGenerationUIB selects

### 4.3.92 PlProdStratUINB Class

Parent Class:PlEntryScreenNB
Public:No
Distributed Object:No
Purpose and Description:
This class is the user interface that allows the user to update/add/delete the production atrategies.

**Attributes:**

**PGEPriorityDescWindow** - This attribute is the window in GUI to be used for entering the PGEs and their priorities for production strategies.
Data Type:
Privilege:Private
Default Value:

**PRTypePriorityDescWindow** - This attribute is the window in the GUI to be used for entering the production request types and their priorities for production strategies.
Data Type:
Privilege:Private
Default Value:

**UserPriorityDescWindow** - This attribute is the window in the GUI to be used for entering the users and their priorities for production strategies.
Data Type:
Privilege:Private
Default Value:

**WeightDescWindow** - This attribute is the window in GUI to be used for entering the weights to priority for the various production strategies.
Data Type:
Privilege:Private
Default Value:

**Operations:**

**Add** - Abstraction of a GUI operation for adding a new value/priority to the production strategy attribute lists(example - adding a newe PGE with an associated priority to the PlPGEPriorityNB list).
Arguments:
Return Type:Void
Privilege:Public

**Delete** - Abstraction of a GUI operation for deleting a new value/priority to the production

strategy attributes lists (example - deleting a PGE with an associated priority from the PlPGEPriorityNB list).
Arguments:
Return Type:Void
Privilege:Public

**Modify** - Abstraction of a GUI operation for modifying a priority from the production strategy attributes lists (example - changing the proirity associated with a PGE from 5 to 10).
Arguments:
Return Type:Void
Privilege:Public

**Select** - Abstraction of a GUI operation for selecting a value/priority from the production strategy attributes lists (example - selecting a PGE ith an associated priority from the PlPgePrioriityNB list).
Arguments:
Return Type:Void
Privilege:Public

**Associations:**
The PlProdStratUINB class has associations with the following classes:
Class: PlProdStratNB displays

### 4.3.93 PlProductionPlannersUI Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class is an abstraction for the user interface to the planning workbench application. The interface will be developed with a suitable GUI builder tool. The class does describe the basic operations that are provided from the interface.

**Attributes:**

**PlanSelectionWindow** - Abstraction of a GUI operation operation of selecting a particular plan to display in a timeline.
Data Type:
Privilege:Private
Default Value:

**SchedulingPeriod** - Abstraction of a GUI operation of selecting the scheduling period of a particular plan.

Data Type:
Privilege:Private
Default Value:

**Operations:**

**ActivateSchedule -** Abstraction of a GUI operation of activating a plan.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlProductionPlannersUI class has associations with the following classes:
Class: PlPlanB creates

### 4.3.94 PlProductionRequestB Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is the instructions describing the order for data set(s) to be produced. A production request typically specifies a request for a Data Set to be produced for an extended period of time (e.g a month's worth of some product). There are three types of production requests: Standard, Re-processing and On-Demand.

**Attributes:**

**myDPRsNB -** This list is a list of DPRs generated by a production request.
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**myDataCollectionStartTime -** Start time for the production request.
Data Type:RWTime
Privilege:Private
Default Value:

**myDataCollectionStopTime -** Stop time for the production request.
Data Type:RWTime
Privilege:Private

Default Value:

**myDescription** - A textual description of the production request.
Data Type:RWCString
Privilege:Private
Default Value:

**myNumDPRsToKeep** - For intermittently activated PGEs, this indicates the number of DPRs to be kept for the PR. For normal PGEs, the value is 0.
Data Type:EcTInt
Privilege:Private
Default Value:

**myNumDPRsToSkip** - For intermittently activated PGEs, this indicates the number of DPRs to skip (after a number, specified in myNumDPRstoKeep, are created). For normal PGEs, this value is 0.
Data Type:EcTInt
Privilege:Private
Default Value:

**myOutputDataType** - Idemtifies the product desired by the ECS User.
Data Type:RWCString
Privilege:Private
Default Value:

**myPGEIdentifier** - Identifies the PGE
Data Type:RWCString
Privilege:Private
Default Value:

**myPRCollectionNB** - Pointer to my parent PlPRCollectorNB
Data Type:RWCString
Privilege:Private
Default Value:

**myPRTypeNB** - Indicates production type: routine, on-demand, or reprocessing.
Data Type:RWCString
Privilege:Private
Default Value:

**myPriority** - User requested priority for the submitted production request.
Data Type:EcTInt
Privilege:Private
Default Value:

**myRequestId** - A unique identifier for this production request.
Data Type:RWCString
Privilege:Private
Default Value:

**myRequesterId** - Identifies the person entering the production request.
Data Type:RWCString
Privilege:Private
Default Value:

**myTargetDate** - The desired time that a production request will be completed - for informational purposes only.
Data Type:RWTime
Privilege:Private
Default Value:

**myUserTypeNB**
Data Type:RWCString
Privilege:Private
Default Value:

**myUsrParaList** - A list of PGE parameters for this production request.
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**myUsrParaValueList** - A list of parameter values for the PGE associated with this production request.
Data Type:RWTValSlist<RWCString>
Privilege:Private
Default Value:

**Operations:**

**CalculateResourceUsageNB** - Determines the resource usage of a Production Request by multiplying the number of DPRs needed to satify this request with the PlResourceRequirement and PlPerformance values in PlPGEProfile for the PGE in the request.
Arguments:enum
Return Type:EcTFloat
Privilege:Public
PDL:{
 total = 0
For each DPR related to this PR
    get PGEId

```
    If  (parameter == Disk space)
        total = total +PlResourceRequirement::myDiskSpace
    If  (parameter == wall clock time)
        total = total +PlPerformance::myElapsedTime
    If  (parameter == RAM)
        total = total + PlPerformance::myMaxMemoryUse
 return total
}
```

**DefinePGERuns** - Defines the DPRs corresponding to the PGE.
Arguments:
Return Type:Void
Privilege:Public

**Modify** - Modifies the production request.
Arguments:RWCString:Modfield, RWCString:ModVal
Return Type:Void
Privilege:Public

**PlProductionRequest** - Constructor for the PlProductionRequest class.
Arguments:(RWCString:Product, RWCString:PGE, RWTime:Start, RWTime:Stop, ECTInt:Prior)
Return Type:Void
Privilege:Public
PDL:

**RetrieveAllProdReq** - Retrieves all the production requests currently in effect.
Arguments:
Return Type:Void
Privilege:Public

**StoreProductionRequest** - Stores the production request in the PDPS database.
Arguments:PlProductionRequestB
Return Type:Void
Privilege:Public

**Associations:**
The PlProductionRequestB class has associations with the following classes:
Class: PlProductionRequestUI displays
Class: PlDPRB isbrokenupintoindividual
Class: PlDPRB ismadeupof
Class: PlPGE requests
Class: PlPlanGenerationUIB selectsforplanning
PlPRCollectionNB (Aggregation)

### 4.3.95 PlProductionRequestUI Class

Parent Class:EcPfClient
Public:No
Distributed Object:No
Purpose and Description:
This class is an abstraction for the user interface to the production request editor application.

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**AddProductionRequest** - Creates a production request.
Arguments:
Return Type:void
Privilege:Public

**ChangeAlternateInputOrder** - Allows the preference ordering for alternate inputs to be changed.
Arguments:RWCString:ProductionRequestID
Return Type:Void
Privilege:Public

**CreatePRCollection** - Initializes a collection, or grouping, of production requests.
Arguments:RWCString
Return Type:void
Privilege:Public

**DelProductionRequest** - Deletes a production request.
Arguments:
Return Type:void
Privilege:Public

**DeletePRCollection** - Deletes a production request collection.
Arguments:RWCString
Return Type:void
Privilege:Public

**DisplayPGEInfo** - Diplays the parameters for a selected PGE.
Arguments:RWCString:PGEID
Return Type:void
Privilege:Public

**DisplayPGEs -** Displays a list of PGEs that can be added to the production request. The list can be filtered with the filter string.
Arguments:RWCString:Filter
Return Type:void
Privilege:Public

**DisplayProductDependencies -** Displays a list of products that are dependent upon the selected product. The list shows products at the next level higher than the selected producted (e.g. level 3 products will be listed as dependents of a level 2 product).
Arguments:RWCString:OutputDataType
Return Type:RWTValSlist<RWCString>
Privilege:Public

**DisplayProducts -** Displays a list of products which can be selected for a production request. The list can be filtered with a filter string.
Arguments:RWCString:Filter
Return Type:void
Privilege:Public

**GenerateCostReport -** Generates a report containing resource utilization costs for the production request. Resources include CPU and disk.
Arguments:
Return Type:void
Privilege:Public

**MapPGEtoProduct -** Returns the product associated with a PGE.
Arguments:RWCString:PGEID
Return Type:RWCString:OutputDataType
Privilege:Public

**MapProducttoPGE -** Returns the PGE associated with a product.
Arguments:RWCString:OutputDataType
Return Type:RWCString:PGEID
Privilege:Public

**ModifyProductionRequest -** Modifies a production request.
Arguments:
Return Type:void
Privilege:Public

**SelectPGE -** Allows for selection of a PGE.
Arguments:RWCString:PGEID
Return Type:Void
Privilege:Public

**SelectProduct -** Allows for selection of a product.
Arguments:RWCString:OutputDataType
Return Type:Void
Privilege:Public

**Associations:**

The PlProductionRequestUI class has associations with the following classes:
Class: PlPRCollectionNB displays
Class: PlProductionRequestB displays
Class: PlDataTypeCatalogue storesrequestsin

### 4.3.96 PlPublishedPlan Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This object encapsulates the methods requred to insert externalized formats of the plan into the document data server

**Attributes:**

None

**Operations:**

**InsertInDDS -** Insert the published plan into the Document Data Server
Arguments:
Return Type:void
Privilege:Public
PDL:{
Build parameter list for insert (this should be fairly constant for plans)
Submit request for insert
}

**Associations:**

The PlPublishedPlan class has associations with the following classes:
Class: PlPlanB ispublishedusing
Class: GlParameterList isusedwhenconstructingINSERTandACQUIREcommands
Class: DsClESDTReferenceCollector usedforACQUIREs
Class: GlCallback usedforcallbacksfromINSERTand

Class: DsClCommand usedtoINSERTandACQUIREplans

### 4.3.97 PlReplanCriteria Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
Abstract base class for criteria for replan notification

**Attributes:**

**myAutoReplanYN** - Attribute indicates whether a particular criteria should initiate an automatic replan - this is a place holder for release C when automatic replanning is done.
Data Type:EcTBoolean
Privilege:Private
Default Value:False

**myCheckThisYN** - Attribute indicates whether a particular criteria should be checked, and if met, the operator notified that a replan should be done.
Data Type:EcTBoolean
Privilege:Private
Default Value:False

**myReport** - A string that contains the report displayed for an operator that indicates why a replan should be considered
Data Type:RWCString
Privilege:Private
Default Value:EcNULL

**Operations:**

**AddToReport** - Virtual Operation - for each criteria, this operation checks if a replan should be done.
Arguments:
Return Type:RWCString
Privilege:Public

**CheckReplan** - Virtual opaeration - for each criteria, this operation check if a replan should be done.
Arguments:
Return Type:EcTBoolean
Privilege:Public

**NotifyOperator** - Display a pop-up window with a notice to the operator that a replan should be done, and the output of the CreateReport operation which indicates why, and two choices "Replan" or "Cancel".
Arguments:
Return Type:EcTVoid
Privilege:Public
PDL: Construct PlPopupMessage object
Display report in window
Wait for Operator selection
If replan
Use a system call to start Planning Workbench with current active plan
if its already running, notify operator

**Associations:**

The PlReplanCriteria class has associations with the following classes:
Class: PlOnDemandPRNB checkforneedforreplannotification
Class: PlReplanCriteriaUI displays
Class: PlPlanGenerationUIB initiates
Class: PlPlanGenerationUIB isinitiatedby
Class: PlDataAvailabilityTimes isusedtocheckforreplanby
Class: PlPopupMessage notifiesOperatorUsing
Class: PlPopupMessage notifiesoperatorusing

### 4.3.98 PlReplanCriteriaUI Class

Parent Class:PlEntryScreenNB

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**Modify**
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlReplanCriteriaUI class has associations with the following classes:
Class: PlReplanCriteria displays

### 4.3.99 PlRescUseThreshNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is a table containing the resource usage thresholds for On-Demand production requests.

**Attributes:**

**myCumulative** - This attribute is an enumerated type which identifies which type of resource usage threshold the object represents where, 1 = the cumulative on-demand usage for a TBD time, and 2 = the usage for each on-demand production request.
Data Type:EcTInt
Privilege:Private
Default Value:

**myPercentResources** - This attribute is the percentage of resources allowed of myresourcetype for an on-demand processing request.
Data Type:EcTFloat
Privilege:Private
Default Value:

**myResourceType** - This attribute is an enumerated type which represents a type of resource to be checked for on-demand production request thresholds, 1 = cpu, 2 = disk space, 3 = memory, etc.
Data Type:EcTInt
Privilege:Private
Default Value:

**Operations:**

**CheckThreshold** - This operation iterates through the resource types comparing the resource requirement to allowed resource threshhold. If the threshhold is exceeded for either the job threshhold or cumulative threshhold then the return value is FALSE otherwise UpdateRescUse is called in PlOnDemandManagerNB to update the Cumulative Resource Use for total On Demand jobs and TRUE is returned to the calling operation.
Arguments:PlOnDemandPRNB &
Return Type:Boolean

Privilege:Public

**PlRescUseThreshNB** - Default constructor
Arguments:
Return Type:PlRescUseThreshNB
Privilege:Public

**~lRescUseThreshNB**
Arguments:
Return Type:Void
Privilege:Public

**Associations:**
The PlRescUseThreshNB class has associations with the following classes:
Class: PlOnDemandPRNB comparewith
Class: PlPlanB constructsplaceholdersforOn-demandPRsusingthecumulative
Class: PlRescUseThreshUINB displays
Class: PlActivity placeholdersforon-demandjobscreatedusing

### 4.3.100 PlRescUseThreshUINB Class

Parent Class:PlEntryScreenNB
Public:No
Distributed Object:No
Purpose and Description:
This class is the user interface that allows the user to update/add/delete resource usage thresholds for on-demand production requests.

**Attributes:**

**ThreshDescWindow** - This attribute is the window in the GUI to be used for entering resource thresholds for on-demand production requests.
Data Type:
Privilege:Private
Default Value:

**Operations:**

**Modify** - Abstraction of a GUI operation for modifying the values of the resource usage threshold with which on-demand Production Request are compared.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**
The PlRescUseThreshUINB class has associations with the following classes:
Class: PlRescUseThreshNB displays

### 4.3.101 PlResource Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is the base class in a generalization heirachy describing the production resource.

**Attributes:**

**myID -** Unique identifier for the resource
Data Type:int
Privilege:Private
Default Value:

**myName -** The name of the resource
Data Type:String
Privilege:Private
Default Value:

**Operations:**

None

**Associations:**

The PlResource class has associations with the following classes:
Class: PlResourceManager

### 4.3.102 PlResourceChange Class

Parent Class:PlReplanCriteria
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
One of the replan criteria - this object is used to compare an old resource plan with the

current resource plan.

**Attributes:**

**myTimeFrame -** This is the time frame (from the time the new resource plan was receiveed) within the ground events which should be checked for a change.
Data Type:RWTime
Privilege:Private
Default Value:

**Operations:**

**AddToReport -** This operation accepts a new ground event or a deleted ground event and adds this to a report of the changes between the new resource plan and the old one.
Arguments:PlGroundEvent &
Return Type:RWCString
Privilege:Public

**CheckReplan -** Checks the new resource plan against the old resource plan.
Arguments:??
Return Type:Void
Privilege:Public

**Associations:**

The PlResourceChange class has associations with the following classes:
None

### 4.3.103 PlResourceManager Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class represents an abstraction for the resource management capabilities used when generating a plan, describing the operations required to match resource requirements of an activity to the available resources, and to allocate the resource for the activity.

**Attributes:**

None

**Operations:**

**AllocateResources** - Allocate the resources to an activity
Arguments:PlResource: Resource, PlActivity: Activity
Return Type:Void
Privilege:Public

**DeallocateResources** - Deallocate the resources for an activity
Arguments:PlResource: Resource, PlActivity: Activity
Return Type:Void
Privilege:Public

**MatchResourceRequirement** - Match the resource requirements of an activity to a resource
Arguments:PlResourceRequirements
Return Type:PlResource
Privilege:Public

**Associations:**

The PlResourceManager class has associations with the following classes:
Class: PlResource
Class: PlResourceManager
Class: PlPlanB usestomanageresources

### 4.3.104 PlResourceRequirement Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class contains a description of the resource requirements of a PGE, which may be matched against the resource configuration known to the Planning subsystem.

**Attributes:**

**myComputer** - A computer within the string required by the PGE
Data Type:String
Privilege:Private
Default Value:

**myDiskSpace** - The disk space required for a PGE
Data Type:int
Privilege:Private
Default Value:

**myNCPUs** - The number of CPUs required for a PGE
Data Type:int
Privilege:Private
Default Value:

**myOperatingSystem** - The operating system for which a PGE is configured
Data Type:String
Privilege:Private
Default Value:

**myString** - The string required by the PGE
Data Type:String
Privilege:Private
Default Value:

**Operations:**

**PlResourceRequirement** - Constructor method
Arguments:
Return Type:PlResourceRequirement
Privilege:Public

**~PlResourceRequirement** - Destructor method
Arguments:
Return Type:void
Privilege:Public

**Associations:**

The PlResourceRequirement class has associations with the following classes:
Class: PlPerformance
Class: PlGroundEvent hasa
Class: PlPGE isreferencedby
Class: PlDPRB isrequiredby
PlPGEProfile (Aggregation)

### 4.3.105 PlRoutineArrival Class

Parent Class:PlDataSource
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:

This class is a specialization of the PlDataSource class and describes the most frequent method for predicting data arrivals within the ECS (at least for the TRMM data sets). This class contains the attributes and operations required to describe routine ingest of external data.

**Attributes:**

**myBoundary** - A definitive boundary time from which predicted times can be derived.
Data Type:PlBoundary
Privilege:Private
Default Value:

**myDelay** - The average delay between data collection and the arrival that of the Data Granule within ECS specified in seconds.
Data Type:EcTULongInt
Privilege:Private
Default Value:

**myPeriod** - A period of time (hour, day, week, set of orbits) used in predicting arrivals.
Data Type:PlPeriod
Privilege:Private
Default Value:

**Operations:**

**PlRoutineArrival**
Arguments:
Return Type:Void
Privilege:Public

**PlRoutineArrival** - Constructor method that is created from the database.
Arguments:const RWCString &
Return Type:Void
Privilege:Public

**PredictArrivals** - Method for predicting the arrival of data granules within an interval.
Arguments:const PlTime &, const PlTime &
Return Type:EcTVoid
Privilege:Public

**~PlRoutineArrival**
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlRoutineArrival class has associations with the following classes:
None

### 4.3.106 PlService Class

Parent Class:PlResource
Public:No
Distributed Object:No
Purpose and Description:

**Attributes:**

All Attributes inherited from parent class

**Operations:**

All Operations inherited from parent class

**Associations:**

The PlService class has associations with the following classes:
None

### 4.3.107 PlSourcetoDsHistoryNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class contains a list of records that track the interval between the time at which the Data Availability Times Schedule predicts that a data type will arrive from its source (i.e. EDOS or another DAAC) to the Data Server.

**Attributes:**

**myDataTypeID** - Identifies the Data Type associated with a Data Granule.
Data Type:RWCString
Privilege:Private
Default Value:

**myLastActualArrival** - This atrbute stores the last actual arrival time of this data type, to

the data server
Data Type:RWTime
Privilege:Private
Default Value:

**myLastPredictedArrival -** This atrribute provides the last predicted arrival time of a data type.
Data Type:RWTime
Privilege:Private
Default Value:

**myMovingAverageTime -** The moving average of the source to data server interval for a given data type.
Data Type:RWTime
Privilege:Private
Default Value:

**myNumberOfEstimates -** This provides the number of samples used to compute the moving average of the source to data server interval.
Data Type:EcTInt
Privilege:Private
Default Value:

**myPrevTransferTimes -** This attribute stores a list of previous transfer times up to a maximum determined by MSS for sampling.
Data Type:RWTValSlist<RWTime>
Privilege:Private
Default Value:

**myStandardDeviation -** Specifies the acceptable standard deviation from the historical averages of source to data server time.
Data Type:EcTInt
Privilege:Private
Default Value:

**Operations:**

**AddLastArrivalTime -** This operation will store the last arrival time of the specified data type, once notification is received that this data type has arrived.
Arguments:RWTime<myActualAvailability>
Return Type:EcUtStatus
Privilege:Public
PDL:{
Update the last arrival time in PlSourceToDsHistory based on data arrival
PlSourceToDsHistory:myLastActualArrival =

```
   PlDataGranule:myActualAvailability
}
```

**UpdateHistoricalInfo -** This operation will update historical information stored on a particular data type.
Arguments:RWTime
Return Type:EcUtStatus
Privilege:Public
PDL:{

```
  This routine is called if a new moving average was computed
  if PlDataTransferHistory:myMovingAverageFlag = True
  {
    Based on new moving average of transfer time and the predicted arrival
    time of the granule from the new Data Schedule, update historical
    information accordingly.
    Add PlDataTransferHistory:myMovingAverage on the top of the list
   if (the number of samples in PlSourcetoDsHistory:myPreviousTransferTimes
      > the MaximumRequiredSamples)
   {
      Remove PlSourcetoDsHistory:myPrevTransferTimes[last] off the list
   }
   else
   {
      PlSourcetoDsHistory:myNumberOfEstimatess++
   }
  }
}
```

**Associations:**
The PlSourcetoDsHistoryNB class has associations with the following classes:
Class: PlDataTranferHistory
Class: PlDataGranule UpdatesArrivalTime
Class: PlTileScheduledNB references

### 4.3.108 PlString Class

Parent Class:PlResource
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
A string describes the logical collection of a number of resources allocated for an instruments processing needs

**Attributes:**

**myComputerList -** Describes the list of computers that make up the string
Data Type:List
Privilege:Private
Default Value:

**Operations:**

**AddComputer -** Add a computer to the list
Arguments:PlComputer: Comp
Return Type:void
Privilege:Public

**RemoveComputer -** Remove a computer from the list
Arguments:PlComputer: Comp
Return Type:void
Privilege:Public

**Associations:**

The PlString class has associations with the following classes:
Class: PlComputer
PlService (Aggregation)

### 4.3.109 PlSubMsgCb Class

Parent Class:EcMpMsgCb
Public:No
Distributed Object:No
Purpose and Description:
Specialization class to provide the subscription manager callbacks to the MsManager in order to be activated for the appropriate lifecycle services (shutdown, resume, suspend notify).

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**HandleCbMsg -** This function performs callback chores.
Arguments:Message, MessageClass, MessageLength, MessageId, ReplyMessageId, SenderName
Return Type:EcUtStatus

```
Privilege:Public
PDL:{
  HandleCbMsg Takes the message, gets the UR and constructs a EcUrUR object
  which contains event Id followed by UR. From this info we can extract UR
  byte stream.
  PlSubMsgCb::HandleCbMsg(Message, MessageClass, MessageLength,
  MessageId, ReplyMessageId, SenderName)
  GlUR urObj((char *)msgP);
  GlClient glClient ("UserProfile");
  DsClESDTReference esdtRef (urObj, urObj,glClient );
  GlParameterList parmList;
  The metadata is obtained from the data server by performing an inspect on
  the data server based on the UR.  The PlDataGranule is then constructed
  and gets and set functions are then performed to get the granule Id and
  update ESDT Parameter values in the database.
  PlDataGranule::RegisterAvailability(GlUR:instUR, GlParameterList:
  instESDTParmVals) registers the data Availability
  The DPRs are searched  for jobs requiring this data to process.
  PlDataGranule::FindAssociatedDPRs(DprList)
  For DPRs awaiting  data granule
  {
     Call GetPGEType to get the PGE type in order to begin processing.
     PlDPRB::GetPGEType()
     If the granule is a geographical tile...
     if (Type = TilingPGE)
     {
       Get Tile coordinates from the PGE profile based on the tile Id.
       Coordinates = PlTile:myCoordinates
       Query the Data Server with coordinates of tile to get
       ESDTReferenceCollector
       DsClQuery::DsQuery
       Inspect the Data Server to get the
       URs for all granules in the tiling scheme.
       For all URs
       {
         Create granules for all granules return from query
         PlDataGranule::Create
       }
       Delete initial Granule created for tiling scheme by the Production
       request editor
       PlDataGranule::DeleteGranule
     }
     Check metadata for all Granules.  For tiling there will be
     several, all other PGE types will do one check at this point.
     For All Granules
```

```
        {
          Check to see if Granule passes QA Threshold.
          if (PlMetaData::CheckForCondition = false)
          {
            If it fails check if the granule is required
            if (PlAlternateDataGranuleNB:myType <> Required)
            {
              If QA fails and the PGE is not a required input set my
              accepted to false, to indicate that it is not a good
              alternate input
              PlAlternateDataGranule:myAccepted = False
              Change the priority ordering of the data granules
              PlAlternateDataGranule:ChangeOrder
            }
            else
            {
              If QA fails and input is required  return with
              failure message.
              Return
             }
           }
        }loop
        Call check Availability function to determine whether all necessary
        data is available to release job to processing
        PLDPRB:CheckAvailability
        release job if check availability is set to true
        if (PLDPRB::CheckAvailability is true)
        {
          Release job to processing
          Release()
        }

      }loop
    }
```

**Associations:**

The PlSubMsgCb class has associations with the following classes:

Class: DsClESDTReference
Class: DsClQuery
Class: EcUrUR
Class: PlDataAvailabilityTimes NotificationofarrivalofDAT
Class: PlDataGranule Notificationofarrivalofgranule
Class: PlSubscriptionManager

### 4.3.110 PlSubscriptionManager Class

Parent Class:EcPfManagedServer
Public:No
Distributed Object:No
Purpose and Description:
This class contains the main application methods associated with the subscription manager application

**Attributes:**

All Attributes inherited from parent class

**Operations:**

**DeinstallReceivers -** Deinstalls the receivers for each data type that is subscribed to
Arguments:
Return Type:void
Privilege:Public

**InstallReceivers -** Installs the receivers for each data type that is subscribed to
Arguments:
Return Type:void
Privilege:Public

**PfGetShutdownSeconds**
Arguments:a_level:EcTAgMgtmLevel

**PfShutdown -**
Arguments:EcTAgMgmtLevel,EcTint:ShutdownReason,EcTint:GracefulFlag
Return Type:EcUtStatus
Privilege:Public

**PfShutdown**
Arguments:shutdownlevel:EcTAgMgmtLevel,EcTInt,EcTInt

**PfStartMonitoring**
Arguments:

**PfStopMonitoring**
Arguments:EcTInt

**Start**
Arguments:
Return Type:EcUtStatus

Privilege:Public

**Associations:**

The PlSubscriptionManager class has associations with the following classes:
Class: PlSubMsgCb
Class: EcEvent logs

### 4.3.111 PlSubscriptionSubmitIF Class

Parent Class:EcPfClient
Public:No
Distributed Object:No
Purpose and Description:
This class is an abstraction for the user interface to the subscription submission application. The interface will be developed with a suitable GUI building tool.

**Attributes:**

**DataTypeSelectionWindow** - Abstraction of a GUI window displaying a list of data types from which the user can select.
Data Type:
Privilege:Private
Default Value:

**SubscriptionSubmissionControl** - Abstraction of a GUI window from which a user can submit or withdraw subscriptions.
Data Type:
Privilege:Private
Default Value:

**Operations:**

**DisplayDataTypes** - This operation displays a list of the dynamic data types known to the Planning subsystem as required input to a PGE.
Arguments:
Return Type:Void
Privilege:Public

**Initialize** - Initialize the subscription submission GUI.
Arguments:
Return Type:Void
Privilege:Public

**SelectDataType** - This operation allows the user to select a Data Type from the displayed list.
Arguments:
Return Type:Void
Privilege:Public

**SubmitSubscription** - Submit a subscription for the selected Data Type.
Arguments:
Return Type:Void
Privilege:Public

**WithdrawSubscription** - Withdraw the subscription for the selected Data Type.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlSubscriptionSubmitIF class has associations with the following classes:
Class: DsClSubscription creates
Class: PlDataSource submitssubscriptionsusing
Class: PlOnDemandManagerNB submitssubscriptionsusing
Class: PlDataTypeB updates

### 4.3.112 PlTile Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This defines a tile used by a TileScheduled PGE. It is a geographic area that will be processed by the PGE.

**Attributes:**

**myCoordinates** - These are the coordinates of the tile, the four Latitude/Longitude pairs that define the area covered by the tile.
Data Type:List of EctFloat
Privilege:Private
Default Value:

**myTileID** - This is the identifier for the tile. Each tile in a cluster will have a unique

identifier.
Data Type:EctInt
Privilege:Private
Default Value:

**Operations:**

**PlTile** - This is the default constructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**PlTile** - This constructor takes in a TileID when it creates an instance of PlTile.
Arguments:EctInt TileID
Return Type:Void
Privilege:Public

**ReturnCoordinates** - This method returns the coordinates of the tile stored in myCoordinates.
Arguments:
Return Type:List of EctFloat
Privilege:Public

**ReturnTileID** - This method returns the value in myTileID.
Arguments:
Return Type:EctInt
Privilege:Public

**SetCoordinates** - This method sets the coordinates for the tile.
Arguments:List of EctFloat Coordinates
Return Type:Void
Privilege:Public

**~PlTile** - This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlTile class has associations with the following classes:
Class: PlCluster

### 4.3.113 PlTileScheduledNB Class

Parent Class:PlPGE
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is a table used by the PGEs scheduled for tiling to generate the DPRs and their input data.

**Attributes:**

**myBoundary** - A time from which all periods are referenced. For tile clusters, it is the time of the first orbit of a cluster.
Data Type:RWTime
Privilege:Private
Default Value:

**myDelayFactor** - An amount of time used in calculating cluster availability which accounts for deviations in predictions.
Data Type:RWTime
Privilege:Private
Default Value:

**myL0DataType** - The LO data type that the tiles will be based upon. Used to calculate the time to move the L0 data from EDOS to the dataserver.
Data Type:ESDT
Privilege:Private
Default Value:

**myNumClusters** - The number of clusters for this PGE.
Data Type:EcTInt
Privilege:Private
Default Value:

**myPeriod** - The amount of time for a complete set of orbits that include a cluster of tiles.
Data Type:RWTime
Privilege:Private
Default Value:

**myProductionDeltaTime** - Estimated time to produce all intermediate products used as input to produce tiles.
Data Type:RWTime
Privilege:Private
Default Value:

**myTileDefinition -** This attribute is used to create the tile or spatial based input data.
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**CalcTileOverlap -** This calculates any geographic or input data overlap between tiles.
Arguments:
Return Type:Void
Privilege:Public

**GenerateDPRs -** Generates a DPR for each tile in each cluster.
Arguments:PlProductionRequest
Return Type:Void
Privilege:Public
PDL://

```
// // Determine cluster boundary:
//
// Query dataserver for most recent L0 data granule of this type
//    with orbit = PlCluster::myLastOrbit
// myBoundary = PlDataGranule::myStartTime
//
// // Determine cluster period:
//
// Query dataserver for second most recent L0 data granule of this type
//    with orbit = PlCluster::myLastOrbit
// myPeriod = difference in data granule start times
//
// // Calculate orbit duration:
// OrbitDuration = myPeriod / MaxOrbitsinPeriod
//
// // Get the time it takes for the L0 data to travel from EDOS to DSS
// TransitTime = PlSourcetoDsHistory::GetMovingAverageTime (myL0DataType)
//
// // Get intermediate product production time (filled in at SSI&T)
// ProductionTime = myProductionDeltaTime
//
// // Generate DPRs for each tile in each cluster period in production request time frame
//
// time = PlTime::ClosestBelow(myBoundary, myPeriod)
// while time < PlProductionRequest::myDataCollectionStopTime {
//
//   for myNumClusters {
```

```
//
//     PlCluster(myNumTiles, myLastOrbit)
//
//     // Calculate predicted availability of all L0 data granules for cluster
//     L0PredictedAvailability = time + PlCluster::myLastOrbit * OrbitDuration
//
//     // Calculate predicted availability of tiles for cluster
//     PlCluster::myPredictedAvailability = L0PredictedAvailability + TransitTime
//       + ProductionTime +  myDelayFactor
//
//      for each tile in cluster loop {
//
//        PlDPR()
//
//        if PlDataType::mySpatialFlag == TRUE {
//
//          // construct input granule placeholder - will be replaced with actual
//          // granules by subscription manager when it receives them.
//          PlDatagranule()
//          PlDatagranule.mySpatialFlag = TRUE
//            PlDPR.myPredictedStartTime = PlCluster::myPredictedAvailability
//          PlDataGranule.myPredictedAvailability = PlCluster::myPredictedAvailability
//          PlDatagranule() // output granule: set PlDatagranule::myTileID = tile number
//
//        }
//
//      }
//
//   } // end for myNumClusters
//
//   // process next period
//   time += myPeriod
//   time += PlTime::ClosestAbove(myBoundary, myPeriod)
//
// } // end while
```

**PlTileScheduledNB** - This is the default constructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**PlTileScheduledNB** - This constructor takes in L0Type, PredictionDelta, ProductionDelta, and Definition and creates an object of the class, giving the corresponding attributes those values.
Arguments:ESDT L0Type, RWTime PredictionDelta, RWTime ProductionDelta,

RWCString Definition

**ReturnL0Type -** Returns myL0DataType.
Arguments:
Return Type:ESDT
Privilege:Public

**ReturnProductionDelta -** Returns myProductionDeltaTime.
Arguments:
Return Type:RWTime
Privilege:Public

**~PlTileScheduledNB -** This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**
The PlTileScheduledNB class has associations with the following classes:
Class: PlCluster
Class: PlSourcetoDsHistoryNB references

### 4.3.114 PlTimeLineDisplay Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Purpose and Description:
This class describes the user interface component that represents the graphical display of a plan. This will be implemented by a COTS or re-use component.

**Attributes:**

**myPlansDisplayed -** List of plans currently being displayed on the timeline
Data Type:RWCollectableString<PlPlanId>
Privilege:Private
Default Value:

**Operations:**

**DisplayPlan -** Abstract operation for displaying a plan on a timeline
Arguments:PlPlan &
Return Type:void
Privilege:Public

PDL:Display activities in plan on timeline (Delphi abstract operation)

**Associations:**

The PlTimeLineDisplay class has associations with the following classes:
PlPlanningWorkbenchUI (Aggregation)

### 4.3.115 PlTimeScheduled Class

Parent Class:PlPGE
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This specialization of the PGE accounts for the classification of PGEs whereby the PGE scheduling is determined from some regular time period, such as an hour, a day, a week, a month etc.

**Attributes:**

**myAcquisitionProcessingBoundary** - Identifies the time needed (for data) for extrapolation purposes.
Data Type:Time
Privilege:Private
Default Value:

**myAcquisitionProcessingPeriod** - Identifies the acquisition period for the input data.
Data Type:Time
Privilege:Private
Default Value:

**myNoOfTimeUnitsPerPgeRun** - Number of Units of time for PGE run.
Data Type:float
Privilege:Private
Default Value:

**myTimeUnits** - Units of time for PGE run frequency.
Data Type:enum
Privilege:Private
Default Value:

**Operations:**

**GenerateDPRs** - { // Iterate over the period of the production request in steps // of myAquisitionProcessingPeriod from the nearest occurance // of the myAcquisitionProcessingBoundary within the request period // Generate a DPR // Invoke the select data method for the DPR }
Arguments:PlProductionRequest
Return Type:void
Privilege:Public

**PlTimeScheduled** - This constructor takes in the PGE ID of the Time Scheduled class to create.
Arguments:RWCString:PgeId
Return Type:PlTimeScheduled
Privilege:Public

**~PlTimeScheduled** - This is the destructor for the class.
Arguments:
Return Type:Void
Privilege:Public

**Associations:**

The PlTimeScheduled class has associations with the following classes:
None

### 4.3.116 PlTimer Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class initiates a timer to alert the subscription manager of the expiration of a wait time, for alternat input activation.

**Attributes:**

**MyTimerExp** - This attribute indicates whether or not the time has expired.
Data Type:EcTBoolean
Privilege:Private
Default Value:

**MyTimerStart** - This attribute indicates whether a timer has started for a DPR.
Data Type:EcTBoolean
Privilege:Private

Default Value:

**Operations:**

**CancelTimer** - This operation cancels the timer for a specified DPR.
Arguments:
Return Type:EcUtStatus
Privilege:Public

**StartTimer** - This operation starts the timer for a specified DPR.
Arguments:RWTime<myTimerNB>
Return Type:EcUtStatus
Privilege:Public
PDL:{

```
  Based on PlAlternateDataGranule:myOrder and the priority ordering of inputs
  For all AlterateDataGranules with PlAlternateDataGranule:myTimerNB <> 0
  {
    Start the timer based on the sequence of ordering.  This will
    implemented using Pthreads.  This processing will resume once the
    specified time has elapsed
    Start(PlAlternateDataGranule:myTimerNB)
    If the wait time has elapsed call check availability
    if (WaitHasElapsed)
    {
      PLDPRB:: CheckAvailability()
       Check to see if there are other timers available if all data is not available.
       if (PlAlternateDataGranule:myTimerNB = 0 .and. PLDPRB::CheckAvailability() =
false)
      {
        Check if either the minimum primaries or backups are available
        if (TotPrim <= PlDataTypeReq:myNumNeeded  and
          TotBackup <= PlDataTypeReq:myNumNeeded)
        {
          Determine if we need to wait for additional input if not available.
          if (PlAlternateDataGranule:myWaitfor = True)
          {
            return false
          }
        }
        Change the priority ordering of the data granules
        PlAlternateDataGranule:ChangeOrder()
        Modify the input list to reflect the data types being used
        PlDPRB::Modify()
        Release job to processing
```

```
            Release()
            return
          }
          Release job and return if the required input is available
          if(PLDPRB::CheckAvailability() = true)
          {
             Release()
             return
          }
          Otherwise, loop to get next timer
        }
      }
      Set TimeExp to true
    }
```

**Associations:**

The PlTimer class has associations with the following classes:
Class: PlDPRB isusedtosettimeronalternateinputs

### 4.3.117 PlUserParameters Class

Parent Class:Not Applicable
Public:Yes
Distributed Object:No
Persistent Class:True
Purpose and Description:
Describes any user defined parameters that are associated to a PGE.

**Attributes:**

**myDefaultValue** - The default value for the user parameter.
Data Type:String
Privilege:Private
Default Value:

**myDescription** - Describes the user parameter
Data Type:String
Privilege:Private
Default Value:

**myLogicalID** - Id of the user parameter
Data Type:int

Privilege:Private
Default Value:

**myName** - Name of the user parameter
Data Type:String
Privilege:Private
Default Value:

**Operations:**

**Create** - Create an entry within the PDPS database
Arguments:
Return Type:void
Privilege:Public

**Delete** - Delete the entry from the PDPS database
Arguments:
Return Type:void
Privilege:Public

**Modify** - Modifies the default value of the user parameter
Arguments:DefValue:String
Return Type:void
Privilege:Public

**PlUserParameters** - Constructor method
Arguments:
Return Type:PlUserParameters
Privilege:Public

**~PlUserParameters** - Destructor method
Arguments:
Return Type:void
Privilege:Public

**Associations:**

The PlUserParameters class has associations with the following classes:
Class: PlPGE isreferencedby
PlPGEProfile (Aggregation)

### 4.3.118 PlUserPriorityNB Class

Parent Class:Not Applicable
Public:No
Distributed Object:No
Persistent Class:True
Purpose and Description:
This class is part of the production strategies which relates a particular user to a particular priority. It can be used to determine the priority of jobs submitted by that user.

**Attributes:**

**myPriority** - This attribute is a part of the production strategy that relates a priority to a particular user.
Data Type:EcTInt
Privilege:Private
Default Value:

**myUserType** - This attribute is a part of the production strategy which relates a user name to a priority.
Data Type:RWCString
Privilege:Private
Default Value:

**Operations:**

**GetPriority** - This operation returns a priority based on the user type of the DPR.
Arguments:PlDPR &
Return Type:Void
Privilege:Public
PDL:{
Search database table for UserType = PlDPR::GetPR::GetUserType()
return PRIORITY from table associated with that User Type
if no priority in table for this user type, return default priority
}

**Associations:**
The PlUserPriorityNB class has associations with the following classes:
PlProdStratNB (Aggregation)

## 4.4 PLANG Dynamic Model

The PLANG Dynamic model presents a number of scenarios and event traces that describe the key interactions of the classes participating in the various components of the Planning subsystem.

### 4.4.1 Production Request Scenario

This scenario describes various operations related to entering production requests. The production request is entered by the Production Planner. The Production Planner has the option of entering a particular PGE to be used during production. If no PGE is specified Planning will determine the default PGE required. This scenario describes the process by which a production request is translated to multiple data processing requests.

When a routine production request is entered, the input data granules needed to fulfill the request may already be available to ECS, or they may be available in the future. Availability of input data depends on data collection times entered in the production request with respect to when the request is entered. The procedure by which the availability of data is determined is described in scenario 4.4.2.

#### 4.4.1.1 Beginning Assumptions

None

#### 4.4.1.2 Interfaces With Other Subsystems and Segments

Routine Production Requests are entered / modified / deleted by the Production Planner (human). These requests are handled by PlProductionRequestUI class. Necessary GUI will be developed to enter these requests.

#### 4.4.1.3 Stimulus

Routine Production request is entered / modified / deleted by the Production Planner.

#### 4.4.1.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningUserUI
- PlProductionRequest
- PlDataTypeCatalogue
- PlPGEFactory
- PlDPRB
- PlTimeScheduledNB

#### 4.4.1.5 Beginning System, Segment and Subsystem State(s)

PDPS Database is in a steady state, up and running.

#### 4.4.1.6 Ending State

PDPS Database is in steady state, up and running.

#### 4.4.1.7 Scenario Description

***Thread 1: Routine Production request entry***

1. Using the Production Request Editor, the Production Planner will enter the command for initiating the entry of a Routine Production request. This command is received by PlProductionRequestUI class. Alternatively, the Planner can enter a Production Request Collection - a group of related PR's. This is useful especially for entering large reprocessing requests.
2. The Production Planner will ask to display a list of products that can be generated. The list can be filtered. Additionally, the Planner can elect to display dependent products at the next higher level when creating collections (e.g. display Level 3 products dependent on a selected Level 2 product).
3. PlProductionRequestUI class will use the PlDataTypeCatalogue class to get the list of products (product catalogue). For each product, the corresponding PGE and input data types are also requested.
4. PlDataTypeCatalogue class will obtain the required information and send it to PlProductionRequestUI class.
5. PlProductionRequestUI class will display the product catalogue.
6. The Production Planner will make the desired selection of the product. At that time, the default PGE and any default user parameters for that PGE are displayed. The Production Planner enters the routine production request along with (1) specific user parameters, or (2) any optional PGE and user parameters associated with that PGE. This request will also have the data collection start and end periods for product generation. Typically the data collection start and end periods refer to one of the input data types for that PGE, and this input type is represented by the PlDataType class.
7. The request is checked for validity. For example, data collection start time should be ahead of data collection end time. If not, the routine production request will be rejected with a warning.
8. The PlProductionRequest class uses the PlPGEFactory class to generate the PGE based on the PGE's id.
9. PlProductionRequest class uses the PlPGE class to generate instances of PlDataGranule class and PlDPRB class.
10. PlPGE class generates PlDataGranule objects. Note: This is described in scenario 4.4.2.
11. PlPGE class has information on how the input data instances map to Data Processing Requests. Note: PlPGE class has specialization classes - PlDataScheduled, PlTimeScheduled, PlOrbitScheduledNB, PlTileScheduledNB, etc. which describe how often the given PGE needs to be run (refer to Figure 4.2-1, PGE Profile View for details). PGE creates many instances of PlDPRB objects based on this information.
12. The production request is stored in the PDPS database.
13. When creating collections, steps 4-11 are repeated until the Planner determines the collection is complete. In addition, a report can be generated that shows the resource utilization cost for the entire Production Request Collection.

***Sub-Thread 1-1: Orbit and Tile-Scheduled Routine Requests***

14. For the case where the PlPGE specialization is PlOrbitScheduledNB, the PlOrbitModel is used to determine the estimated start and stop times of the orbit. For the PlTileScheduledNB case, the tile definition will be determined. In either case, the number of PlDPRB objects will be based on this.

***Thread 2: Routine Production request modification***

1. Using the Production Request Editor, the Production Planner will enter the command for initiating a Routine Production request modification. This command is received by PlProductionRequestUI class.
2. PlProductionRequestUI class will send a request to PlProductionRequest class to send a list of all current Routine Production requests.
3. PlProductionRequest class will gather the requested list (through Planning Database queries), and send the list to PlProductionRequestUI class. PlProductionRequestUI class will display the list to the Production Planner.
4. The Production Planner will modify the intended field on the intended production request (which can be data collection start and end times, or PGEs).
5. The entered command is checked for validity. For example, data collection start time should be ahead of data collection end time (If not, the routine production request modification will be rejected with a warning).
6. The valid command is sent to PlProductionRequest class.

Note: PlProductionRequest class has the pointers to PlDataGranule and PlDPRB objects corresponding to the Production Request.

***Sub-Thread 2-1: Data collection start/end time/both is/are modified***

7. PlProductionRequest class compares original data collection period with modified data collection period.
8. Any new data collection period is treated as if a new routine production request is entered (refer to Thread 1: Routine Production request entry for this case).

***Sub-Thread 2-2: PGE is modified***

9. When a PGE is modified, the PlPGE object corresponding to the new PGE is determined. With the new PGE, it is possible that some of the input data types may change.
10. If there is no change in any of the input data types, PlDPRB objects corresponding to the production request are modified to indicate the new PlPGE class.
11. If there is change in any of the input data types, PlDataGranules corresponding to the old input data type of the old production request are deleted. New input data types are treated as if a new routine production request is entered (refer to Thread 1: Routine Production request entry for this case), which will result in new instances of PlDataGranule corresponding to the new input data type of the modified production request. PlDPRB objects corresponding to the production request are modified to indicate the new PlPGE class, and new PlDataGranule classes.

***Thread 3: Routine Production request deletion***

1. Using the Production Request Editor (Routine Production request deletion window), the Production Planner will enter the command for initiating a Routine Production request deletion. This command is received by PlProductionRequestUI class.
2. PlProductionRequestUI class will send a request to PlProductionRequest class to send a list of all current Routine Production requests.
3. PlProductionRequest class will gather the requested list (through Planning Database queries), and send the list to PlProductionRequestUI class. PlProductionRequestUI class will display the list to the Production Planner.
4. The Production Planner will delete the intended production request.
5. The deletion command is sent to PlProductionRequest class.
6. PlProductionRequest class deletes PlDataGranule and PlDPRB objects corresponding to the production request as long as the PlDataGranule and PlDPRB objects correspond to an another production request, in which case they are not deleted.
7. PlProductionRequest object is deleted.

#### 4.4.1.8 Event Trace

See Figure 4.4-1.

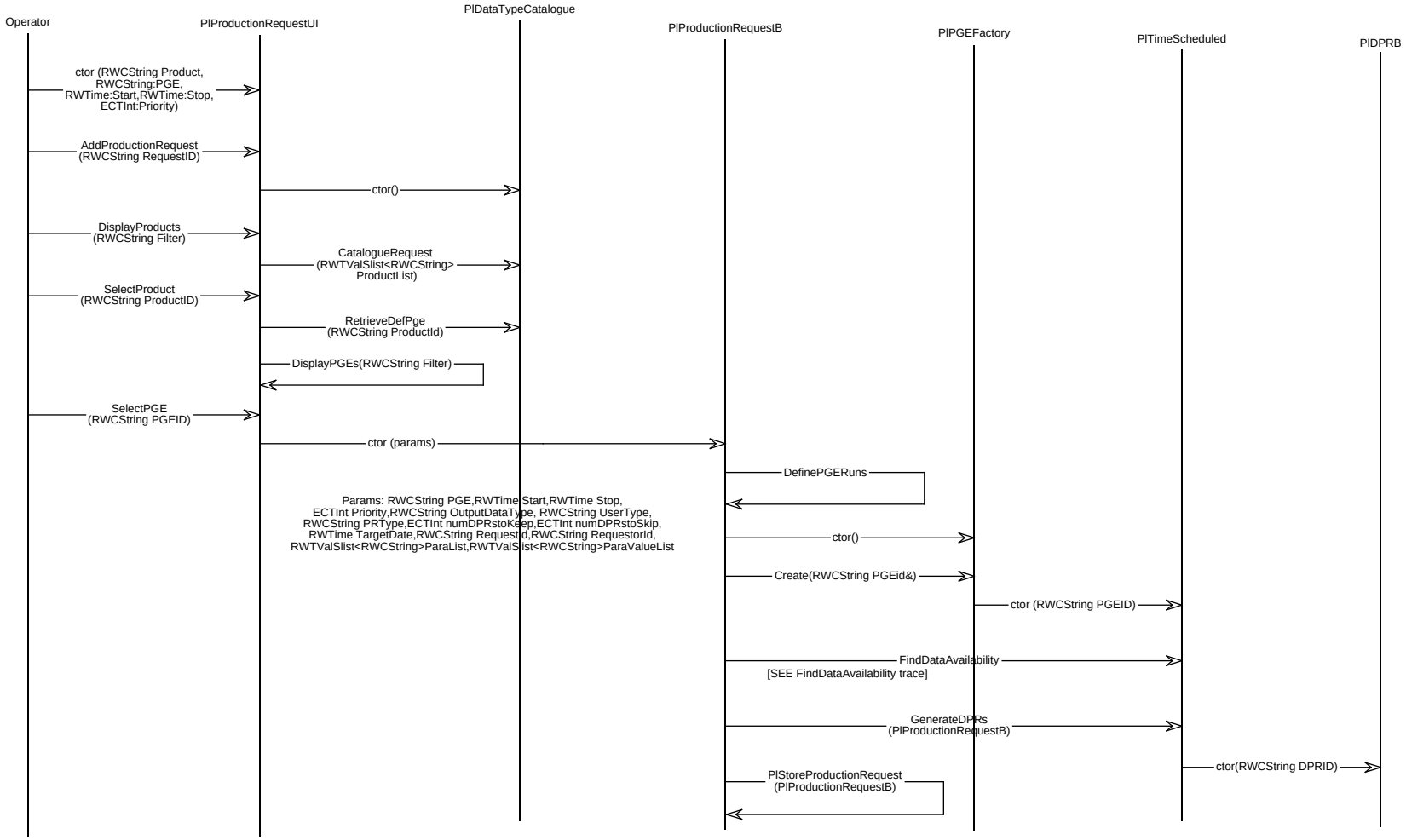



***Figure 4.4-1. Production Request Scenario***

### 4.4.2 Data Availability Scenario

#### 4.4.2.1 Abstract

This scenario describes the methods by which the data available for a Production Request are determined. Data may be already available within the PDPS database or the Data Server (in the case of reprocessing or processing of historic data); otherwise a prediction of the arriving data is generated. The prediction can be based on a Planning Data Availability Schedule (PDAS) or heuristics. If the planning subsystem is not currently subscribed to data that is not currently available at the Data Server, an automatic subscription is submitted. This scenario describes a sequence of events that occur in the explosion of a Production Request into Data Processing Requests, as described in Scenario 4.4.1.

#### 4.4.2.2 Interfaces With Other Subsystems and Segments

The Science Data Server is queried to determine the availability of data.

#### 4.4.2.3 Stimulus

This scenario describes the procedures by which the data available for a Production Request are determined. The overall stimulus is the receipt of a new production request. This is described in scenario 4.4.2.

#### 4.4.2.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlProductionRequest
- PlPGE
- PlDataTypeReq
- PlOutputYield
- PlDataType
- PlDataGranule
- PlDataSource
- PlDataSourceFactory
- PlDATRecord
- PlSubscriptionSubmitIF
- PlMetaDataChecks

#### 4.4.2.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

#### 4.4.2.6 Ending State

No change in PDPS database server state.

#### 4.4.2.7 Scenario Description

1. The PlProductionRequest class invokes the FindDataAvailability method of the PGE in order to determine the availability of the data required for that production request, using the start and stop time of the Production Request.
2. The PGE iterates through each of the input data types for that PGE, and invokes the FindDataAvailability for that data type.

***Sub-Thread 1-1 - Search local database***

3. Withinin the FindDataAvailability method, the PDPS Database is searched for the existence of the data.
4. For each data granule found, metadata checks (e.q. for QA) are performed.

***Sub-Thread 1-2 - Search Data Server***

5. Within the FindDataAvailability method it is determined whether any of the period of interest is in the past, if so the Data Server which provides for the archive of that Data Type is searched for existence of the Data.
6. A query and an inspect are constructed and applied against the Data Server (see the DID 313 scenario primitive for the Data Server query and inspect processes for details).

***Sub-Thread 1-2-1 - Data server finds all the input data***

7. The results set returned from the query is iterated through, extracting the required metadata that describes the instance. If there are metadata conditions associated with the instance, the conditions are checked by the CheckforCondition method within the PlMetaDataChecks class. If the conditions are met, an instance of the PlDataGranule for each instance is created, ensuring that instance is not a duplicate already within the table.

***Sub-Thread 1-2-2 - Data server doesn't find some of the data***

8. If part or all of the data isn't found, then the Data Availability Time records (DAT) are queried to determine the time the data will be produced by a remote DAAC. These records are returned for the collection time frame of the production request. Metadata checks are applied.

***Sub-Thread 1-3 - Data Predictions***

9. If all or part of the input data for the production request is to be collected in the future, then the FindDataAvailability method predicts the data arrivals using the PlRoutineArrival subclass of the PlDataSource class. PlRoutineArrival is used for data with known arrival patterns.
10. These input granules are then created based on these predictions, and marked as predicted granules.
11. For each PlDatatype that is not currently subscribed to and that has granules that are not currently available, the SubmitSubscription method of PlDataSource is called. This method submits a subscription using the PlSubScriptionSubmitIF to advertising.

#### 4.4.2.8 Event Trace

See Figure 4.4-2.

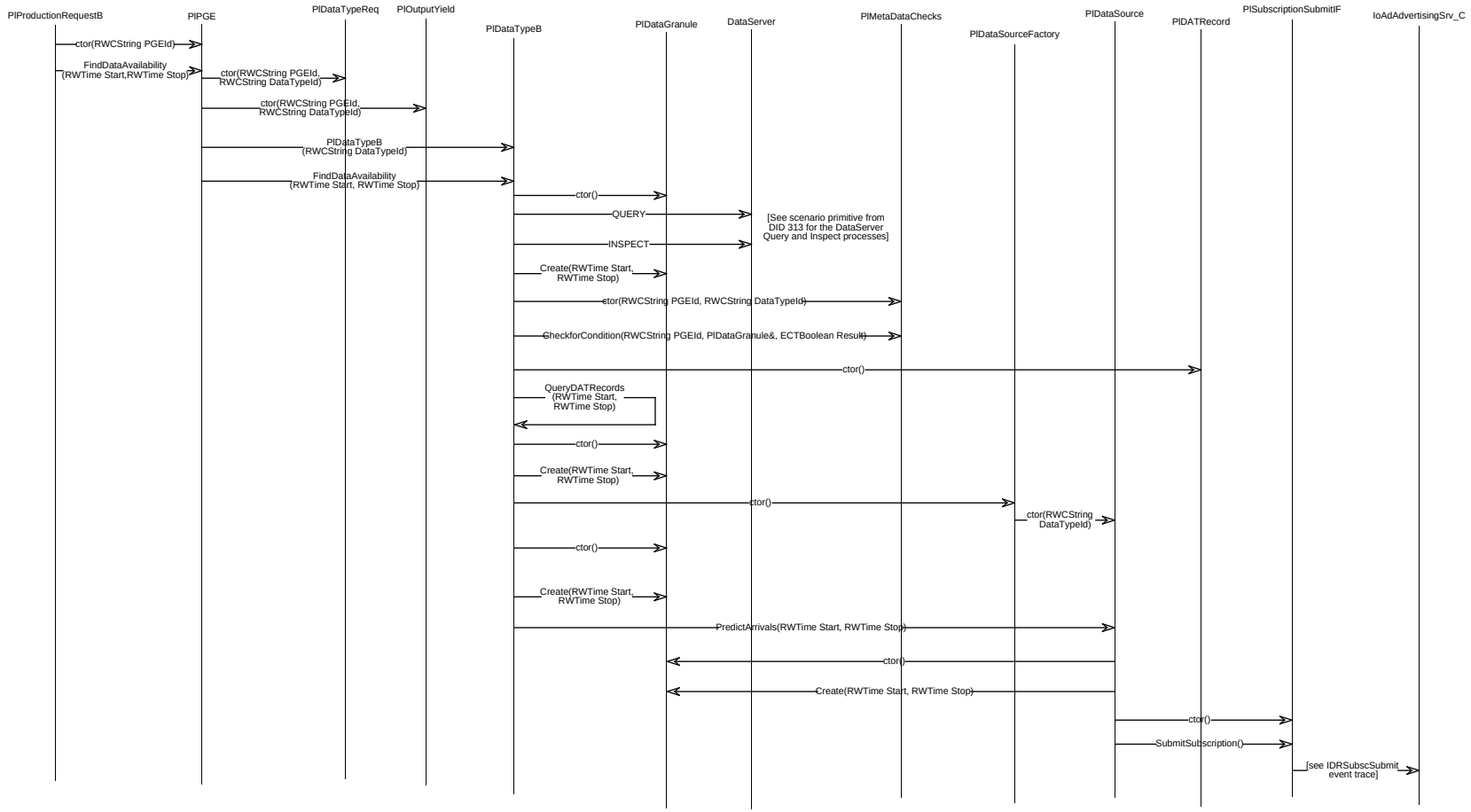



***Figure 4.4-2. Data Availability Scenario***

### 4.4.3 Subscription Submission Scenario

#### 4.4.3.1 Abstract

This scenario describes submitting a subscription to a Data Server from the Subscription submission utility.

#### 4.4.3.2 Interfaces With Other Subsystems and Segments

The Interoperability subsystem provides advertisements to the Planning subsystem. The subscription is submitted to the Data Server subsystem.

#### 4.4.3.3 Stimulus

The Production Planner initiates the Subscription Submission Interface in order to submit a subscription.

#### 4.4.3.4 Participating Classes From the Object Model

**The following are participating classes from the Object Model:**

- PlSubscriptionSubmitIF
- PlDataType
- IoAdProductSearchCommand
- IoAdProductList
- IoAdService

#### 4.4.3.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

#### 4.4.3.6 Ending State

No change in PDPS database server state. The subscription is submitted.

#### 4.4.3.7 Scenario Description

1. The user starts the subscription submission utility, the standard user authentication process applies (see scenario 4.4.18). The user selects to set up a subscription and is offered a display of those Data Types which may be subscribed to.
2. The user selects the Data Type for which to set up a subscription.
3. The subscription submission utility selects the advertisement corresponding for the selected Data Type, using the Data Server UR and product name attributes of the Data Type class.
4. The user confirms the submission of the subscription.
5. The utility builds the Data Server subscription from the advertisement, submits the subscription and, on a successful status from the Data Server updates the Data Type object within the PDPS database.

### 4.4.3.8 Event Trace

See Figure 4.4-3.

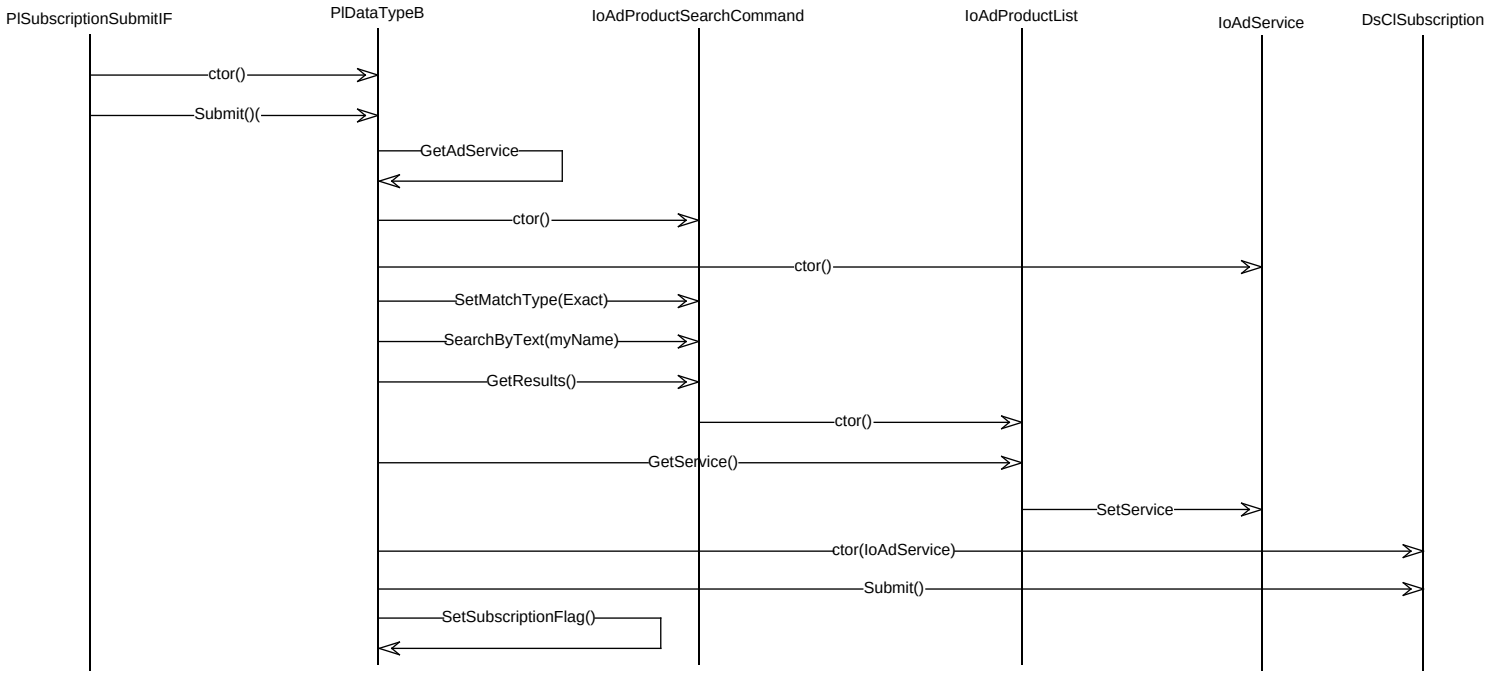



***Figure 4.4-3. Subscription Submission Scenario***

### 4.4.4 Subscription Withdrawal Scenario

#### 4.4.4.1 Abstract

This scenario describes withdrawing a subscription to a Data Server from the Subscription submission interface.

#### 4.4.4.2 Interfaces With Other Subsystems and Segments

The Interoperability subsystem provides advertisements to the Planning subsystem. The subscription is submitted to the Data Server subsystem.

#### 4.4.4.3 Stimulus

The Production Planner initiates the Subscription Submission Interface in order to withdraw a subscription.

#### 4.4.4.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlSubscriptionSubmitIF
- PlDataType
- IoAdProductSearchCommand
- IoAdProductList
- IoAdService

#### 4.4.4.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

#### 4.4.4.6 Ending State

No change in PDPS database server state. The subscription is withdrawn.

#### 4.4.4.7 Scenario Description

1. The user starts the subscription submission utility, the standard user authentication process applies (see scenario 4.4.18). The user selects to withdraw a subscription and is offered a display of those Data Types which are subscribed to.
2. The user selects the Data Type for which to withdraw a subscription.
3. The subscription submission utility selects the advertisement corresponding for the selected Data Type, using the Data Server UR and product name attributes of the Data Type class.
4. The user confirms the withdrawal of the subscription.
5. The utility builds the Data Server subscription from the advertisement, withdraws the subscription and, on a successful status from the Data Server updates the Data Type object within the PDPS database.

#### 4.4.4.8 Event Trace

See Figure 4.4-4.

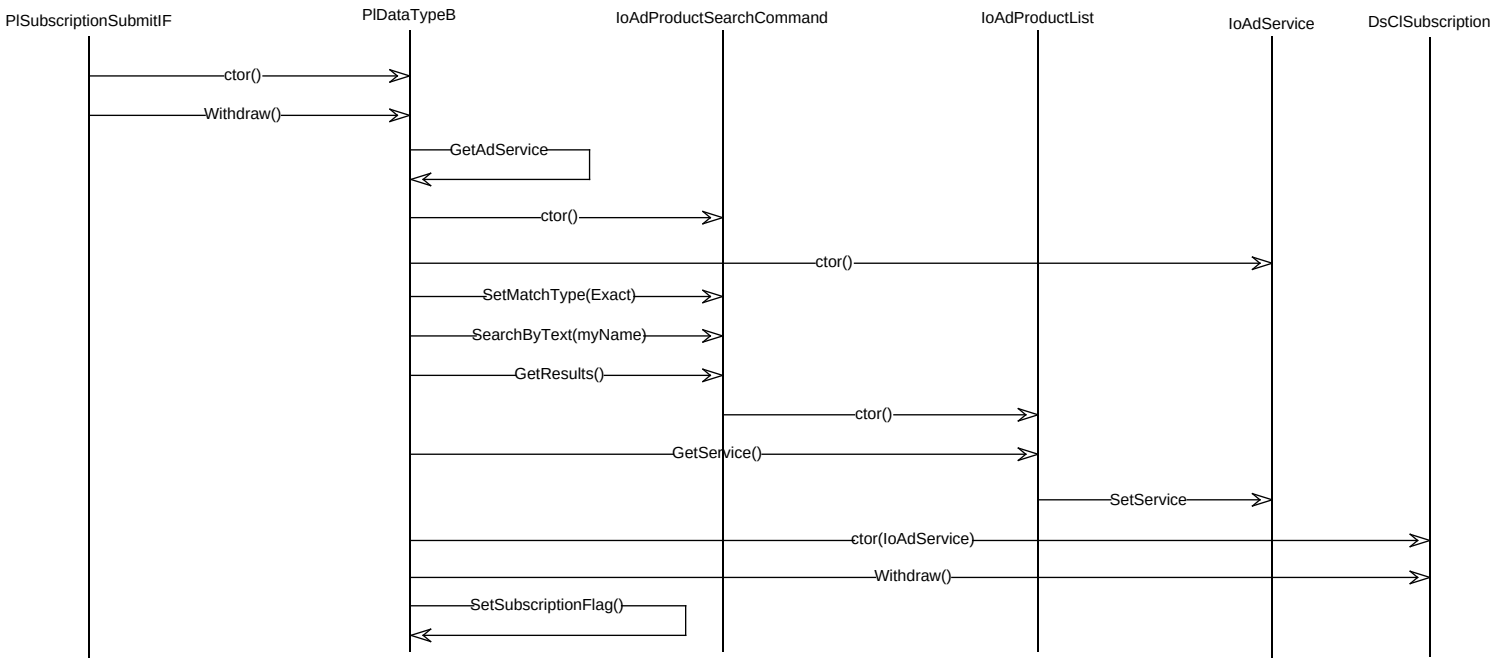



***Figure 4.4-4. Subscription Withdrawal Scenario***

### 4.4.5 Plan Creation Scenario

#### 4.4.5.1 Abstract

This scenario describes the creation of a plan within the Production Planning Workbench.

This scenario presents an abstract representation of the activities that occur within the Production Planning Object Library. The full detail of generating a plan is very complex, and intimately tied to the Production Planning Object Library. This scenario is presented to describe the process at a reasonable level of detail. For fuller description please refer to the Production Planning Object Library CSC section (Section 4.5.5).

#### 4.4.5.2 Interfaces With Other Subsystems and Segments

None.

#### 4.4.5.3 Stimulus

The production planner initiates the Production Planning Workbench in order to generate a new plan.

#### 4.4.5.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlGroundEvents
- PlGroundEvent
- PlGroundActivity
- PlDPRs
- PlDPRB
- PlPGEActivity
- PlResourceManager
- PlProdStratNB

#### 4.4.5.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

#### 4.4.5.6 Ending State

No change to system state.

#### 4.4.5.7 Scenario Description

1. The user starts the planning workbench utility, the standard user authentication process applies (see scenario 4.4.18). The user specifies the time period for which to generate a plan within the GUI and initiates the creation of a plan object.

2. The plan object determines which ground events are defined within the duration of the planning time-period, by creating an instance of the PlGroundEvents collection object.
3. The plan object iterates through the ordered list of ground events, creating an activity to fulfill each event
4. The activity is allocated to the appropriate resources by the resource manager, thus setting out a timeline of when the resources are unavailable for production.
5. The user then specifies the Production Requests which are to be included in the plan. This would include reprocessing requests, standard requests, and any deferred on-demand requests.
6. The PlDPRs collection class is used to determine which Data Processing Requests should be included in the plan. All DPRs associated to the Production Requests specified above are searched. A DPR is selected if all it's input data is predicted to be available within the time frame of the plan.
7. The plan object iterates through the ordered list of Data Processing Requests, creating an activity to signify the run of the PGE.
8. Activities are created as place holders for on-demand production requests. These activities will be dispersed throughout the planned day to reserve resources and maintain schedule time frames when on-demand requests are entered.
9. Activities are assigned a priority using the production strategies listed in the PlProdStrat class. See scenario 4.4.6, Assigning a Priority to an Activity for more details.
10. The activity is allocated to the appropriate resources by the resource manager.
11. The schedule for the PGE executions is redetermined by the plan object, defining the predicted start and predicted stop times of all the activities.

#### 4.4.5.8 Event Trace

See Figure 4.4-5

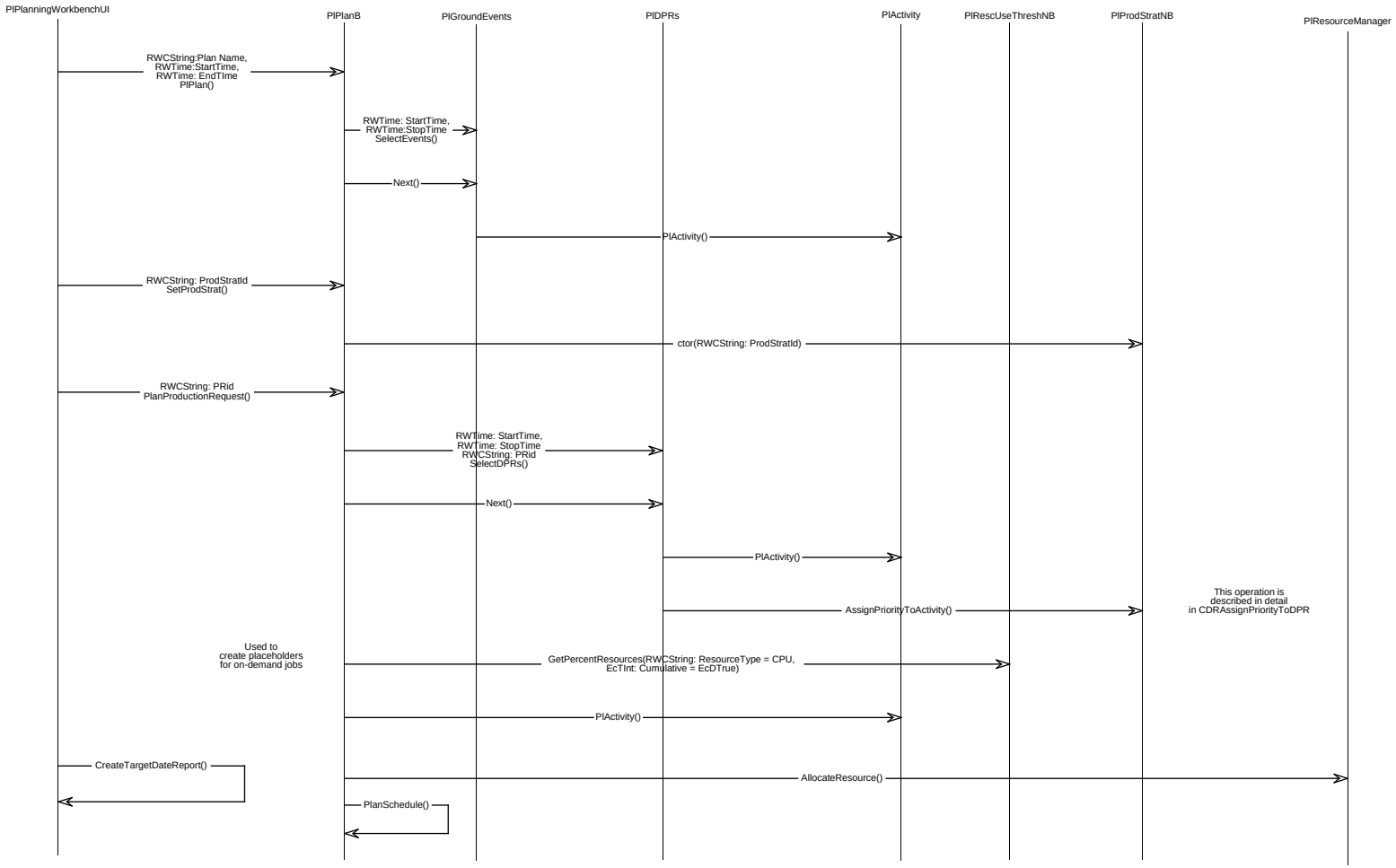



***Figure 4.4-5. Plan Creation Scenario***

### 4.4.6 Assigning a Priority to an Activity Scenario

#### 4.4.6.1 Abstract

This scenario describes how the Production Planning Workbench assigns a priority to an activity in a plan based on the production strategies associated with that plan. Production strategies allow the operator to define priorities at a high level based on the values of selected attributes of each Data Processing Request (DPR) such as PGE, Production Request type or User Type. In addition, an operator selected addition can be made to the calculated priority based on a few conditions such as whether or not this DPR produces data needed at a remote DAAC or if this DPR has been waiting for execution for more than one day.

#### 4.4.6.2 Interfaces With Other Subsystems and Segments

None.

#### 4.4.6.3 Stimulus

During the creation of a candidate plan, the Production Planning Workbench assigns a priority to the DPRs within that plan before it begins to schedule these jobs.

#### 4.4.6.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlan
- PlActivity
- PlDPRB
- PlProdStrat
- PlPGEPriority
- PlPRPriority
- PlUserPriority
- PlProductionRequest

#### 4.4.6.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. This scenario shows the details of step 9 of Section 4.4.5, the plan creation scenario.

#### 4.4.6.6 Ending State

The activities in a plan are assigned priorities and the Planning Workbench can then begin scheduling them.

#### 4.4.6.7 Scenario Description

1. The Plan object selects all the Data Processing Requests for the plan to be scheduled and creates an activity for each one.

2. The DPR for that activity is compared with the Production Strategy tables and a priority is returned for each table attribute: the PGE, the Production Request type, and the user type associated with the DPR.
3. A calculation of a base priority is done using the values from the tables and the weights associated with each value. The priority entered by the operator for this Production Request is also figured into this calculation.
4. If the DPR has been waiting in the Processing Queue for more than 1 day, an operator defined additional amount is added to the priority.
5 If the DPR produces data that is needed at a remote DAAC, another operator defined additional amount is added to the priority.

#### 4.4.6.8 Event Trace

See Figure 4.4-6.

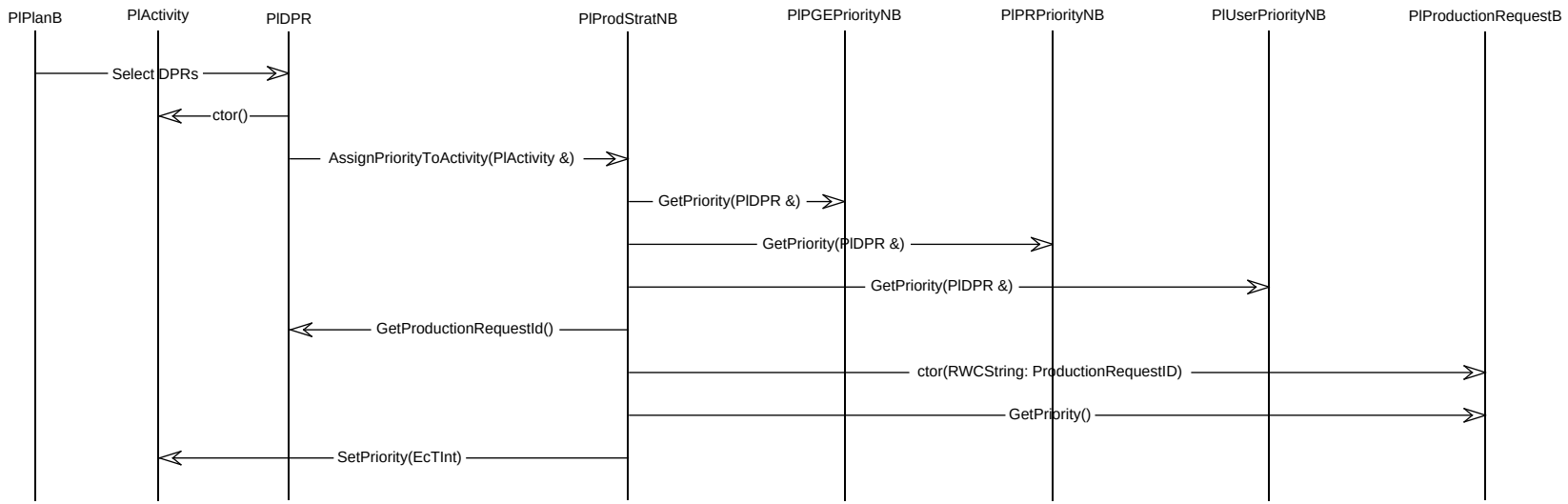



***Figure 4.4-6. Assigning a Priority to an Activity Scenario***

### 4.4.7 Publishing a Plan Scenario

#### 4.4.7.1 Abstract

This scenario describes the system response to a production planner publishing a plan which is maintained within the PDPS database. After being selected, the plan would be formatted by the system into a published plan consisting of three different types of file, which are Metadata file, ASCII report file, Binary report file and an exported plan that will enable other DAACs and the SMC to view the plan at the detailed level required for inter-DAAC planning. The published plan then would be inserted into Documentation Data Server. The method to insert a published plan into Document Data Server is an operation encapsulated in the PlPublishedPlan Class.

#### 4.4.7.2 Interfaces With Other Subsystems and Segments

Document Data Server provides storage for published plans. The interface to the Document Data Server is shown by the DsCtProductionPlan object, which is a Document Data Server abstraction for the plans.

#### 4.4.7.3 Stimulus

A production planner initiates PublishPlan Function from the Production Planning Workbench.

#### 4.4.7.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlPublishedPlan
- PlPlanMetadataFile
- PlPlanASCIIReportFile
- PlPlanBinaryReportFile
- PlPlanExportedPlan

#### 4.4.7.5 Beginning System, Segment and Subsystem State(s)

Steady state, up and running.

#### 4.4.7.6 Ending State

No change in the state of the system.

#### 4.4.7.7 Scenario Description

1. The production planner starts the planning workbench utility, the standard user authentication process applies (see scenario 4.4.18).
2. The production planner is presented with a function to publish a plan, and initiates the function.
3. Published Plan object is created.

4. The published plan is inserted into Document Data Server.
5. Subscribers to these plans will be notified via the normal subscription service.

#### 4.4.7.8 Event Trace

See Figure 4.4-7.

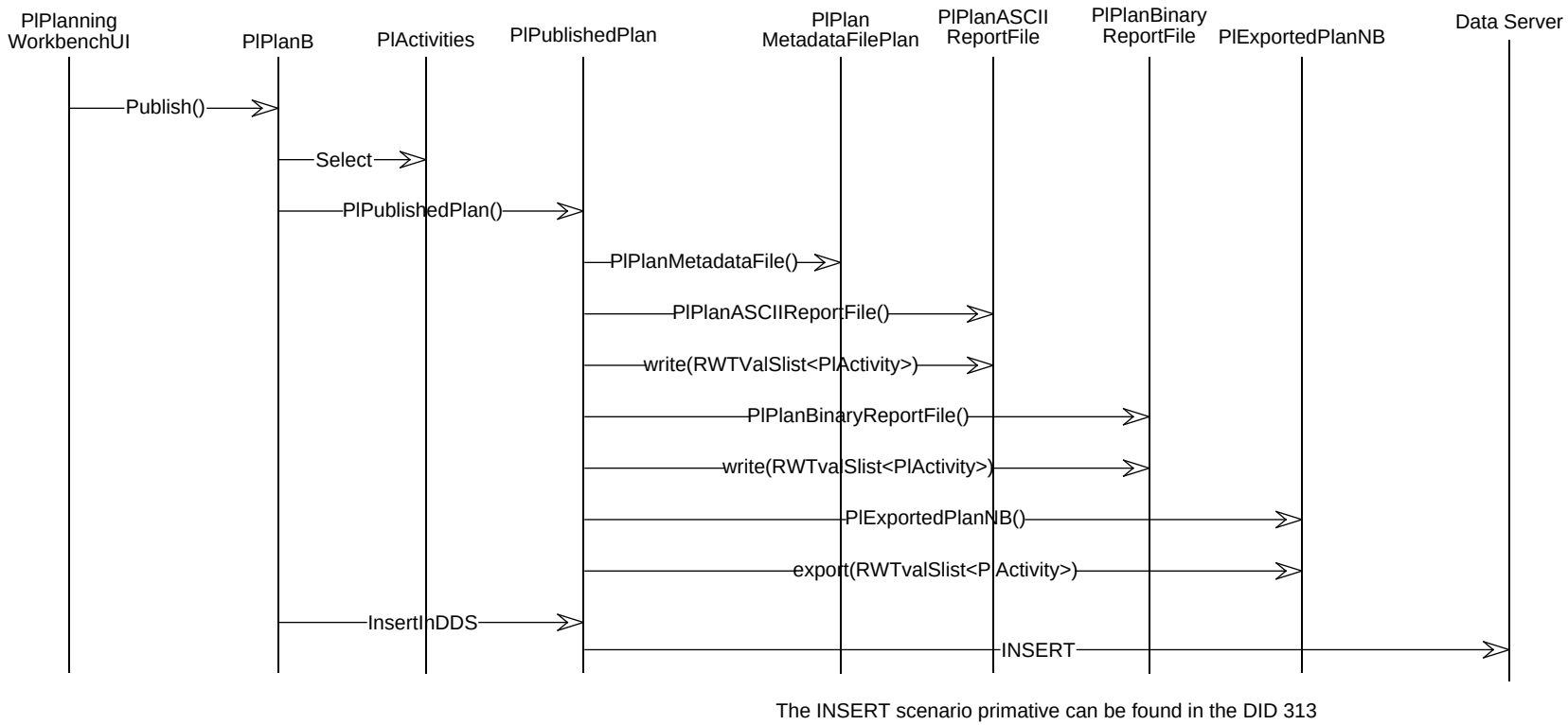



***Figure 4.4-7. Publishing a Plan Scenario***

### 4.4.8 Importing a Plan from the Data Server Scenario

#### 4.4.8.1 Abstract

This scenario describes the process of querying other DAAC's plans, importing plans from remote DAACs and identifying any data dependencies. This scenario presents an abstract representation of the activities that occur within the Production Planning Workbench. This scenario is presented to describe the process at a reasonable level of detail.

#### 4.4.8.2 Interfaces With Other Subsystems and Segments

Data Management, Data Server.

#### 4.4.8.3 Stimulus

The production planner initiates the Production Planning Workbench in order to compare DAAC plans.

#### 4.4.8.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlActivity
- PlPublishedPlan
- PlExportedPlanNB

#### 4.4.8.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

#### 4.4.8.6 Ending State

No change to system state.

#### 4.4.8.7 Scenario Description

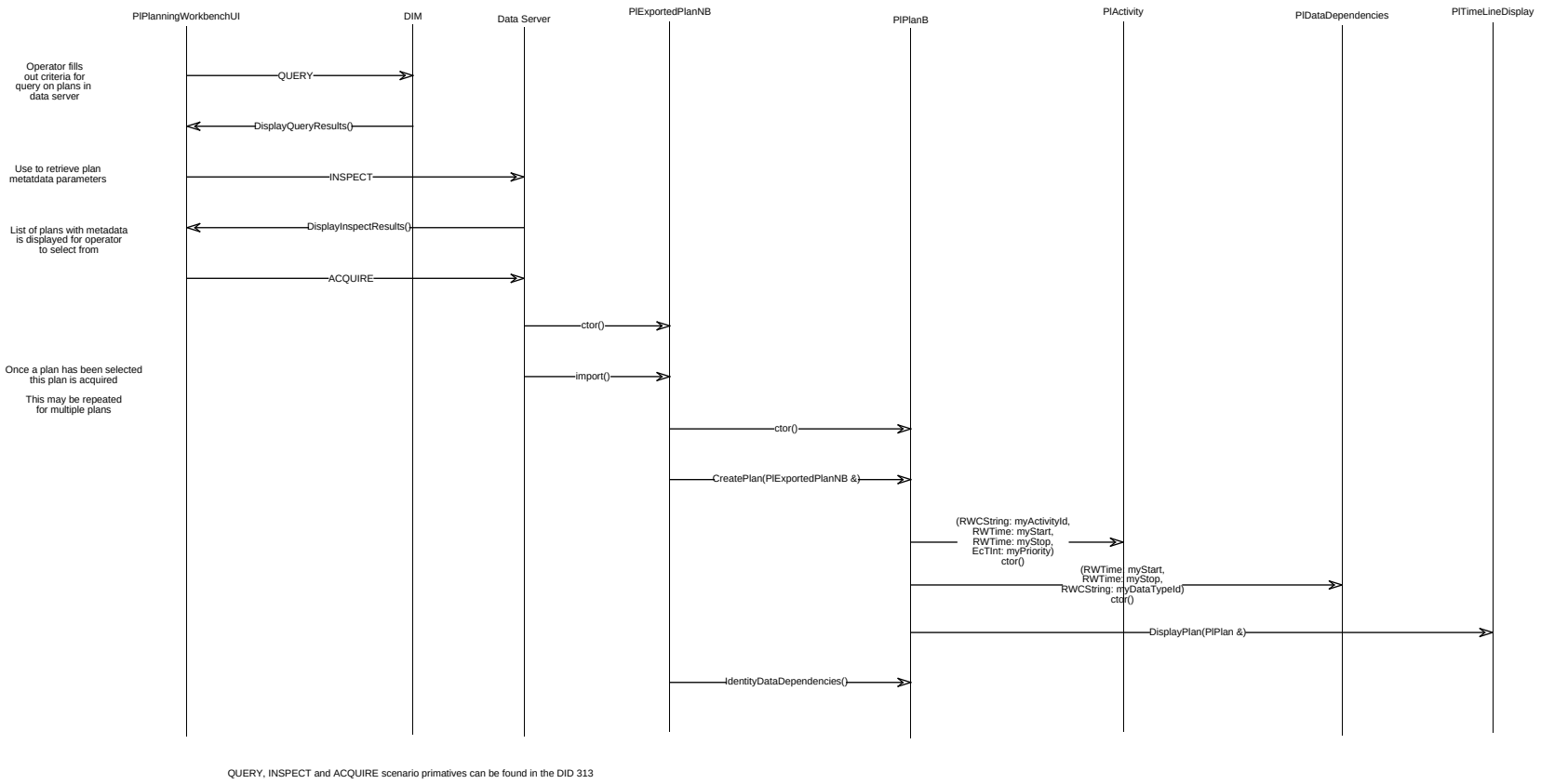
1. The user starts the planning workbench utility, the standard user authentication process applies (see scenario 4.4.18). The user specifies the time period for which to generate a plan within the GUI and initiates the creation of a plan as described in scenario 4.4.5 (Plan Creation).
2. Using the PlPlanGenerationUI, the user selects an option to add a plan from another DAAC to the display. The AddPlan() method of the PlPlanGenerationUI() requests query criteria from the user that will be used to find the plan(s) the user wishes to import.
3. The PlPlanGenerationUI creates a query for the DIM based on criteria entered by the user and then gets the metadata for any plan that meets the query criteria. This information is displayed for the user.
4. The user can then select which plans to import. For each plan, an ACQUIRE must be done to retrieve the appropriate exported plan. The user interface invokes the Import() method

of PlExportedPlanNB, specifying which DAAC's plan to use. As stated in scenario 4.4.7 (Publishing a plan), a plan can be published at any time.

5. The display of this second plan effectively looks as if it was another candidate plan from the same site, with different resources and jobs.
6. The IdentifyDataDependencies() method of PlPlan is then called to 'connect' the two plans by identifying any data dependencies on the display. Data dependency conflicts are identified.
7. Conflict resolutions are discussed in the 604 Operations Concept document Section 4.1.6.3.
8. Plans from other DAACs may be added as necessary.

#### 4.4.8.8 Event Trace

See Figure 4.4-8




QUERY, INSPECT and ACQUIRE scenario primatives can be found in the DID 313

***Figure 4.4-8. Importing a Plan from the Data Server***

### 4.4.9 Plan Activation Scenario

#### 4.4.9.1 Abstract

This scenario describes the activation of a plan from the Production Planning Workbench.

This scenario presents an abstract representation of the activities that occur within the Production Planning Object Library. The full detail of generating a plan is very complex, and intimately tied to the Production Planning Object Library. This scenario is presented to describe the process at a reasonable level of detail. For fuller description of the “planning/scheduling” aspects of this scenario please refer to the Production Planning Object Library CSC section (Section 4.5.5).

#### 4.4.9.2 Interfaces With Other Subsystems and Segments

Data Server for inserting Planning Data Availability Schedules. Data Processing for inserting DPRs from the current active plan into the job scheduler queue.

#### 4.4.9.3 Stimulus

The production planner initiates the Production Planning Workbench in order to activate a plan**.**

#### 4.4.9.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlGroundActivies
- PlGroundEvent
- PlGroundEvents
- PlPGEActivities
- PlDPRs
- PlDPRB
- PlActivities
- DPJobScheduler
- PlResourceManager

#### 4.4.9.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. There may be scheduled jobs in the data processing subsystem's job scheduler COTS.

#### 4.4.9.6 Ending State

No PDPS database server state change. A plan is activated (a portion of the active plan has been rolled into the data processing subsystem's job scheduler COTS).

#### 4.4.9.7 Scenario Description

1. The user starts the planning workbench utility, the standard user authentication process applies (see scenario 4.4.18). The user selects a plan which is to be used to drive the production schedule.
2. On recreation from the PDPS database the plan is automatically updated to reflect changes that have occurred since the plan was generated.
3. The plan object determines whether any new ground events have been defined, or old ground events have been deleted since the plan was generated or last updated. The Activities describing the ground events within a plan are updated accordingly and the plan is updated.
4. The plan object determines whether any of the data processing requests within the plan have been completed and updates the plan accordingly. Any data processing requests that have not been completed are consolidated into the new plan. The user may modify the plan at this point by redefining priorities in order to achieve a revised schedule.
5. The user initiates activation of the plan. The plan object will create an ordered list of the activities which are within the scheduling window. This window defines the portion of the plan that needs to be rolled into the COTS. By default this window is 24 hours from the current time, however it is modifiable within from the workbench utility.
6. The plan object will create a Planning Data Availability Schedule (PDAS) for distribution using the PlPDASFile object. These files are then INSERTED into the Data Server.
7. The plan object iterates through the activities defined in the order indicated by the plan, and calls the appropriate scheduling operation for that activity.
8. The Data Processing Request activities will initiate a call to create a PGE job within the data processing subsystem's job scheduling COTS.
9. The Ground Events will initiate a call to create a Ground Event Job within the data processing subsystem's job scheduling COTS.

Note: For details of how the jobs are created within the COTS and subsequently managed from the COTS please refer to scenarios within the Data Processing Design documentation.

#### 4.4.9.8 Event Trace

See Figure 4.4-9.

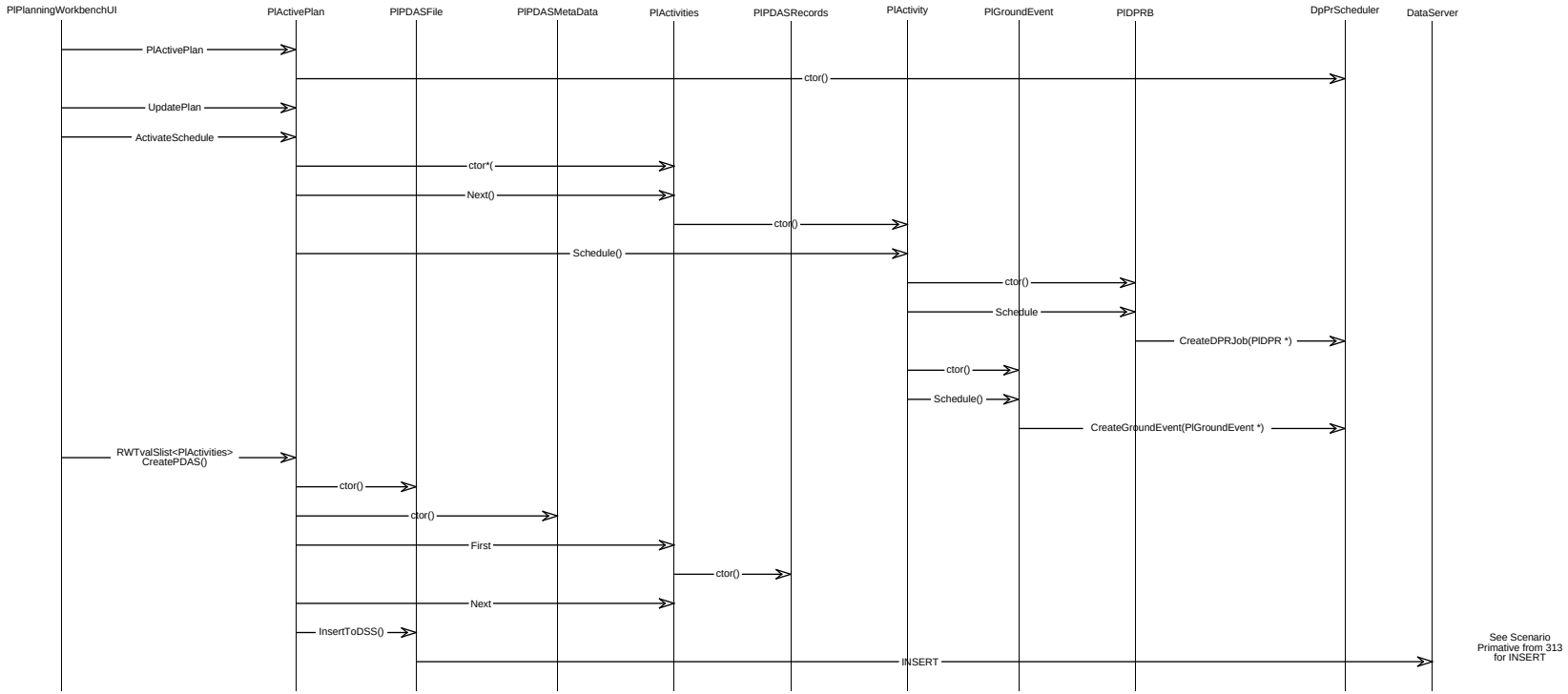



***Figure 4.4-9. Plan Activation Scenario Scenario***

### 4.4.10 Statusing a Plan Scenario

#### 4.4.10.1 Abstract

This scenario describes the statusing of a plan from the Production Planning Workbench. A component of the Job Scheduling COTS, AutoXpert now provides the dynamic display of the production schedule, therefore there are no requirements for dynamic updating to the plans. However there are occasions when the Plan from which the schedule originated would require updating, or statusing. Primarily the need for updating this will be before “downloading” a day's schedule into the COTS, taking into account the previous days production.

This scenario presents an abstract representation of the activities that occur within the Production Planning Object Library. The full detail of generating a plan is very complex, and intimately tied to the Production Planning Object Library. This scenario is presented to describe the process at a reasonable level of detail. For fuller description of the “planning/scheduling” aspects of this scenario please refer to the Production Planning Object Library CSC section (Section 4.5.5).

#### 4.4.10.2 Interfaces With Other Subsystems and Segments

None.

#### 4.4.10.3 Stimulus

The production planner initiates the Production Planning Workbench in order to activate a plan.

#### 4.4.10.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlGroundActivies
- PlGroundEvent
- PlPGEActivities
- PlDPRB
- PlActivities
- DPJobScheduler

#### 4.4.10.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. A portion of a plan has been activated (there are scheduled jobs in the data processing subsystem's job scheduler COTS).

#### 4.4.10.6 Ending State

No PDPS database server state. The plan is updated.

#### 4.4.10.7 Scenario Description

1. The user starts the planning workbench utility, the standard user authentication process applies (see scenario 4.4.18). The user selects the active plan which is to be updated.
2. On recreation from the PDPS database the plan is automatically updated to reflect changes that have occurred since the plan was generated.
3. The plan object determines whether any new ground events have been defined, or old ground events have been deleted since the plan was generated or last updated. The Activities describing the ground events within a plan are updated accordingly and the plan is updated.
4. The plan object determines whether any of the data processing requests within the plan have been completed and updates the plan accordingly.
5. The plan object creates an ordered list of the activities which are part of the active schedule (determined from the status attribute). For those activities which have not completed the plan object invokes the status method.
6. The PlDPRB and object interfaces with the DPJobScheduler class to return the status of the activity.
7. The plan is updated.

#### 4.4.10.8 Event Trace

See Figure 4.4-10.

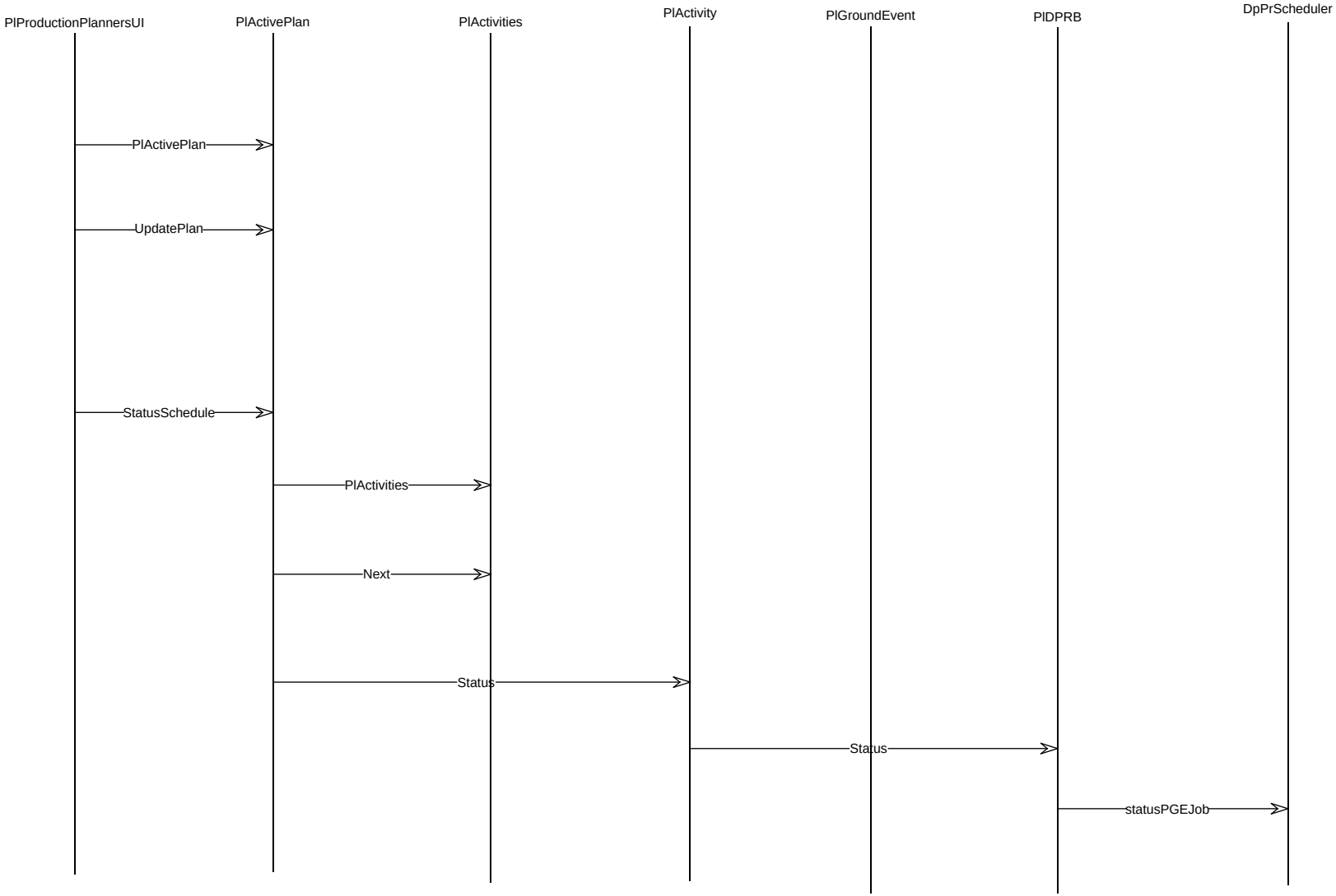



***Figure 4.4-10. Statusing a Plan Scenario***

### 4.4.11 Canceling a Plan Scenario

#### 4.4.11.1 Abstract

This scenario describes the canceling of an active plan from the Production Planning Workbench. The scenario illustrates the interface to the Data Processing subsystem through the DPJobScheduler interface class.

#### 4.4.11.2 Interfaces With Other Subsystems and Segments

The Planning subsystem interfaces with the Data Processing subsystem in order to cancel the scheduled Data Processing Requests.

#### 4.4.11.3 Stimulus

It's assumed that the Production Planning Workbench application is running and displaying the active plan. The user decides to cancel the production schedule from the workbench.

#### 4.4.11.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlActivePlan
- PlPGEActivities
- PlPGEActivity
- PlGroundEvent
- PlDPRB
- DPJobScheduler

#### 4.4.11.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. A portion of a plan has been activated (there are scheduled jobs in the data processing subsystem's job scheduler COTS).

#### 4.4.11.6 Ending State

No change in PDPS database server state. The scheduled activities are canceled.

#### 4.4.11.7 Scenario Description

1. The operator initiates the command to cancel the production schedule.
2. The plan object creates an ordered list of the activities which are part of the active schedule (determined from the status attribute).
3. The plan object iterates through the activities and invokes the cancel method from the activity.
4. The PlDPRB and PlGroundEvent classes interface with the DPJobScheduler class to cancel the activity.
5. The plan is updated.

#### 4.4.11.8 Event Trace

See Figure 4.4-11.

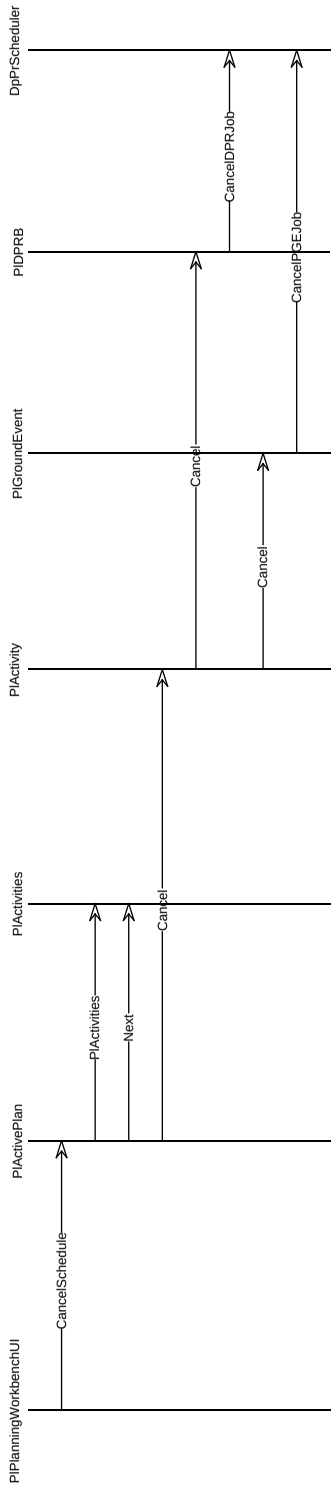



***Figure 4.4-11. Cancelling a Plan Scenario***

### 4.4.12 Deleting a Plan Scenario

#### 4.4.12.1 Abstract

This scenario describes the deletion plan from the Production Planning Workbench. This function does not delete the associated DPRs or Ground Events from the PDPS database, since they may be associated to other plans, and are also maintained for a set period such that reports may be generated against their completion status.

#### 4.4.12.2 Interfaces With Other Subsystems and Segments

None.

#### 4.4.12.3 Stimulus

It's assumed that the Production Planning Workbench application is running. The user decides to delete the production schedule from the workbench.

#### 4.4.12.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlPlanningWorkbenchUI
- PlPlan
- PlPGEActivities
- PlPGEActivity

#### 4.4.12.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. The Production Workbench application is running and that a plan has been selected.

#### 4.4.12.6 Ending State

No change in PDPS database server state. The plan is deleted.

#### 4.4.12.7 Scenario Description

1. The operator initiates the command to delete the displayed plan.
2. The plan object creates an ordered list of the activities within the plan.
3. The plan object iterates through the activities and invokes the delete method from the activity [note that the delete method means that the activity is removed from the PDPS database, as opposed to the destructor operation for the class, which deallocates the memory associated to the object within the application, the destructor is called after the delete].
4. The entry describing the plan itself in the database is deleted.

#### 4.4.12.8 Event Trace

None.

### 4.4.13 Subscription Notification Scenario

#### 4.4.13.1 Abstract

This scenario describes the procedure by which the subscription manager is informed of data arrival and performs the appropriate actions based on the methods of the data type subscribed to. PGE jobs defined within the Data Processing subsystem are released when all the necessary data become available. FOS Detailed Activity Schedules and Planning Data Availability Schedules create Data Availability Times persistent objects in the PDPS database.

Note: Upon notification of the receipt of an FOS DAS or a PDAS, please refer to the Data Availability Times scenario

#### 4.4.13.2 Interfaces With Other Subsystems and Segments

The Data Server subsystem initiates the subscription notification. The Planning subsystem releases PGE jobs within the Data Processing Subsystem when all requirements are satisfied.

#### 4.4.13.3 Stimulus

A notification of data arrival is sent from the Data Server subsystem.

#### 4.4.13.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- EcMpMsgCb
- PlSubMsgCb
- PlDataGranules
- PlDataGranule
- EcUrUR
- DsClESDTReference
- PlDataAvailabilityTimes

#### 4.4.13.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. The subscription manager application is running.

#### 4.4.13.6 Ending State

No change in PDPS database server, subscription manager state.

#### 4.4.13.7 Scenario Description

1. The HandleCbMsg callback is invoked upon notification of data arrival, and the UR is created from the notification message.
2. The InspectDataArrival method of the Data Server is invoked to extract the metadata associated with the UR presented in the data arrival notification.
3. Based on the information extracted from the metadata, RegisterDataArrival updates the availability information pertaining to the Data Granule.

4. The subscription manager determines the DPRs for which this data was required by selecting those predicted data instances which match the arrived data metadata.
5. The subscription manager searches the PGE profile to determine the type of PGE to be executed.

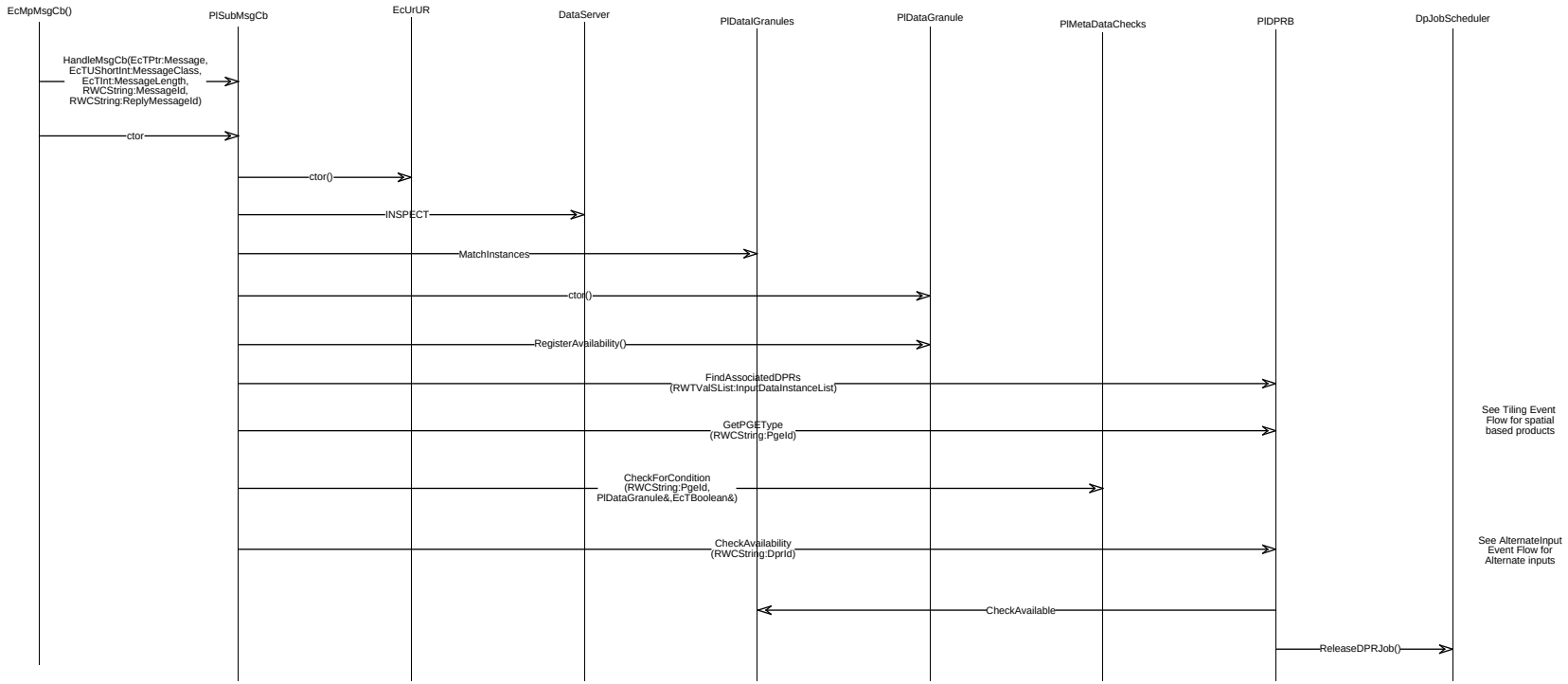
For Spatial based input. Please refer to the scenario entitled Subscription Notification with Spatial Based Input Scenario

6. The subscription manager checks the metadata to determine whether the data satisfies the QA requirements.
7. If the data dependencies have been fulfilled then the job associated with the DPR is released within the Data Processing subsystem

For Alternate inputs. Please refer to the scenario entitled Subscription Notification with Alternate Inputs Scenario

#### 4.4.13.8 Event Trace

See Figure 4.4-12.




INSPECT scenario primative can be found in the DID 313

***Figure 4.4-12. Subscription Notifications Scenario***

### 4.4.14 Subscription Notification with Spatial Based Input Scenario

#### 4.4.14.1 Abstract

This scenario describes the method by which the subscription manager processes notification of data granules that are geographical tiles instead of time continuous data. A description of the methods invoked once the GetPGEType operation determine that the type is spatial based . Please refer to the Subscription Notification Scenario for the Data notification procedures which precede this step.

#### 4.4.14.2 Interfaces With Other Subsystems and Segments

The Data Server subsystem initiates the subscription notification. The Planning subsystem queries the Data Server for additional input, and releases PGE jobs within the Data Processing Subsystem when all requirements are satisfied.

#### 4.4.14.3 Stimulus

A tile-based PGE receives notification of the arrival of input data.

#### 4.4.14.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlSubMsgCb
- PlDataGranule
- EcUrUR
- DsClESDTReference

#### 4.4.14.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. The subscription manager application is running.

#### 4.4.14.6 Ending State

No change in PDPS database server, subscription manager state.

#### 4.4.14.7 Scenario Description

1. The associated tile coordinates are obtained from the PGE Profile based on the PgeId and the tiling information pertaining to the PGE.
2. The query services of the Data Server are used to get the ESDT Reference Collector.
3. The InspectDataArrival method of the Data Server is invoked to extract the UR and accompanying metadata of all granules within the tiling scheme.
4. The subscription manager iterates through the URs and constructs a PlDataGranule for each granule within the tiling scheme creating an entry of this instance in the database with the corresponding metadata information.
5. For each granule, the meta data is checked to determine if the data has passed specified QA requirements.

6. The initial granule that was created by the production request editor to represent the DPR is deleted and jobs associated with the DPR are released to the Data Processing Subsystem.

#### 4.4.14.8 Event Trace

See Figure 4.4-13.

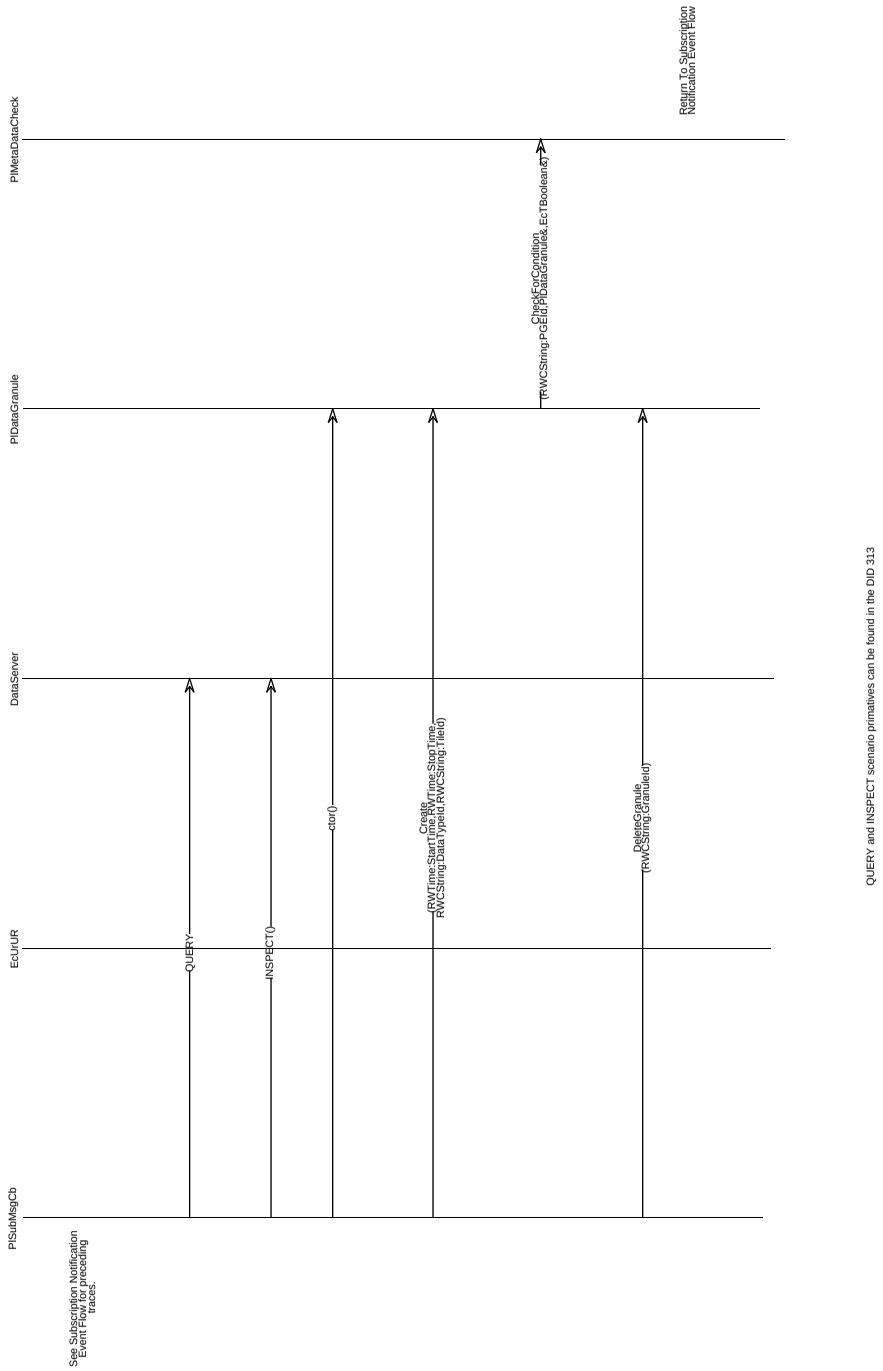



*Figure 4.4-13. Subscription Notification with Spatial Based Input Scenario*

### 4.4.15 Subscription Notification with Alternate Inputs Scenario

#### 4.4.15.1 Abstract

This scenario describes the procedure which occurs during the CheckAvailability process of a subscription notification, in the event that not all data dependencies have been satisfied and primary and backup data types are specified in the DPR. The methods below are invoked once the Check availability function determines that the required input associated with the DPR is available and the primary input specified for this DPR has not arrived. A description of the notification procedure prior to the check availability step may be found in the Subscription Notification Scenario.

#### 4.4.15.2 Interfaces With Other Subsystems and Segments

None.

#### 4.4.15.3 Stimulus

Upon notification that data has arrived, it is determined that all the required data for a DPR is available, but the primary input is still pending.

#### 4.4.15.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlAlternateDataGranuleNB
- PlDataGranule
- PlDPRB
- PlTimer

#### 4.4.15.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. The subscription manager application is running.

#### 4.4.15.6 Ending State

No change in PDPS database server, subscription manager state.

#### 4.4.15.7 Scenario Description

1. The Check Availability operations determine that not all data dependencies have been fulfilled for a particular DPR, and the required input is available.
2. The wait time associated with the primary input is obtained, and an instance of the timer is constructed to start the timer. The DPR continues to wait for notification of the arrival of additional data.
3. The callback functions of the timer object are notified once the time has expired.
4. The database is checked once again to determine if all the dependencies have been fulfilled. The primary input is still not available, and the alternate input specified is used thus changing the priority ordering of data.

5. Modify is invoked to modify the input data instance list for processing and the timer associated with the primary input is canceled. The job associated with the DPR is then released within the Data Processing subsystem

### 4.4.15.8 Event Trace

See Figure 4.4-14.

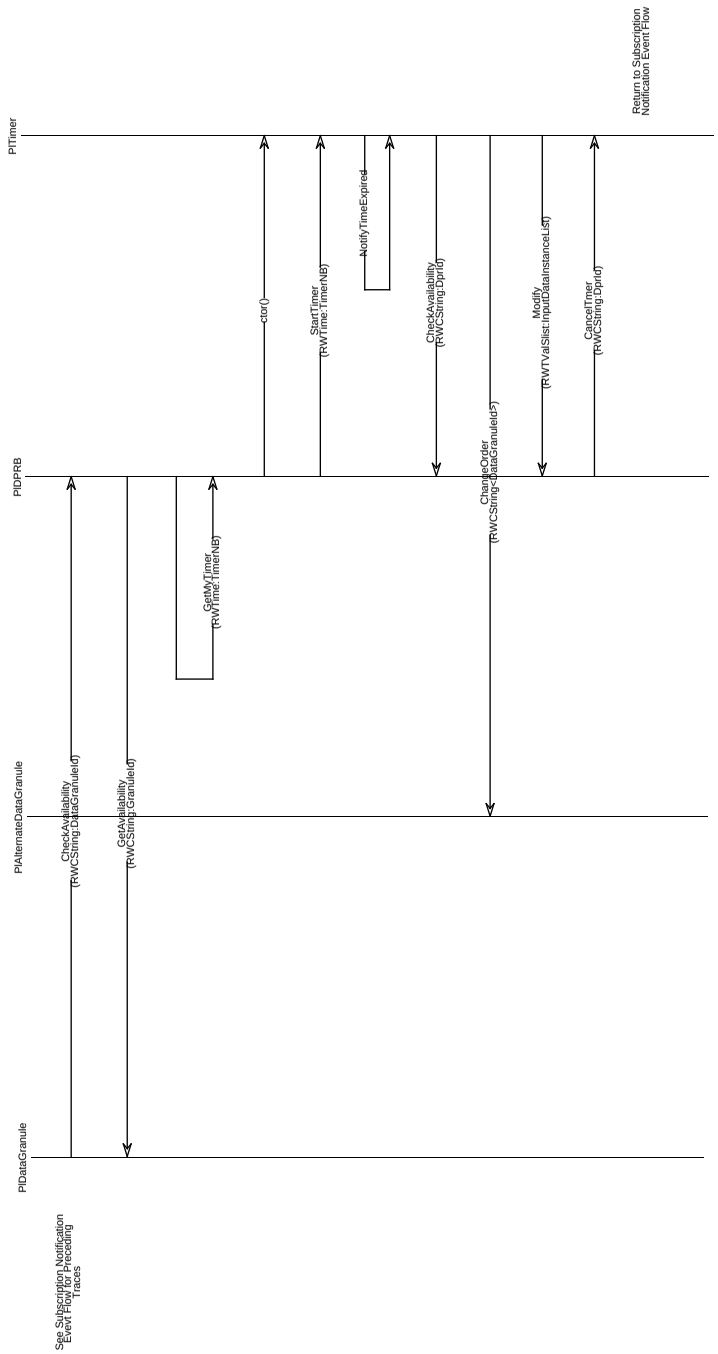



*Figure 4.4-14. Subscription Notification with Alternate Inputs Scenario*

### 4.4.16 Data Availability Times Scenario

#### 4.4.16.1 Abstract

This scenario describes the construction and disposition of Data Availability times schedules by the Planning Subsystem from within the Subscription manager.

#### 4.4.16.2 Interfaces With Other Subsystems and Segments

The Data Server subsystem provides and receives Planning Data Availability Schedules and FOS Detailed Activity Schedules.

#### 4.4.16.3 Stimulus

A notification of availability of a FOS Detailed Activity Schedule or a Planning Data Availability Schedule is sent by the Data Server subsystem.

#### 4.4.16.4 Participating Classes From the Object Model:

The following are participating classes from the Object Model:

- PlDataAvailabilityTimes
- PlDataSchedules
- PlFOSDASFile
- PlPDASFile
- PlDATRecord
- PlDataTransferHistory
- PlSourceToDsHistory
- PlInstrumentModes
- PlInstModeRecords

#### 4.4.16.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. The subscription manager application is running.

#### 4.4.16.6 Ending State

A persistent DataAvailabilityTimes object is stored in the PDPS database. PlSourceToDsHistoryNB is updated with predicted source (EDOS or another DAAC) to data server transfer times for all data types in the Schedule received.

#### 4.4.16.7 Scenario Description

1. The PlDataAvailabilityTimes( ) constructor is called by handleMsgCb when either a PDAS or an FOS Detailed Activity schedule arrival notification is received from the Data Server.
2. The PlDataSchedules Object invokes the data server services to acquires the Schedule.
3. The PlDATRecord( ) constructor is called to construct records based on the scheduling information received. The data is checked to see if this DAAC uses data referenced in this record, and records pertaining to data not subscribed to by the DAAC are ignored.

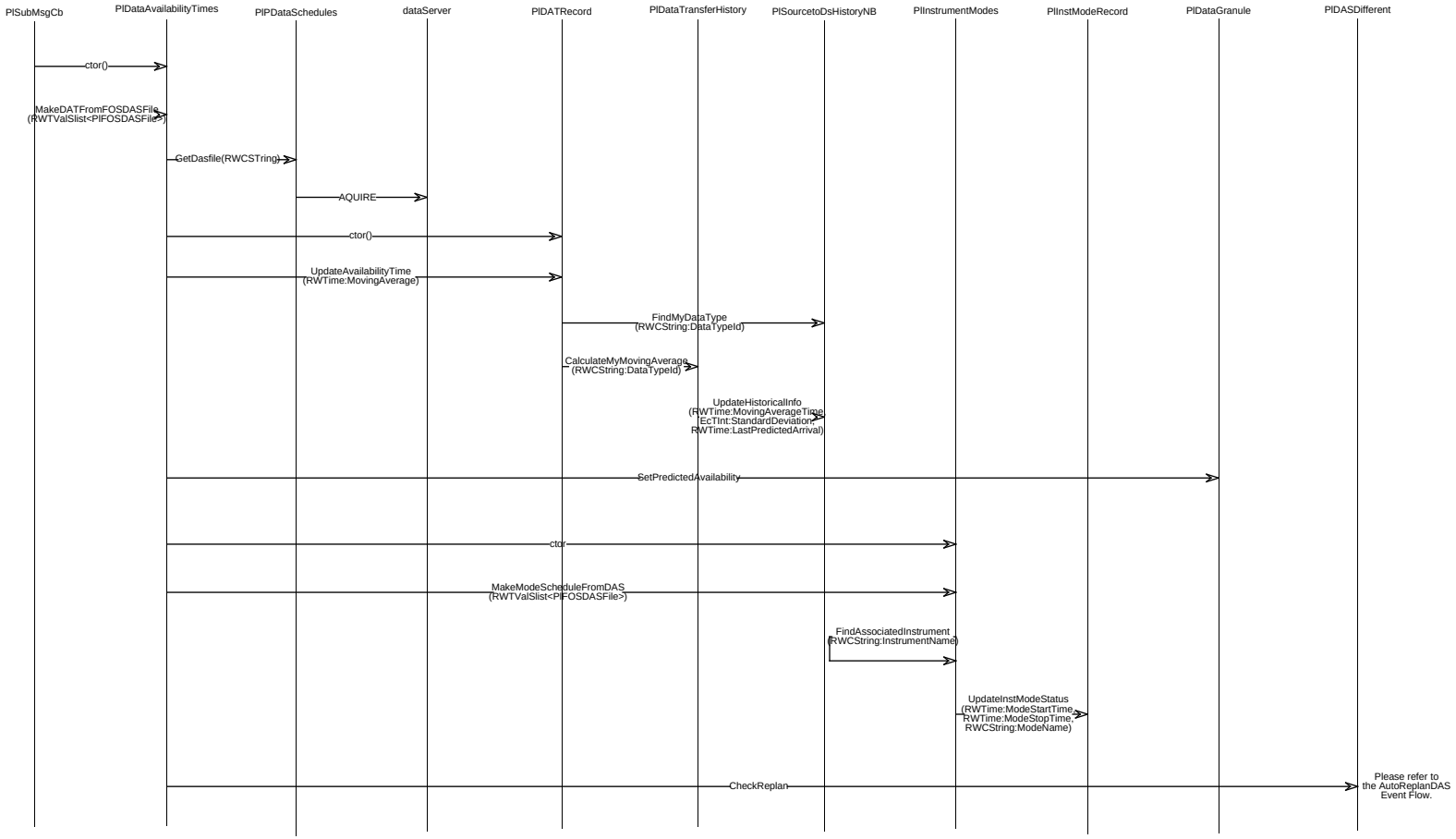
4. Based on the estimated time that the data is scheduled to be complete, the UpdateAvailability operation of the PlDATRecord is called to update the predicted arrival time to reflect the arrival time of the granule on the data Server.
5. The PlDataTransferHistory persistent object is referenced to calculate a new moving average of the transfer time of the granule to the data server. The new empirical estimates of the transfer time is reflected in the persistent object for that data type.
6. The corresponding PlDataGranule objects updated to reflect the new predicted arrival time, based on the predicted completion time of the granule and the historical transfer rate.

***Thread 1: constructing FOS instrument schedule.***

1. When a FOS DAS is received, the PlInstrumentModes constructor is called to construct an instrument mode schedule.
2. Mode records are constructed for all instruments relative to the DAAC. The FindAssociatedInstruments operation is called to parse PlInstModeRecords for related records.
3. The associated instrument mode records in PlInstModeRecords objects are retrieved and the mode set accordingly.
4. The check replan function is then invoked to determine if the new predictions differ significantly from the previous ones which may require a replan. See the scenario 4.4.17, Limited Automatic Replan based on a New DAS, for a complete description.

#### 4.4.16.8 Event Trace

See Figure 4.4-15.




AQUIRE scenario primitive can be found in the DID 313

***Figure 4.4-15. Data Availability Times Schedule***

### 4.4.17 Limited Automatic Replan based on a New DAS

#### 4.4.17.1 Abstract

This scenario describes how a limited automatic replan is initiated by the receipt of a new FOS Detailed Activity Schedule or Planning Data Availability Schedule. In the Planning Workbench, the operator can define a delta time for each data type that is an acceptable delay before a replan notification should be initiated.

Notification that a new DAS is available is sent to the Subscription Manager. The DAS is retrieved and information concerning the times that external data will be available to the local DAAC is extracted and stored in tables in the PDPS database. These times are compared with the times from the last DAS received; in the event that a product is going to be delayed greater than the delta specified by the operator, the operator is notified that a replan should be considered.

#### 4.4.17.2 Interfaces With Other Subsystems and Segments

None.

#### 4.4.17.3 Stimulus

After the data availability times have been extracted from an FOS DAS or a PDAS (see 4.4.16 Data Availability Times Scenario for more details), these times are compared previous predictions to determine if a replan is necessary.

#### 4.4.17.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlSubMgrCB
- PlDASDifferent
- PlDataAvailabilityTimes
- PlDATRecord
- PlDASDelta
- PlDataGranule
- PlDPRB
- PlPopupMessage
- PlPlanGenerationUI
- PlPlan

#### 4.4.17.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. This scenario is the details of step 7 of scenario 4.4.16 - the Data Availability Times Scenario.

#### 4.4.17.6 Ending State

No PDPS database server state. The new plan is activated (See scenario 4.4.9 Plan Activation Scenario for the details of plan activation).

#### 4.4.17.7 Scenario Description

1. After the data availability times are extracted from a DAS, these times are compared with the previous predictions of when this data would be available.
2. If a product is going to be delayed longer than an operator configurable amount for that data type, the Data Processing Requests waiting for that product are checked. If any Data Processing Request is predicted in the current active plan to start before this product will be available, a description of the product, the new availability time, and information on the DPR will be added to a report.
3. A pop-up display notifies the operator that a replan should be considered and displays the report with information on the delayed product and it's associated DPRs.
4. The operator decides based on this information to do a replan and presses the "replan" button on the pop-up display.
5 The Planning Workbench application is started if it is not already executing. The plan is rescheduled, taking into account the new data availability times and the current state of the Processing Queue. See Scenario 4.4.9 Plan Creation Scenario for more details.
6. The new plan is displayed on the timeline and is reviewed by the operator.
7. The operator determines that this plan is acceptable and activates this plan.

#### 4.4.17.8 Event Trace

See Figure 4.4-16.

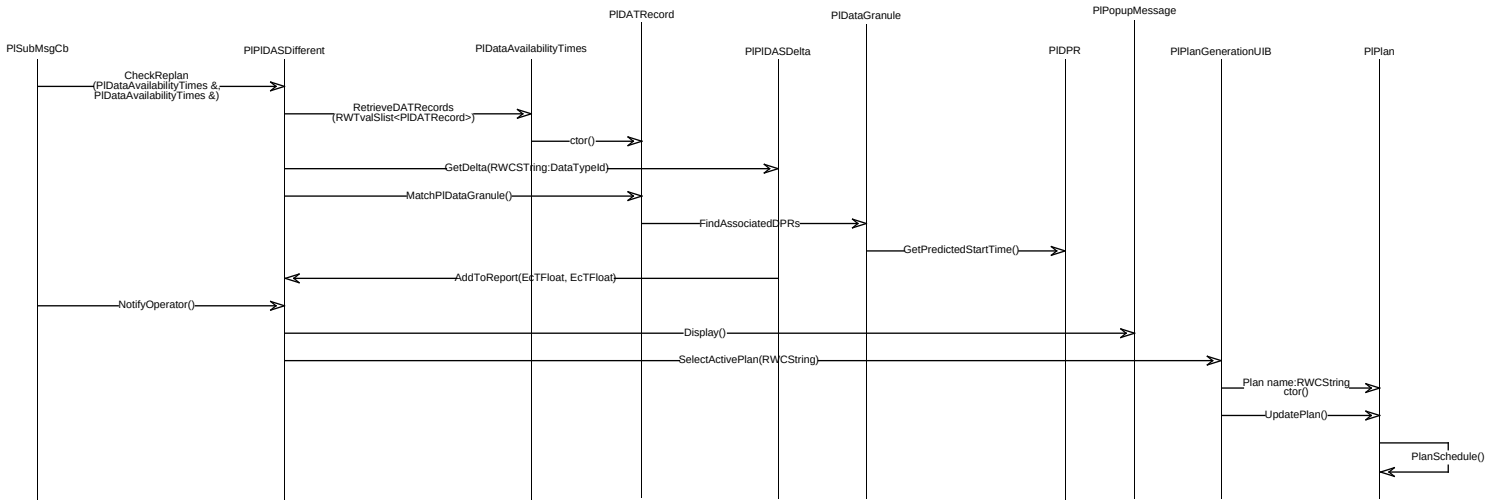



***Figure 4.4-16. Limited Automatic Replan Based on a New DAS***

### 4.4.18 User Logon and Authentication Scenario

#### 4.4.18.1 Abstract

This scenario describes user logon and authentication to access the Planning operational software. There are three classes of users, the Production Planner, Resource Manager, and the ECS user. The Production Planner and Resource Manager are operational staff primarily responsible for scheduling planning activities and managing resources respectively. The ECS user is primarily responsible for entering production requests.

The Planning software only permits users to access certain functional areas within the Planning software system. To this end, certain restrictions are place on different types of users. For instance, an ECS user cannot create a candidate plan whereas a Production Planner is allowed to create a candidate plan. These access rights are enforced using CSS User Authentication services. User accounts and associated privileges will be registered with CSS and maintained by local database administrative personnel.

#### 4.4.18.2 Interfaces with other Subsystems and Segments

CSS User Authentication services

#### 4.4.18.3 Stimulus

The operator logs into the Planning software and initiates the Planning software.

#### 4.4.18.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlApplication,
- PlPlanningWorkbenchUI,
- PlProductionRequestUI, and
- PlUser.

#### 4.4.18.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running.

#### 4.4.18.6 Ending State

No change to the database server state.

#### 4.4.18.7 Scenario Description

1. The user logs into UNIX supplying an account name and password.
2. The user invokes the Planning software to gain access to its operational environment.
3. Upon initiation of the Planning software, the user's account name is captured.
4. Once the user account name is captured, the Planning software makes a connection to the database server to verify the account name.

***Thread 1: Invalid User***

5. If there is no match in the database for the corresponding account name, access will not be permitted to the Planning operational environment.
6. The Planning software notifies the user that he is not authorized to use Planning and all Planning services are denied.

***Thread 2: Valid User***

7. If the user is a valid user, the Planning software queries the database to obtain the appropriate privileges associated with the user.
8. The software assesses the privileges retrieved from the database and restricts the user to the access only the services within the system that are applicable to the user's class type.
9. Upon successful authentication the user proceeds to use the Planning software based on the user privileges retrieved from the database.

#### 4.4.18.8 Event Trace

None.

### 4.4.19 On-Demand Production Request Scenario (Processed)

This scenario describes how an On-Demand Production Request (OPR) is handled by the Planning subsystem. In this scenario: the input data is available, the resource usage of the DPRs is below the thresholds set for individual OPRs and the combined threshold for all OPR jobs for the production period, and the job is submitted to processing for production, no replan is necessary.

#### 4.4.19.1 Beginning Assumptions

On-Demand Manager is up and running

#### 4.4.19.2 Interfaces with other Subsystems and Segments

On-Demand Production Request modifications and cancellations are received by the Planning subsystem from the Data Server. Planning subsystem sends disposition and status information to the Data Server.

#### 4.4.19.3 Stimulus

The On-Demand Production Request is received by the On-DemandPR class from the Data Server and the Data Server makes a call to the Validate operation of On-DemandPR class.

#### 4.4.19.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlOnDemandManagerNB
- PlOnDemandPRNB
- PlRescUseThreshNB
- PlDPRB
- PlOnDemandExceed

- DpPrScheduler

#### 4.4.19.5 Beginning System, Segment and Subsystem State(s)

PDPS Database and On-Demand Manager are in a steady state, up and running.

#### 4.4.19.6 Ending State

PDPS Database and On-Demand Manager are in a steady state, up and running.

#### 4.4.19.7 Scenario Description

1. The Data Server creates an instance of a PlOnDemandPRNB.
2. Data Server executes the ValidateOPR operation of PlOnDemandPRNB..
3. PlOnDemandPRNB executes PlDefinePgeRuns which it inherits from PlProductionRequestB to generate the DPRs and associated objects as in Production Request event trace.
4. PlOnDemandPRNB adds this production request to the list of production requests maintained by On-Demand Manager by calling AddOPR in PlOnDemandManagerNB.
5. PlOnDemandPRNB generates an instance of PlRescUseThreshNb.
6. Compare the predicted resource usage with PlRescUseThreshNB thresholds for the On-Demand Production Request and the cumulative thresholds for all On-Demand Production Requests by calling CheckThresholds.
7. Resource usage associated with On-Demand Production Request does not exceed thresholds for requests, add resource time to Cumulative Resource attribute in On Demand Manager via the UpdataRescUse operation.
8. Call the Schedule routine in PlDPRB to add the Dpr to the AutoSys production list.
9. The above operation results in CreateDprJob in DpPrScheduler being called which is what actually does the transfer of Dpr information to AutoSys.
10. Check for data availability at the Data Server via the CheckAvailability operation in PlDPRB. (Data Availability Scenario)
11. On-Demand Production Request is released for production
12. On-Demand Production Request is submitted to Processing for production.
13. Replan threshold are checked and no replan notification is necessary.
14. PlOnDemandPRNB notifies Data Server of processing submission (via return of ValidateOPR operation).
15. Data Server then calls the ReturnUR operation to initiate notification of the generation of the product and it’s UR.

16. PlOnDemandPRNB then calls the PlOnDemandManagerNB SubmitSubscription operation to create a subscription for notification of Data Server receiving a product of the data type requested by this production request.
17. PlOnDemandManagerNB receives notification of the insertion of a product of the type requested by the production request.
18. PlOnDemandManagerNB calls RequestCompleted with the UR of the product to notify OnDemandPRNB of the product completion.
19. PlOnDemandPRNB then calls RemoveOPR to remove the production request from the list of production requests maintained by PlOnDemandManagerNB.
20. ReturnUR then returns the UR of the product to Data Server and Data Server calls The PlOnDemandPRNB destructor to delete the instance of PlOnDemandPRNB.

#### 4.4.19.8 Event Trace

See Figure 4.4-17.

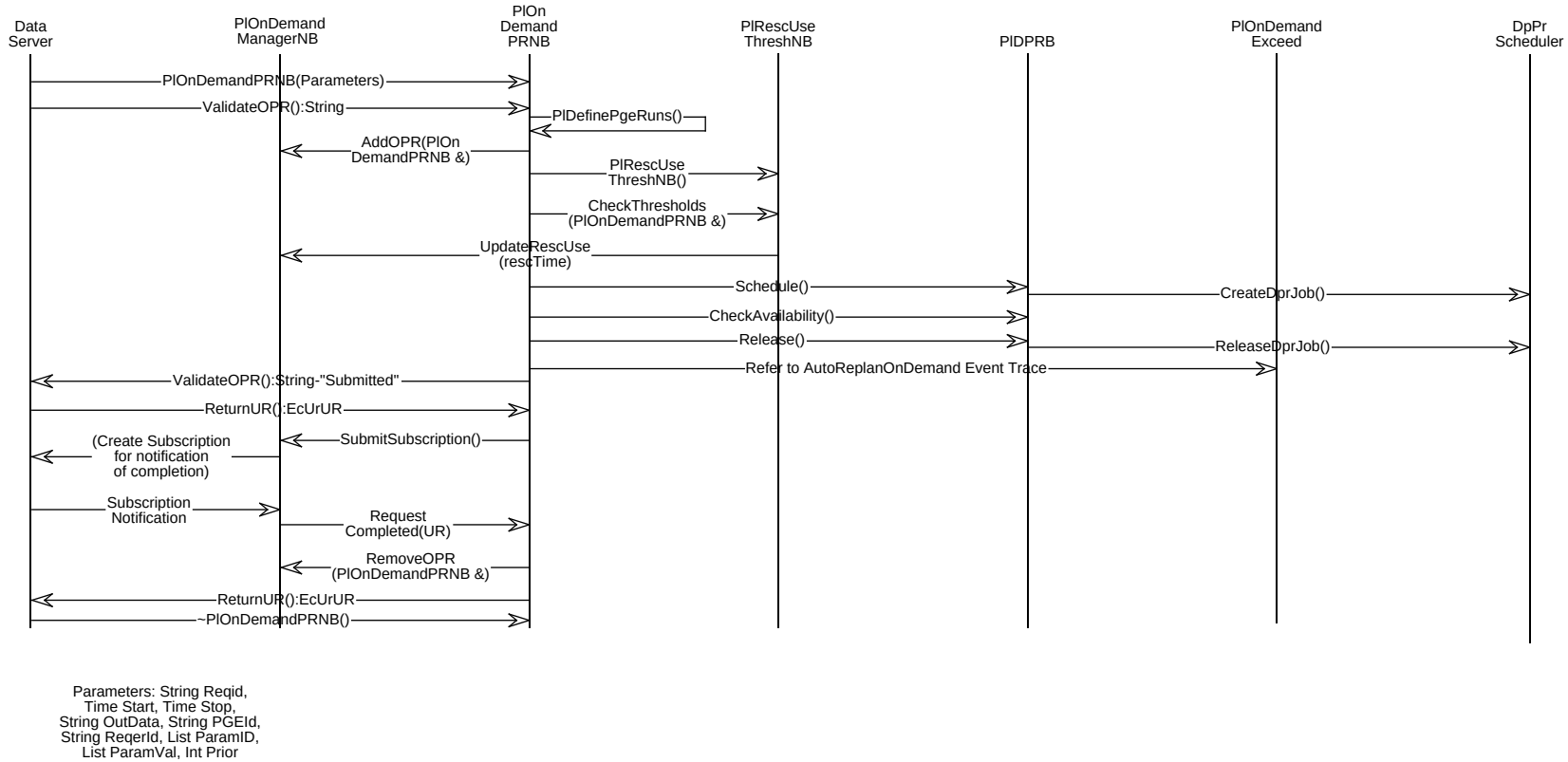



***Figure 4.4-17. On-Demand Production Request Scenario (Processed)***

### 4.4.20 Limited Automatic Replan based on an On-demand Production Request Scenario

#### 4.4.20.1 Abstract

This scenario describes how an automatic replan notification is initiated by an On-demand Production Request (OPR). There are two thresholds associated with an OPR - a basic resource usage threshold determines whether a OPR goes directly to processing or is deferred to be planned later by an operator. The second threshold determines whether a replan notification is sent to the operator based on this OPR.

While the default is that these thresholds are equal, there are good cases for these thresholds being different. In one situation, an operator may not want to know every time an OPR is deferred for later planning and instead only wants to know when a group of OPRs has been deferred - in this case, the limited automatic replan threshold may be much higher than the resource usage threshold. However, in another situation where the DAAC expects many OPRs, such as EDC with ASTER, the resource usage threshold may be set very high so OPRs are sent directly to processing, while the replan threshold is set lower, so the Production Plan can be periodically kept in sync with what is going on in Processing.

#### 4.4.20.2 Interfaces With Other Subsystems and Segments

None.

#### 4.4.20.3 Stimulus

After an On-demand Production Request is checked against the resource usage thresholds, it is compared to the limited automatic replan thresholds.

#### 4.4.20.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlOnDemandPR
- PlOnDemandExceed
- PlOnDemandReplanValues
- PlPopupMessage
- PlPlanGeneration
- PlPlan

#### 4.4.20.5 Beginning System, Segment and Subsystem State(s)

The PDPS database server is running. This scenario shows the details of step 13 of Section 4.4.19, the On-Demand Production Request Scenario (Processed).

#### 4.4.20.6 Ending State

No PDPS database server state. The new plan is activated (See scenario 4.4.9 Plan Activation Scenario for the details of plan activation).

#### 4.4.20.7 Scenario Description

1. After an On-demand Production Request is compared with the current resource usage thresholds, it is compared with each of the current replan thresholds.
2. Each time a replan threshold is exceeded, information about the exceeded resource is written to a report.
3. A pop-up display notifies the operator that a replan should be considered and displays the report with information on the On-demand Production Request and the exceeded thresholds.
4. The operator decides based on this information to do a replan and presses the "replan" button on the pop-up display.
5 The Planning Workbench application is started if it is not already executing. The on-demand production request is added to the current active plan and the plan is rescheduled, taking into account the current state of the Processing Queue. See Scenario 4.4.5 Plan Creation for more details.
6. The new plan is displayed on the timeline and is reviewed by the operator.
7. The operator determines that this plan is acceptable and activates this plan.

#### 4.4.20.8 Event Trace

See Figure 4.4-18.

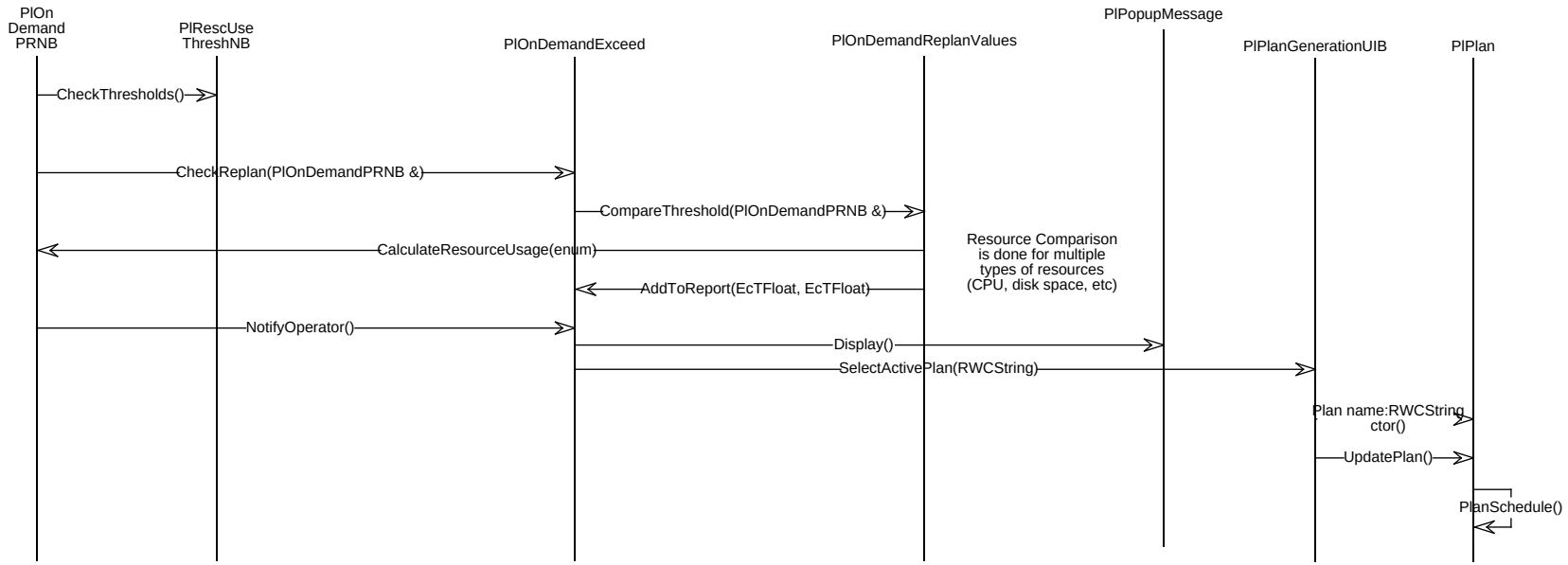



***Figure 4.4-18. Limited Automatic Replan Based on an On-Demand Production Request Scenario***

### 4.4.21 On-Demand Production Request Scenario (Delayed)

This scenario describes how an On-Demand Production Request (OPR) is handled by the Planning subsystem. In this scenario the request meets the acceptance criteria, however the input data is not available, so a subscription is created for notification of receipt of the data and the job is held in the PDPS database for inclusion in a subsequent plan for production.

#### 4.4.21.1 Beginning Assumptions

**On-Demand Manager is up and running**

#### 4.4.21.2 Interfaces with other Subsystems and Segments

On-Demand Production Request modifications and cancellations are received by Planning subsystem from the Data Server. Planning subsystem sends disposition and status information to the Data Server.

#### 4.4.21.3 Stimulus

The On-Demand Production Request is received by the On-DemandPR class from the Data Server and the Data Server makes a call to the Validate operation of On-DemandPr class.

#### 4.4.21.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlOnDemandManagerNB
- PlOnDemandPRNB
- PlRescUseThreshNB
- PlDPRB
- PlOnDemandExceed
- DpPrScheduler
- PlSubscriptionSubmitIF

#### 4.4.21.5 Beginning System, Segment and Subsystem State(s)

PDPS Database and On-Demand Manager are in a steady state, up and running.

#### 4.4.21.6 Ending State

PDPS Database and On-Demand Manager are in a steady state, up and running.

#### 4.4.21.7 Scenario Description

1. The Data Server creates an instance of a PlOnDemandPRNB.
2. Data Server executes the ValidateOPR operation of PlOnDemandPRNB..
3. PlOnDemandPRNB executes PlDefinePgeRuns which it inherits from PlProductionRequestB to generate the DPRs and associated objects as in Production Request event trace.

4. PlOnDemandPRNB adds this production request to the list of production requests maintained by On-Demand Manager by calling AddOPR in PlOnDemandManagerNB.
5. PlOnDemandPRNB generates an instance of PlRescUseThreshNb.
6. Compare predicted resource usage with PlRescUseThreshNB thresholds for the On-Demand Production Request and the cumulative thresholds for all On-Demand Production Requests by calling CheckThresholds.
7. Resource usage associated with On-Demand Production Request does not exceed thresholds for requests, add resource time to Cumulative Resource attribute in On Demand Manager via the UpdataRescUse operation.
8. Call the Schedule routine in PlDPRB to add the Dpr to the AutoSys production list.
9. The above operation results in CreateDprJob in DpPrScheduler being called which is what actually does the transfer of Dpr information to AutoSys.
10. Check for data availability at the Data Server via the CheckAvailability operation in PlDPRB. (Data Availability Scenario)
11. The data is not all available so a subscription is requested to notify planning when the data becomes available.
12. PlOnDemandPRNB notifies Data Server of processing delay (via return of ValidateOPR operation.
13. Data Server then calls the ReturnUR operation to initiate notification of the generation of the product and it's UR.
14. PlOnDemandPRNB then calls the PlOnDemandManagerNB SubmitSubscription operation to create a subscription for notification of Data Server receiving a product of the data type requested by this production request.
15. PlOnDemandManagerNB receives notification of the insertion of a product of the type requested by the production request.
16. PlOnDemandManagerNB calls RequestCompleted with the UR of the product to notify OnDemandPRNB of the product completion.
17. PlOnDemandPRNB then calls RemoveOPR to remove the production request from the list of production request maintained by PlOnDemandManagerNB.
18. ReturnUR then returns the UR of the product to Data Server and Data Server calls The PlOnDemandPRNB destructor to delete the instance of PlOnDemandPRNB.

#### 4.4.21.8 Event Trace

See Figure 4.4-19.

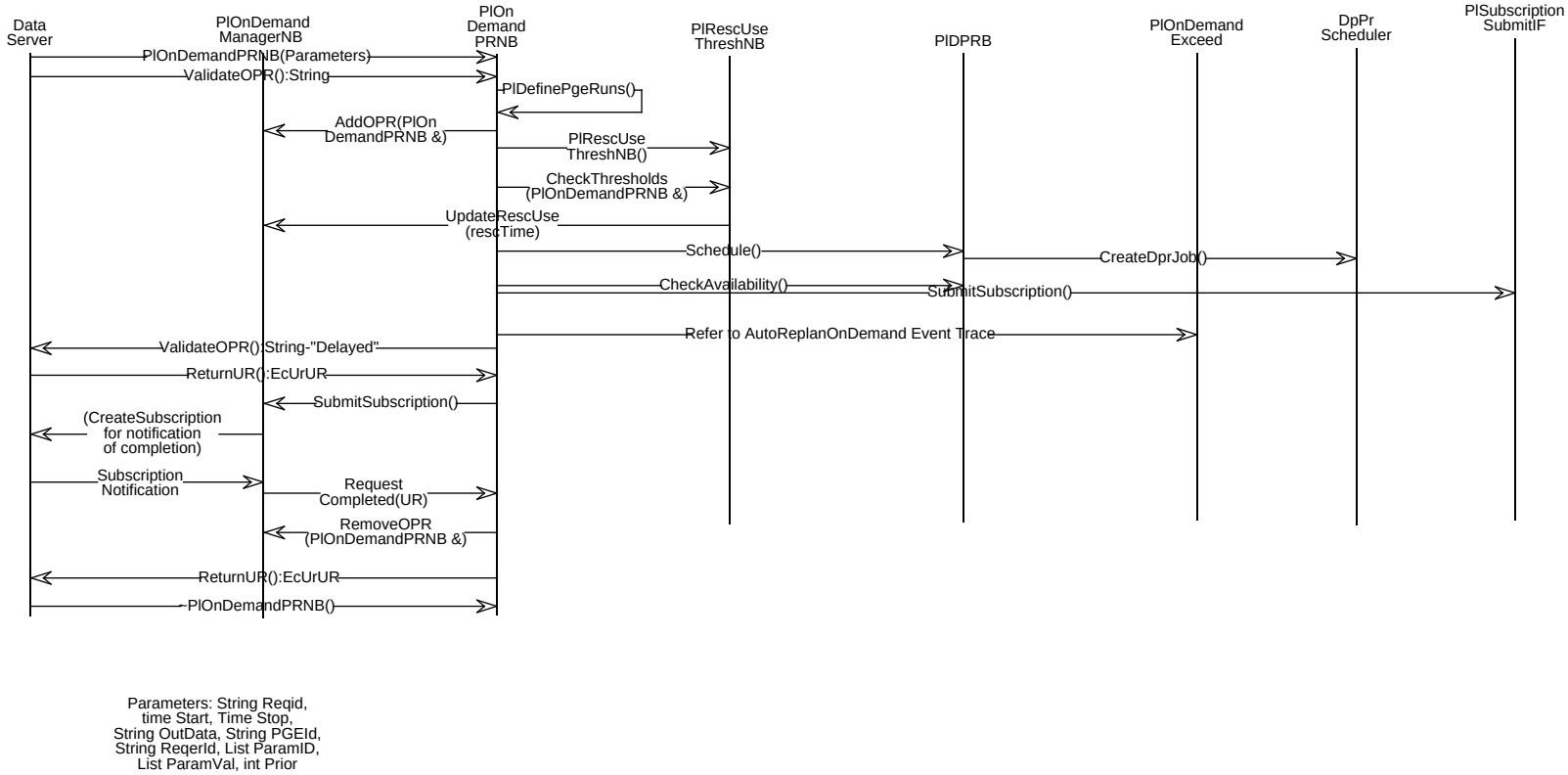



***Figure 4.4-19. On-Demand Production Request Scenario (Delayed) Request***

### 4.4.22 Entering and Approving a Resource Reservation Scenario

#### 4.4.22.1 Abstract

This scenario describes the methods by which a Production Planner enters and approves a hardware resource reservation . Resource reservations are entered, validated and approved. Validation is not software driven; rather it is based upon the Planner's evaluation of the reservation as complete and reasonable. Approval consists of determining whether any conflicts exist between reservations for resources. This scenario describes a sequence of events that occur in the entry of a resource reservation.

#### 4.4.22.2 Interfaces With Other Subsystems and Segments

The MSS DB is queried to extract actual resource usage data in order to generate planned versus actual usage reports.

#### 4.4.22.3 Stimulus

A resource reservation is entered or modified.

#### 4.4.22.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlRpResourcePlanningUI
- PlRpResourceReservationPool
- PlRpActivityType
- PlRpResourceReservation
- PlRpExecutable
- PlRpResourceState
- PlRpComplexResourceReservation
- MsMdBrowseLog

#### 4.4.22.5 Beginning System, Segment and Subsystem State(s)

PDPS and MSS Databases are in a steady state, up and running. The PDPS database has been loaded with the MSS resource configuration and initialized with default resource allocations and other ground event activities.

#### 4.4.22.6 Ending State

PDPS and MSS Databases are in a steady state, up and running.

#### 4.4.22.7 Scenario Description

1. The Production Planner starts the Resource Planning Workbench user interface.
2. The Planner enters a resource reservation with the EditReservation method in the PlRpResourcePlanningUI class. The Planner can either choose a reservation from a displayed list and modify it or enter a new one. Information entered includes the reservation start and stop times, the planned activity, whether the reservation shares a

resource with another reservation, desired resources, and descriptive information. If the reservation is for an activity that is performed at regular intervals, the Planner can also enter the frequency and the reservation is exploded into multiple reservations.

3. The Planner validates the reservation, using the PlRpResourceReservation class Validate method. If the Planner instead marks the reservation as not valid, the reservation can be modified with the EditReservation method. The PlRpResourceReservation TransitionTo method is used to change the reservation's validation state.
4. If validated, the reservation is allocated to a resource and added to the resource state list. The PlRpResource Allocate and PlRpResourceState constructor methods are employed. The PlRpResourceReservation Insert method is used to insert the reservation into the reservation pool.
5. The Planner initiates the approval process with the EditReservation method of the PlRpResourcePlanningUI class.
6. The PlRpResourceReservation Approve method is invoked. This in turn calls the DetectConflicts method. This method loops over all the resources allocated for the reservation to determine whether there are conflicts for a resource. If a reservation is flagged as sharing a resource, those conflicts are ignored.
7. Depending on whether or not there are conflicts, the Planner sets the reservation approval state with the TransitionTo method. If it's not approved, the Planner can modify the reservation again with EditReservation method.

#### 4.4.22.8 Event Trace

See Figure 4.4-20.

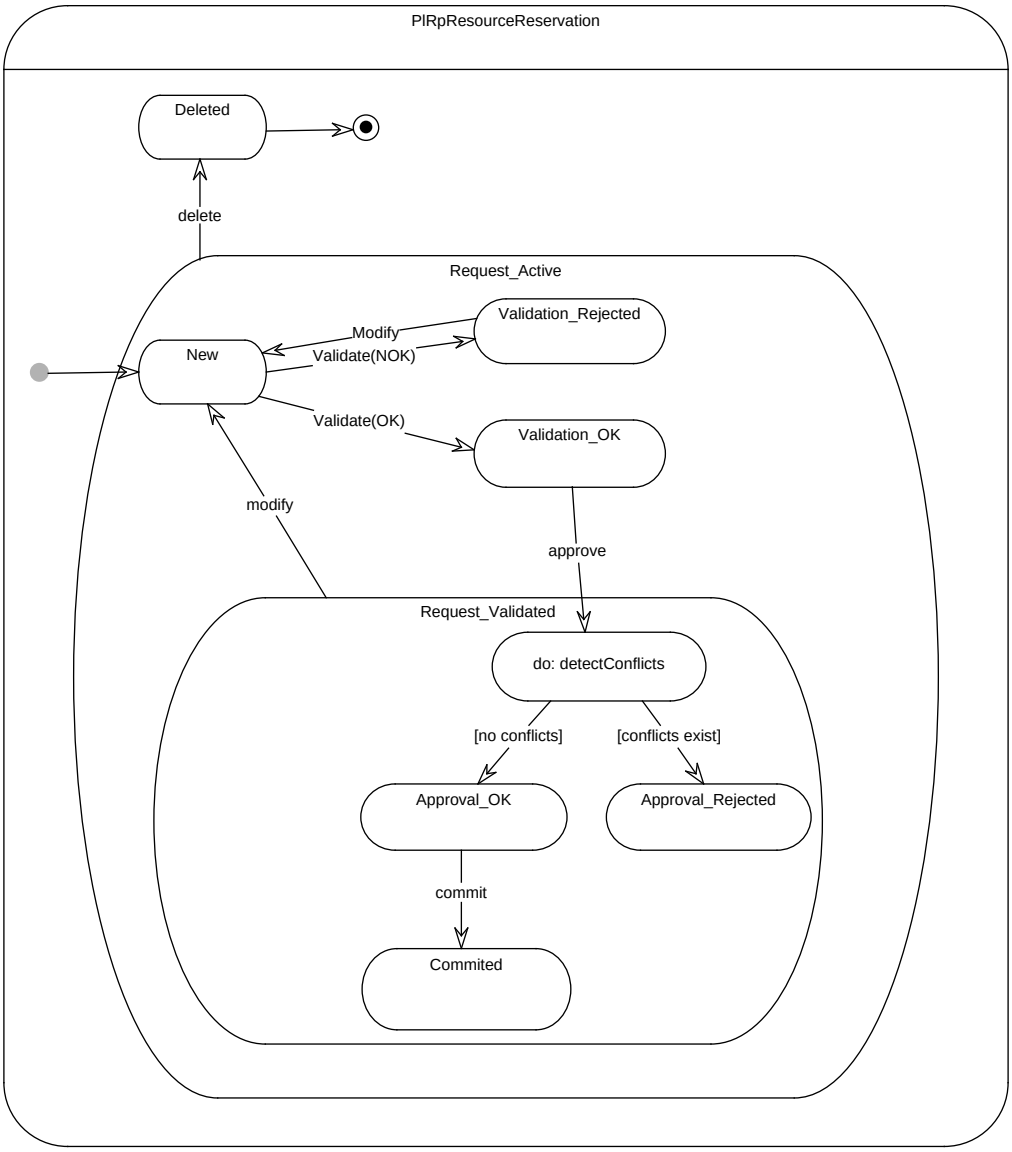



***Figure 4.4-20. Entering and Approving a Resource Reservation Scenario***

### 4.4.23 Creating a New Resource Plan

#### 4.4.23.1 Abstract

This scenario describes how the Resource Planning Workbench displays a new resource plan with all the currently approved resource reservations.

#### 4.4.23.2 Interfaces With Other Subsystems and Segments

None.

#### 4.4.23.3 Stimulus

The Resource Planner selects 'Display' in the Resource Planning Workbench.

#### 4.4.23.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlRpResourcePlanningUI
- PlRpResourceReservation
- PlRpResourceState
- PlRpResource
- PlRpResourceTimeline

#### 4.4.23.5 Beginning System, Segment and Subsystem State(s)

There exists multiple approved resource reservations in the database.

#### 4.4.23.6 Ending State

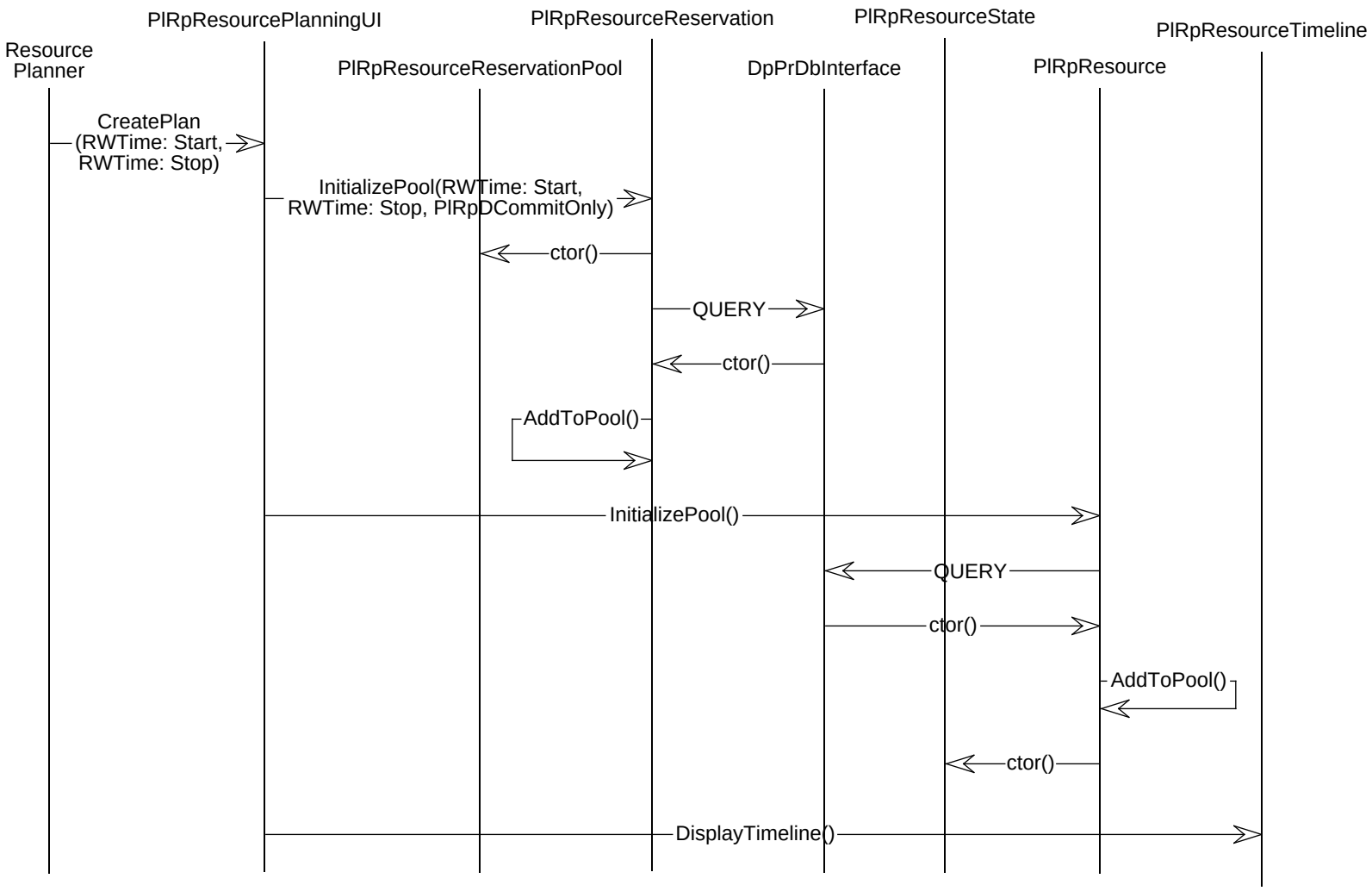
No database server state.

#### 4.4.23.7 Scenario Description

1. The Resource Planner decides that all the resource reservations have been reviewed and enters the command to generate a new Resource Plan.
2. A query on the database initializes the resource reservation pool with every request that has been approved or committed.
3. A second query on the database initializes the resource pool with the available resources.
4. This information is used by the Timeline to generate a graphical display of the plan which can be reviewed by the Resource Planner.

#### 4.4.23.8 Event Trace

See Figure 4.4-21.




For details on querying a database, see the Data Processing 305

***Figure 4.4-21. Create Resource Plan Scenario***

### 4.4.24 Committing a Resource Plan

#### 4.4.24.1 Abstract

This scenario describes what happens when the Resource Planner decides to commit the resources in a plan. This alters the resource baseline for production planning.

#### 4.4.24.2 Interfaces With Other Subsystems and Segments

None.

#### 4.4.24.3 Stimulus

After the creation of a resource plan, the Resource Planner selects 'Commit' in the Resource Planning Workbench.

#### 4.4.24.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlRpResourcePlanningUI
- PlRpResourceReservation
- PlRpResourceState
- PlRpResource
- PlResource
- PlGroundEvent
- PlGroundEventExecutable
- PlGroundEventAllocation
- PlResourceRequirement

#### 4.4.24.5 Beginning System, Segment and Subsystem State(s)

The database server is running.

#### 4.4.24.6 Ending State

No database server state.

#### 4.4.24.7 Scenario Description

1. The Resource Planner reviews the current Resource Plan and decides to commit it.
2. First, all resource reservations that have been approved have their state changed to "committed"
3. Next, the list of currently defined resources in Resource Planning are compared with those used for Production Planning currently in the PDPS database. Any new resources who have a default activity of "Production" are added to the PDPS database. Any modified or deleted resources are deleted. The new definition for resources that have been modified are added.

4. Finally, the list of currently defined resource reservations in Resource Planning are compared with the Ground Events defined in Production Planning. The Production Planning database is updated as with the resources in step 3. In addition, for each reservation added, a PlResourceRequirement and a PlGroundEventAllocation need to added as well. For reservations that are associated with executables (such as backup scripts), a PlGroundEventExecutable is created as well.

#### 4.4.24.8 Event Trace

See Figure 4.4-22.

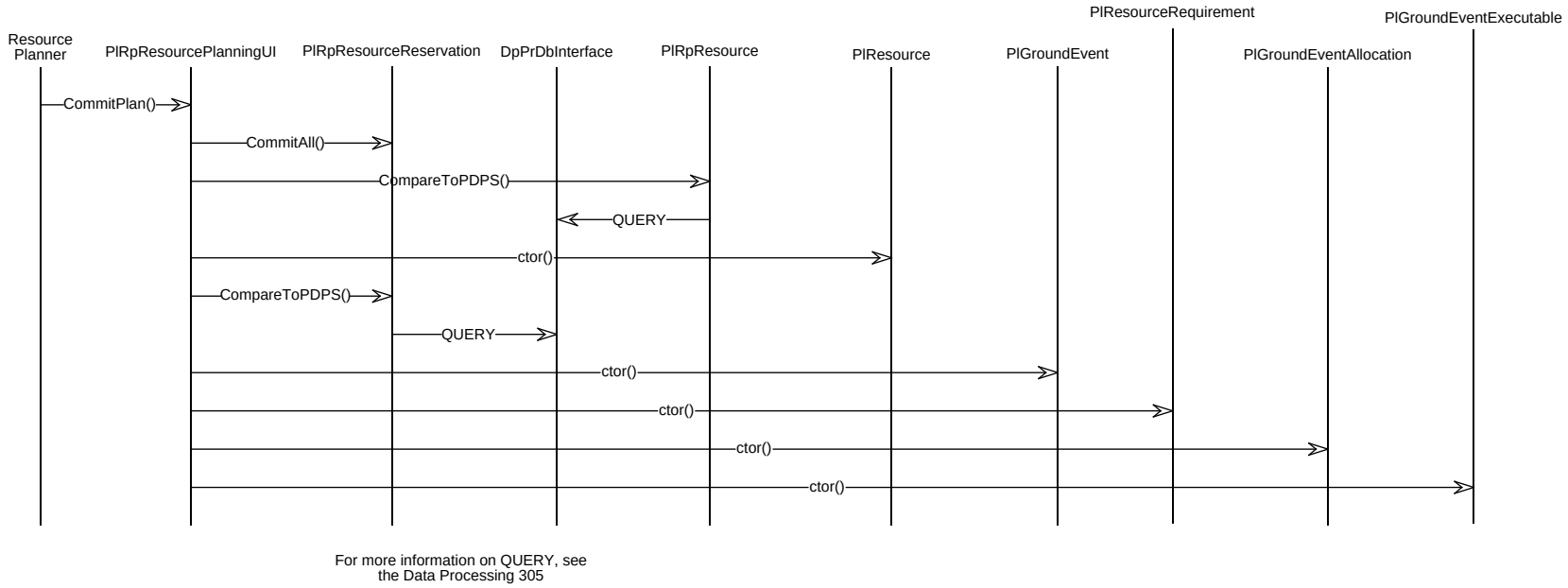



***Figure 4.4-22. Commit Resource Plan Scenario***

## 4.5 CSCI Structure

The CSCI is decomposed into a number of CSCs. The CSCs correspond either to an application, or a class category describing a logically related set of functionality. The table below briefly outlines the CSC breakout of the PLANG CSCI.

***Table 4.5-1. Production Planning CSCs***

| CSC | Description | Type | |
|---|---|---|---|
| Subscription Editor | Interfaces to Data Server for submission of Subscriptions required so that Planning subsystem is notified of arriving data. | DEV | |
| Production Request Editor | Accepts Production Requests which describe an order for Data production, translates to Data Processing Requests which specify the PGEs which have to be run to service the Production Request. | DEV | |
| Subscription Manager | Interfaces to Data Server for notification of data availability, releases Data Processing Requests queued in Data Processing subsystem. | DEV | |
| Production Planning Workbench | Provides the ability to create, modify, and activate a plan for the scheduling of Data Processing Requests. | DEV | |
| On-Demand Manager | Handles the validation of an on-demand production request from the Data Server before passing it on to processing. | DEV | |
| Planning Object Library | A set of C++ class libraries that provides a framework for the Planning Workbench. | OTS | DELPHI (Hughes class libraries) |
| PDPS DBMS | DBMS Provides persistent storage for Planning data (e.g. PGE Profiles, Production Requests, Data Processing Requests, Plans etc.) | OTS | Sybase |
| Resource Planning Workbench | Allows the submittal of reservations for hardware resources and allocation of those resources to various activities (e.g. 'Production' or 'Maintenance'). Subsequent resource plan is used in production planning. | DEV | |

### 4.5.1 Planning Subscription Editor CSC

#### 4.5.1.1 Purpose and Description

This application provides the capabilities required to submit subscriptions to the Data Servers responsible for the storage of ingested data. Registration of a subscription at a Data Server is required for the Planning CSCI to receive notification when data arrive within the ECS. The submission of subscriptions may still be managed as an operator initiated activity during Release B, but the Planning Subsystem will automatically submit a subscription for any input data needed for a Production Request.

#### 4.5.1.2 Objects implemented by this component

PlDataTypeB

PlSubscriptionSubmitIF

#### 4.5.1.3 Candidate products

None. This is a custom component required to interface with ECS specific services.

### 4.5.2 Production Request Editor CSC

#### 4.5.2.1 Purpose and Description

This application allows the user to submit production requests that describe the data products to be produced. The application uses the PGE descriptions (profiles) entered during SSI&T in order to work out the tasks - Data Processing Requests - that, in sum, meet the request. The application provides the capabilities to add, modify and delete Production Requests, as well as review and modify the resulting Data Processing Requests. The production request editor is identified as a distinct application and separate from the workbench in anticipation that defining production requests will be a discrete activity, unrelated to the “planning” of these events.

The Production request editor may also be used by authorized operations staff to schedule Data Processing Requests derived from a production request directly to the Data Processing subsystem.

#### 4.5.2.2 Objects implemented by this component

PlAlternateDataGranuleNB

PlAlternateNB

PlDataGranule

PlDataSource

PlDataSourceFactory

PlDataTypeB

PlDataTypeCatalogue

PlDataTypeReq

PlDATRecord

PlDPRB

PlMetaDataChecks

PlOnDemandPRNB

PlOutputYield

PlPGE

PlPgeFactory

PlPRCollectionNB

PlProductionRequestB

PlProductionRequestUI

PlRoutineArrival

PlSubscriptionSubmitIF

#### 4.5.2.3 Candidate products

None. This is a custom component required to manage ECS specific entities such as the PGE.

### 4.5.3 Subscription Manager CSC

#### 4.5.3.1 Purpose and Description

The Subscription Manager is used to manage the receipt of a subscription notification from the Data Server. Subscription notification is used to notify Planning on the arrival of input data required by a given PGE, and of new Planning Data Availability Schedules and FOS Detailed Activity Schedules. Subscription Notification contains Universal References (URs) which are pointers to the data objects stored on the Data Server. The Subscription Manager updates the PDPS database to indicate when data become available and calculates new Predictions of the availability of data as schedules are received. When all input data for a Data Processing Request is available, the job defined for that Data Processing Request is released within the Data Processing subsystem.

#### 4.5.3.2 Objects implemented by this component

PlAlternateDataGranuleNB

PlDASDelta

PlDASDifferent

PlDASNB

PlDataAvailabilityTimes

PlDataGranule

PlDataSchedules

PlDataTranferHistory

PlDATRecord

PlDPRB

PlEDASModeRecordNB

PlEDASRecordNB

PlFOSDASFile

PlInstModeRecords

PlInstrumentModes

PlPDASFile

PlPDASMetaData

PlPDASRecords

PlPlanGenerationUIB

PlPopupMessage

PlReplanCriteria

PlSourcetoDsHistoryNB

PlSubMsgCb

PlSubscriptionManager

PlTimer

#### 4.5.3.3 Candidate products

This is a custom component required to interface with ECS specific services.

### 4.5.4 Production Planning Workbench CSC

#### 4.5.4.1 Purpose and Description

The application is used to prepare a schedule for the production at a site, and forecast the start and completion times of the activities within the schedule. These functions provided by the workbench include the following high-level activities:

1) Candidate Plan Creation—from the production requests prepared by the Production request Editor
2) Plan Activation—activating a candidate plan
3) Updating the Active Plan—feedback from the processing into the active plan.
4) Canceling/Modifying the Active Plan
5) Publishing and Import Plans - to share plans between DAACs and with the outside world
6) Cross-DAAC planning—identifying data dependencies between DAACs

As described previously, activating a plan entails rolling a portion of a selected plan into the AutoSys COTS. This “schedule” is then managed within the Data Processing subsystem. The forecast times generated within the planner are used to set up operator alerts that would make the operator aware of departures from the predicted schedule. The production planning workbench can periodically update it's predictions using feedback from the AutoSys.

The AM-1 Release has a number of additional features added to the production planning workbench to give the operator control over DAAC processing. A simple production strategy based on priorities will be implemented. Resource usage thresholds for on-demand products can be entered and modified. Criteria for initiating limited automatic replanning can be entered.

#### 4.5.4.2 Objects implemented by this component

PDPSDatabase

PlActivePlan

PlActivities

PlActivity

PlComputer

PlDASDelta

PlDASDifferent

PlDataDependencies

PlDataGranule

PlDiskPartition
PlDPRB
PlDPRs
PlEntryScreenNB
PlExportedPlanNB
PlFile
PlGroundActivity
PlGroundEvent
PlGroundEventAllocation
PlGroundEventExecutable
PlGroundEvents
PlImportedActivity
PlOnDemandExceed
PlOnDemandReplanValues
PlPDASFile
PlPDASMetaData
PlPDASRecords
PlPGEActivity
PlPGEPriorityNB
PlPlanASCIIReportFile
PlPlanB
PlPlanBinaryReportFile
PlPlanGenerationUIB
PlPlanMetadataFile
PlPlanningWorkbenchUI
PlProdStratNB
PlProdStratUINB
PlProductionPlannersUI
PlProductionRequestB
PlPRPriorityNB
PlPublishedPlan
PlReplanCriteria
PlReplanCriteriaUI
PlRescUseThreshNB
PlRescUseThreshUINB

PlResource

PlResourceChange

PlResourceManager

PlResourceRequirement

PlString

PlTimeLineDisplay

PlUserPriorityNB

#### 4.5.4.3 Candidate products

The Production Planning Workbench is built largely from COTS components. The planning workbench is built on top of a Planning Object Library that provides a framework for the application. This object library is provided by C++ class libraries which are described in the following section.

Job scheduling COTS are coming close to providing more capabilities that would meet some of the production planning capabilities, none of the packages at present meet the full requirements of the ECS. The ECS is committed to influencing vendor direction to leverage capabilities to cover additional ECS requirements.

### 4.5.5 Planning Object Library

#### 4.5.5.1 Purpose and Description

The Planning Object Library provides a reuse framework for building the Planning Workbench application. As described within the Planning Workbench CSC Section 4.5.4, there is no complete "off the shelf" solution that covers all the Production Workbench requirements, and therefore reuse of robust class libraries is the best approach for developing this application. A number of planning and scheduling frameworks have been evaluated for scheduling in the Planning Object Library; these frameworks are described in the Scheduling Engine Evaluation Trade. In general, these libraries provide similar capabilities. The Hughes Delphi Scheduling Class Software has been selected from those considered since it consists of generic, non-application specific libraries and there is a high degree of expertise within the ECS with using these Libraries, both within the Planning subsystem through prototypes and within FOS where these libraries are used in the mission planning.

#### 4.5.5.2 Objects Implemented by this component

The objects implemented by the Planning Object Library are considered COTS, similar to Rogue Wave objects. Therefore a list of these objects is not presented here. The CSCI object model presented in Section 4.3.4 showed an abstract view of the Production Planning Workbench. This was mainly due to the fact that the complexity of these planning and scheduling libraries would overwhelm the rest of the model and scenario descriptions. It is also the case that the capabilities could be implemented a number of different ways given different frameworks or COTS approach taken. The following sections first give an overall view of the Delphi scheduling class libraries and then show in detail how the Planning Workbench application is designed to reuse the Delphi libraries.

#### 4.5.5.3 Hughes Delphi Scheduling Class Libraries

The Hughes Delphi Scheduling Software is designed to assist in developing efficient and effective scheduling and planning. It is composed of a set of user-oriented, integrated, modular tools.

The toolset provides building blocks to allow developers to:

- Model system resources that reflect all relevant operating states and constraints.
- Automatically create coarse or detailed schedules for all system resources based on requests, using a variety of tailorable scheduling algorithms.
- Provide high functionality interfaces for planners to review and edit service requests, corresponding resource information, and generated schedules.
- Provide support for interactive development of contingency or impact studies (*What-Ifs*).

Delphi is based upon a system of distributed, modular components. Each component represents a distinct planning function, and each component can be plugged in, disconnected, or replaced as changing concepts, system needs or software upgrades require. In addition, each component has been engineered utilizing object-oriented methodologies. This provides many significant benefits, including:

- Functionality - The software can be easily tailored to implement application specifics.
- Extensibility - Once delivered the system can be easily enhanced without the need for complete replacement or extensive and costly block changes.
- Maintainability - All functions (data and process behavior) are well encapsulated in the software architecture. Therefore, should a problem occur it can be easily identified and isolated. Metrics for maintenance indicate exceptional savings for customers in O&M costs for this software.

These features combine to support a high degree of system flexibility and expandability, and in the long term contribute to a significantly lower system life-cycle cost.

All of the tools in Delphi are built on a foundation called the Hughes Class Library (HCL). This product provides a framework for all the objects in the system and provides generic functions for services such as time, collections (list, sets, arrays, etc.), stream input and output, inter-process communication, and windows displays.

##### 4.5.5.3.1 Delphi Resource Model

At the heart of Delphi is the Resource Model. In order to generate a timeline schedule for activities the Planning Workbench system must have detailed knowledge of both the activities that are to be performed and the resources that are required to be utilized or expended in order to complete each activity. Delphi's Resource Model supplies the structure to define these entities. In addition, the Resource Model acts as the owner of all data used by Delphi and therefore provides all data management services to the toolkit and the user.

The Resource Model consists of resources, resource states, and all relevant resource constraints.

*Resources*. All resources that are necessary for planning and scheduling are implemented as objects within the Resource Model. Application specific resource classes/objects can be derived from the resource classes provided in Delphi. These more specific classes/objects contain the attributes and behaviors that are unique to them.

*Resource State.* Besides modeling real world objects, another function of the Resource Model is to keep track of the state of all resources over time. Real-time and periodic (batch) updates of the state of all components of the system are sent to and stored by the resource model. For example, the state of a resource may indicate whether it is available or unavailable, and the nature of its current tasking.

*Activities.* In addition to resources and resource states, the resource model contains activities. These are schedulable entities that represent system tasking. During scheduling, resources are assigned to an activity. The resource's state is then updated to include each new activity. In turn, these updates can be directed as real-time modification to the timeline.

*Constraints.* Also in the Resource Model are the constraints present in the planning system. The scheduling algorithm will consider constraints between resources when attempting to properly allocate resources in a given plan.

The Resource Model defines, in both data structure and functional behavior, the resources being utilized. Any constraints concerning reasonable, proper, of safe behavior are defined to the system. The Resource Model also retains control over the definition of activities. These activities define a sequence of operations that are required to perform a high level goal. Both elements to scheduling, requests and resources, are managed within the flexible architecture of the Resource Model. Using these pools of information, the scheduling algorithms optimize the application of activities to resources to develop constraint-free plans for operational use.

Resource Model Hierarchies

There are two primary hierarchies within the Resource Model: whole/part activities and resource activities. The whole/part hierarchy provides an aggregation of many associated system objects; for instance, a “whole,” such as the plan, contains “parts,” such as resources and activities. In this manner. the Resource Model relates its individual inheritance hierarchies to create a global, consolidated, constraint-free plan.

A second hierarchy involves the individual resources and activities with which the planning and scheduling system must deal. In general, a resource hierarchy provides a detailed model of all relevant system resources. The activities are defined as resource-independent descriptions of operations that ultimately will be assigned to resources at specific times.

Each resource is responsible for maintaining a record of its state through time, for exporting algorithms of general interest, and for modeling and applying constraints. The resource state models what the resource is doing through time and may be different for each kind of resource. Resources know what plans are available and can have a different state on different plans.

Use of Resource State

Clients of the resource model can interact with resource state in several ways. A client can interact with resource state directly by asking a resource for all state on a given plan for a given time interval. The client would then have a detailed knowledge of the resource and can have as much knowledge of the resource and its inner workings as appropriate.

A client can also ask a resource for time intervals in which particular constraints are satisfied. These intervals may be displayed on another process, for example a timeline, in order to communicate to the user the window of opportunity for the resource. To obtain these intervals, the resource would iterate through all of its states and check with compatibility against existing state.

Resources can have a heterogeneous state. For examples, a resource could be tasked for a time interval and also be unavailable for another time interval. Both of these cases are supported by the resource state mechanisms.

Scheduling Activities to Resources

Resources that can be tasked can generally be asked to allocate (check constraints), unallocate (remove tasking), force allocate (do not check constraints), check allocate (check constraints but do not change state), and when allocate (tell me when constraints, if ever, are satisfied for a given set of tasking). This functionality is generally used by schedulers. For example, a sequential scheduler could order activities and then try activity/resource combinations until the activity was scheduled or until no combinations were left. The sequential scheduler will then move on to the next activity. Activities can be ordered by user-set priority, laxity, availability, or any other mechanism. The important point is that the scheduler only organizes the activities and orders the requests for resources. The resources themselves know what constraints to check and how to generate resource state. The scheduler may have to have resource-specific knowledge during the ordering process, or may use heuristics to optimize utilization of a resource.

#### 4.5.5.3.2 Delphi Architecture

The following section describes the software architecture used by Delphi. Delphi is composed of a series of modular *libraries* of software. Each library defines a series of groupings of both data and functions that are referred to as classes. The classes serve to provide the mechanism for data structures and manipulation functions to be encapsulated or localized, thereby providing discrete functionality that can easily be developed and debugged. These objects provide the atomic functionality to all Delphi tools.

#### 4.5.5.3.3 Hughes Class Library

The Hughes Class Libraries form the core from which all scheduling products are developed, thus providing real cost savings in development, test time, and maintenance costs. The Hughes Class Libraries are implemented in C++ according to the current standard, as set forth in the AT&T Version 2.0 C++ Programming Language Standard. The bottom layer of software usage is composed of X-Windows code, per the X11 standard. HCL utilizes X11 and allows the user application code to make direct calls to X11, if necessary. Within HCL is a display class of utilities that support the Motif display standards. These libraries have been in existence since 1990, and have been thoroughly tested and fielded in several operational systems.

The Hughes Class Library (HCL) is a library of C++ class declarations. These declarations are general purpose programming utilities, and include:

- display classes (XView, XGL, and Motif)
- collection classes
- Inter-process communication classes
- other miscellaneous utilities (e.g. string, rectangle, command line, etc. classes)

HCL contains several libraries: misc, dispx, mdisp, and ipc. These are summarized below:

misc Library

The 'misc' library supports the collections in linked list array, or set format. It provides iterators

over each type of collection for ease of movement throughout the collection. Collections can safely have multiple concurrent consumers. HCL allows classes derived from a common base class (HObject) to be stored in the same collection, thus allowing heterogeneous collections.

The choice of collection type (array, list, set) does not affect the application code. These collection types all are derived from a common class that established the protocol for the derived classes.

This library also contains classes that provide various date/time functions, command line information, timer functions, string functions, and vector and matrix functions.

dispx Library

The 'disp' library includes classes for drawing using XGL utilities. It has classes that encapsulate XGL contexts, rasters, fonts, and color usage. It also has classes to maintain display regions and collections of sub-regions.

mdisp Library

The 'mdisp' library supports displays with the look and feel specified in the Motif Style Guide. The library uses the Motif toolkit from the Open Software Foundation (OSF). The Motif toolkit is based on the X-Toolkit Intrinsics (Xt), which is the standard mechanism on which many of the toolkits written for the X-Windows System are based. The user will notice that the library encapsulates capabilities at all three levels (i.e. Motif, Xt, and Xlib).

The library provides classes that allow the user to create and manipulate X-Windows, color maps, events (i.e. keyboard, mouse buttons, window enter/exit), and user interface objects called widgets (i.e. menu bars, pulldown menus, buttons, scrollbars). There are classes that provide displayable regions, sub regions, rectangles, scalable fonts, strings, and colors. Mdisp also provides a class implementation of the graphics context (GC).

The DAppl class provides a template for a display application. It provides the behaviors to create, run, and destroy the application. The class does not provide a base frame.

ipc Library

The 'ipc' library contains classes that encapsulate interprocess communications, providing a simple interface for the programmer. It is an implementation of Berkeley sockets and XDR streams.

Messages derived from a common message class (HMessage) can be passed between processes without the ipc code knowing anything about the contents of the message. This isolates the code that does know about the message content, making for a much easier, faster development and easier maintenance.

#### 4.5.5.3.4 Delphi Reuse Libraries

The framework for all scheduling applications is HCL. In addition to this core product are reusable libraries of classes for virtually all aspects of a scheduling system. For purposes of discussion, these libraries will be grouped into four categories. These categories are: Scheduling, Resource, Timeline, and Planning Class Libraries. All libraries build upon the base of the HCL and are developed in the C++ programming language.

##### 4.5.5.3.4.1 Scheduling Class Library

The Scheduling Class Library (see Figure 4.5-1) provides a framework for the incorporation of scheduling algorithms.

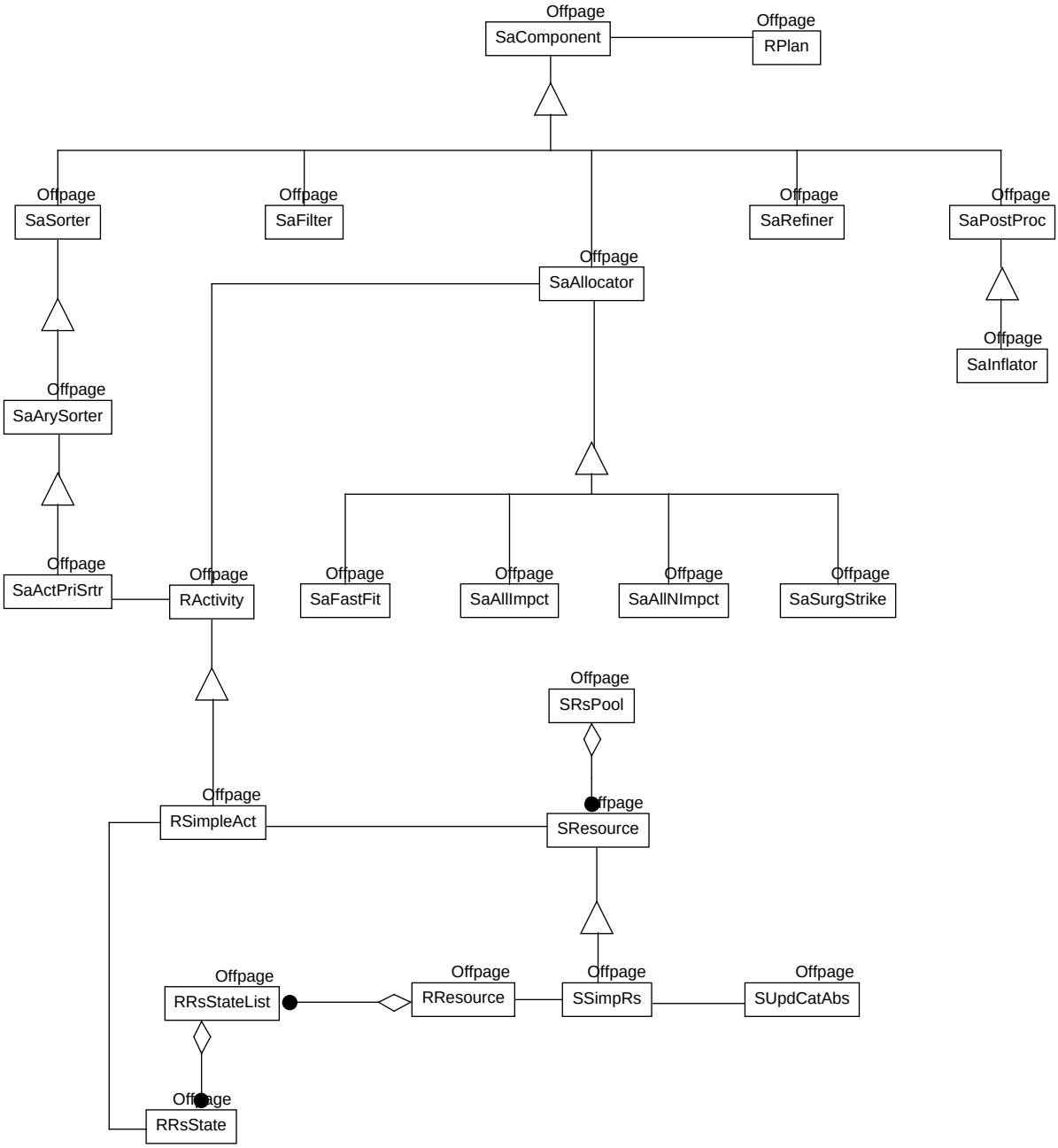



***Figure 4.5-1. Delphi Scheduler Object Model***

Reusable Classes:

SResource (class modeling resources with scheduling) - Each SResource specifies protocol for allocation of an activity to a resource for a time interval.

SRsPool (scheduling resource pool) - An instance of this class provides a storage mechanism for quick retrieval of scheduling resources (indexing by associated resource ID). Derived classes can provide special queries that are typically process non-specific. For example, fill a given collection with all scheduling resources of a given type that have no state for a given time interval.

SSimpRs (class modeling resources with mutual exclusion) - Each SSimpRs is associated with an RResource and implements allocation members that assume a mutual exclusion constraint.

SUpdCatAbs (resource model change notifier) - SUpdCatAbs is a base class that sets up protocol for registration of changes of resources, their states, activity changes, and plan changes. It also has protocol for flushing these change notices. Derived instances of this class might keep lists of clients interested in changes, and notify these clients when changes take place.

SaActPriSrtr (activity priority sorter) - This class customizes the sorter for activity priority.SaAllImpct (Impact Scheduler) - Instances of SaAllImpct know how to accept lists of activities and edit the resource model accordingly, using impact scheduling. The plan that is edited is the one already set by using the SaComponent member function plan().

SaAllNImpct (Non-Impact Scheduler) - Instances of SaAllNImpct know how to accept lists of activities and edit the resource model accordingly, using non-impact scheduling. The plan that is edited is the one already set by using the SaComponent member function plan().

SaAllocator (activity allocator) - Abstract class used to define generic protocol for allocating schedulers.

SaArySorter (array sorter) - This class uses the system sort algorithm to perform sorting.

SaComponent (scheduling component) - Abstract class used to define generic protocol for scheduling algorithm components (sorters, filters, allocators, refiners and post processors).

SaFastFit (fast fit allocator) - This schedule establishes protocol for fast fit scheduling. The algorithm used is a first come, first serve algorithm.

SaFilter (allocation filter) - Abstract class used to define generic protocol for filtering an area of a plan.SaInflator (allocation inflation post processing) - This scheduling component class tries to inflate allocations.

SaPostProc (allocation post processor) - Abstract class used to define generic protocol for post processing schedulers.

SaRefiner (allocation refiner) - Abstract class used to define generic protocol for refining an area of a plan.

SaSorter (activity sorter) - Abstract class used to define generic protocol for sorting a collection of activities.

SaSurgStrike (surgical strike allocator) - This schedule establishes protocol for surgical strike fit scheduling. The consumer can provide an actual allocation algorithm.

Customization of the Scheduling Class Hierarchy

The Delphi Scheduling Class Library provides a framework for incorporating a variety of algorithms in the Planning Workbench. The Planning Workbench reuses the classes the Scheduling which define protocols for sorting, filtering, resource activity generation, and refinement. It is the goal of the Planning Workbench to employ the Scheduling framework to construct algorithms which optimally allocate activities to resources, especially local disk storage.

#### 4.5.5.3.4.2 Resource Class Library

The Resource Class Library (see Figure 4.5-2) provides the structure for defining application domain resources and it builds upon both HCL and the Scheduling Class Library. The Resource Class Library provides templates to define resource models capable of managing discrete resource states and mechanisms for assigning resource availabilities to resources. In addition, inter-request correlations, such as pre-requisite, co-requisite, and post-requisite constraints, preferences, consumable resource modeling, and specific configuration requirements, are addressed. Each of these resource definitions will be used by the schedule deconfliction processes to provide a constraint free plan. When an activity is defined, multiple resources can be identified as being applicable to that activity.

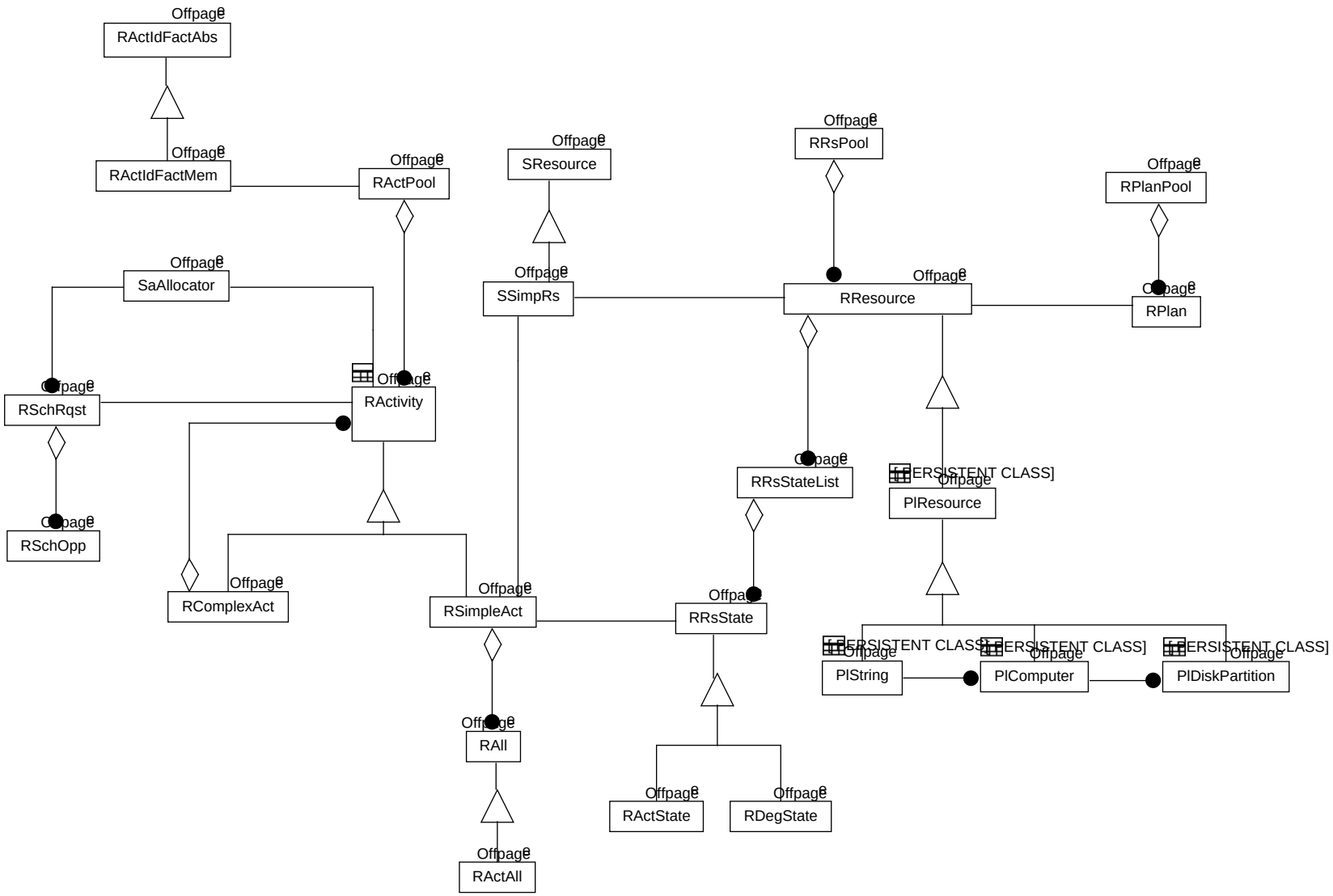



***Figure 4.5-2. Delphi Resource Object Model***

Reusable Classes:

RActAll (information about an activity allocation) - This class adds an activity to the base allocation. An example use is as a base class for modeling allocation of an activity to a bunch of resources. In that case, the derived class would have explicit resource support.

RActIdFactAbs (Unique activity id factory) - Instances of RActIdFactAbs are activity id generators. You can ask them for the next allowable activity id that can be assigned. This class has the notion of a global RActIdFactAbs object and provides a static nextId member function so that you do not have to have a specific instance everywhere you want to use it.

RActIdFactMem (Unique activity id factory) - Instances of RActIdFactMem are activity ID generators. You can ask them for the next allowable activity ID that can be assigned. This class initializes its activity ID range from the current activity pool.

RActPool (activity pool) - An instance of this class provides a storage mechanism for quick retrieval of activities (indexing by activity ID). Derived classes can provide special queries that are typically process non-specific. For example, fill this collection with all activities of a given type that have been allocated on a given plan.

RActState (tasking on a resource) - RActState is a resource state class that is generated by an activity.

RActivity (scheduling activity) - This is a base class for scheduling activities. It establishes protocol for all derived activities.

RAll (information about an allocation) - Each allocation has a time interval an optional plan name and a lock. If the plan name is empty, the allocation can pertain to all plans.RComplexAct (collection of activities) - Each instance of this class holds a collection of activities.

RDegState (tasking on a resource) - RDegState is a resource state class that represents a degraded resource. Presence of a direct instance of RDegState means that a resource is broken. Derived classes can represent how a resource is degraded.

RPlan (plan) - This is a base class that models the allocations of plans to resources.

RPlanPool (plan pool) - An instance of this class provides a storage mechanism for retrieval of plans (indexing by plan name). Derived classes can provide special queries that are typically process non-specific. For example, fill a given collection with all of the activities in the activity pool that have different allocations on two plans.

RResource (class modeling an entity with state through time) - RResource is a class that models an entity that has state through time. State is kept in RRsStateLists. Each RResource may have more than one RRsStateList. These state lists contain information which is mapped to a particular name (a “plan”). State lists usually contain things derived from RRsState. In other words, each resource object may maintain state information for multiple plans simultaneously. There is an additional, plan-independent state list that each RResource maintains. This state list may be used to hold plan independent information such as state that applies across all plans. RResource objects maintain a notion of a “current” state list.

RRsPool (resource pool) - An instance of this class provides a storage mechanism for quick retrieval of resources (indexing by resource ID). Derived classes can provide special

queries that are typically process non-specific. For example, fill a given collection with all resources of a given type that have no state for a given time interval.

RRsState (base class for resource state) - Abstract class used to establish protocol for any kind of resource state.

RRsStateList (resource state list) - An HEpochIntervalList with a name. An important difference between this class and an HEpochIntervalList is that this class empties() its contents upon destruction (unlike a vanilla HObjCollection, which clears() it's contents).

RSchOpp (a scheduling opportunity) - Each scheduling opportunity has a time interval and a resource id, representing an possible allocation of an activity to that resource and interval.

RSchRqst (resource scheduling request) - This class is a list of scheduling opportunities, which are specified as instances of RSchOpp, with an activity id specifying the activity to be allocated using the opportunities and a parameter indicating how many of the opportunities must be satisfied for the entire scheduling request to be satisfied. Only derivations of RSchOpp should be added to this class, and the add behavior of HObjList has been overridden to enforce this.

RSimpleAct (simple scheduling activity) - Each instance of this class models something happening on a resource in the system. Simple activities add allocations to the concept of Activity.

Customization of the Resource Model Hierarchy

The Planning design takes advantage of Delphi by utilizing the object-oriented mechanisms of inheritance. Elements of the Resource Model hierarchy have been customized for the Planning problem domain. Derived resources have been designed to customize resource state and override or add specialized algorithms and constraints. Some derived resources add scheduling operations which take activities and check constraints and generate resource state through time.

#### 4.5.5.3.4.3 Timeline Class Library

The Timeline Class Library (see Figure 4.5-3) provides mechanisms to support the displays of time-ordered information in a graphical manner. The generic Delphi timeline contains a two-dimensional, composite region displaying resources and their use versus time. Each display is composed of rectangular subregions that represent resources aligned down the vertical axis to be viewed over time, and time across the horizontal axis. Scrollable windows are utilized to allow the user to manipulate the displays to view the required resource profiles. Events for any period of time are modeled and displayed. Each event is indicated on the display with a text-label, color and state information, all of which can be customized for the specific application. The Timeline is a powerful visualization device allowing the user to navigate hundreds of resources profiles. Each event on the timeline can be compressed, expanded, edited, unallocated from its resource, locked and unlocked. Visually the user sees more detailed information the shorter the timespan being viewed, thus allowing a zoom cycle capability for all Timeline displays. Use of interactive point-and-click devices speed the user through birds-eye views of schedules down to the component parts and resources allocated to a single user request on the timeline. The timespan is adjustable and has no technical limits (although practical limits may insist only a portion of large schedules be viewed at any time).

The Timeline Class Library provides class implementations for drag-and-drop features between subregions, event manipulation, and supports multiple views of events. The use of color to convey importance, hierarchy, or groupings can be defined by the Customer, and color changing functions are provided with the Timeline classes. Scroll bars and multiple windows are supported. A time probe that anchors the center of focus of the display and provides default values of the time window of interest to other Delphi components, such as schedule Activity Editors, is also provided. The Timeline classes establish a generic structure for events.

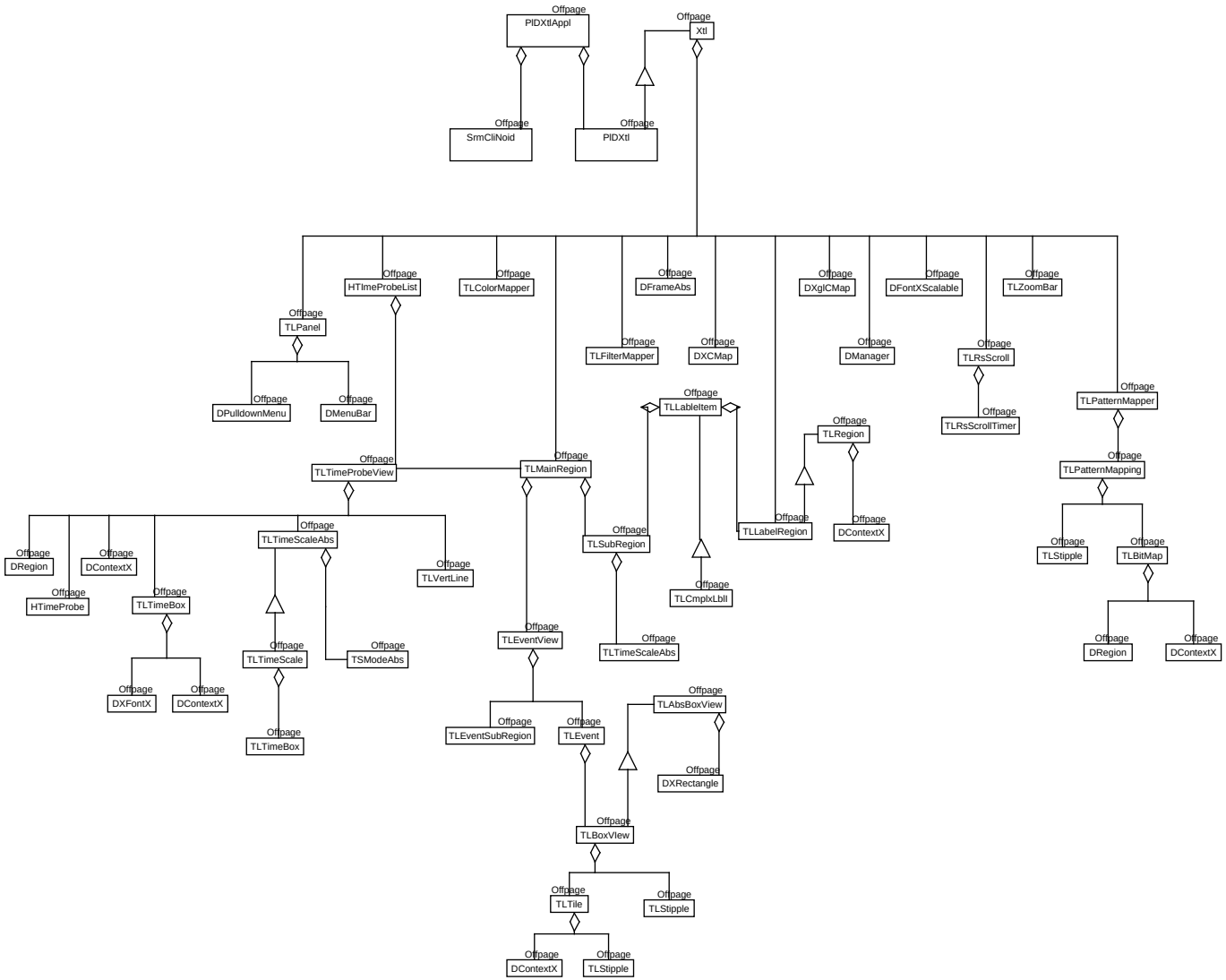



***Figure 4.5-3. Delphi Timeline Object Model***

Customization of the Timeline Class Hierarchy

The Timeline is one of the primary visualization tools within the Planning Workbench. The Timeline will provide a GANTT style view of the requested allocation of PGEs. In the Timeline, horizontal boxes will represent PGEs, and the boundaries of the boxes will represent start and end times. The Timeline enables the user to view in an instant the breadth of system resource utilization over time. Information displayed on the box (textual, color, graphical, etc.) will aid the user to search for individual PGEs. Several instances of the Timeline can be viewed simultaneously to compare the results of various scheduling algorithms.

#### 4.5.5.3.4.4 Planning Class Library

The Planning Class Library provides the display and integration structure to all Delphi scheduling applications. This library provides the interface slots to allow a developer to build up from the foundation of the components of Delphi into a coordinated scheduling environment. Additional capabilities provided within the Planning Class Library are: error message display and logging, distribution of resource states and coordination of schedule modifications, resource editing, scheduling activity creation, deletion, and dissemination, (planning synchronization across multiple users).

### 4.5.5.4 Use Of Delphi In The Planning Workbench

#### 4.5.5.4.1 Planning Workbench Architecture

Delphi takes advantage of a client-server model to allocate various display and algorithmic functions to separate physical processes. As applied to the Planning Workbench, this model is illustrated in Figure 4.5-4 below.

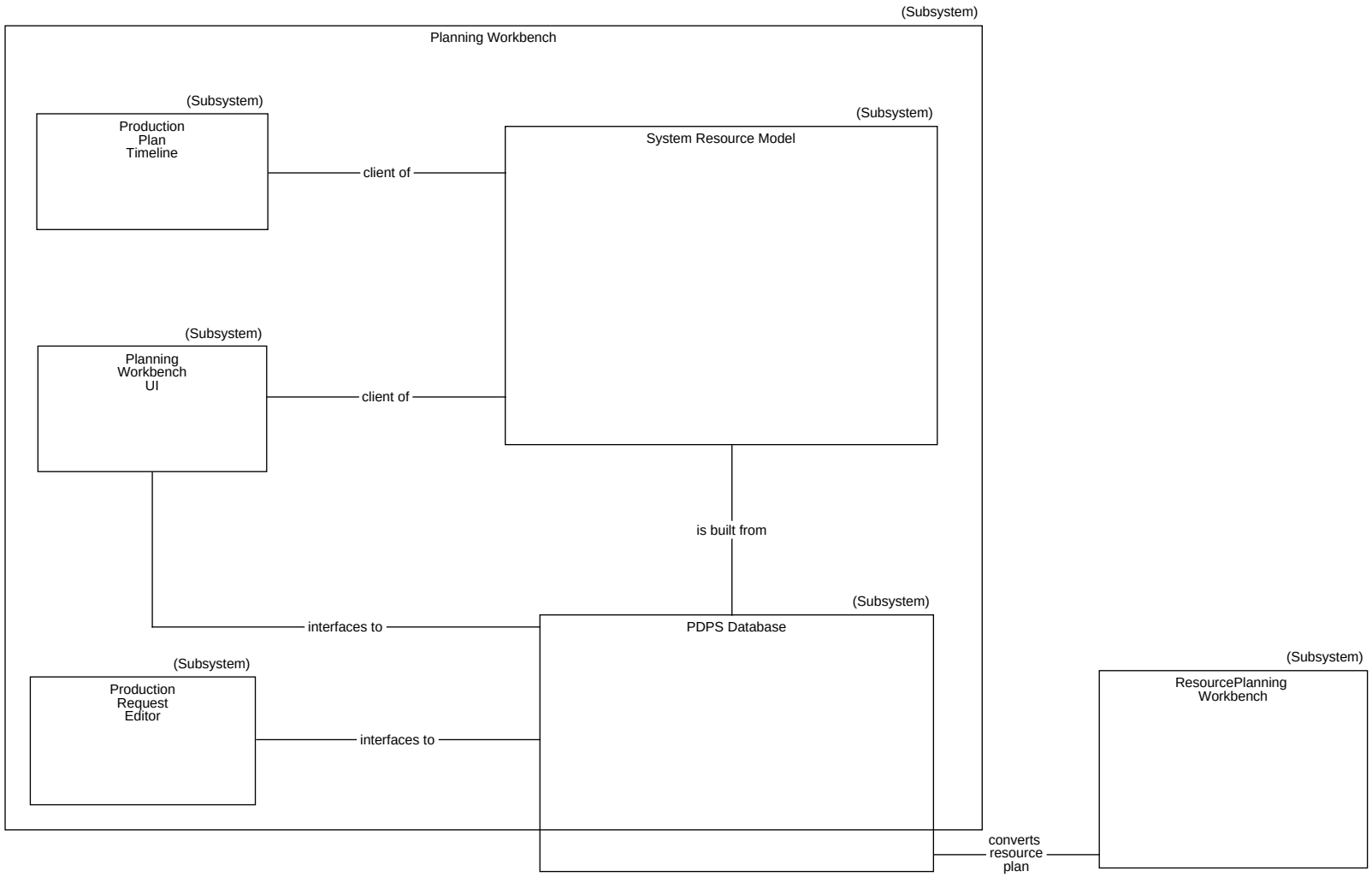



***Figure 4.5-4. Planning Workbench Architecture***

At the heart of Delphi and thus the Planning Workbench is the System Resource Model. This is the key server process in the Delphi client server model. All interfaces to persistent data required for the Planning Workbench are maintained within the Resource Model. The persistent data items relevant to the Planning Workbench are the Plans, Resources and Activities. The persistent data is maintained in the PDPS database. The Resource Model retrieves data from the database to describe the Resources and Activities and previously generated Plans, the Resource model stores data in the database to describe new or modified Plans.

The clients of the System Resource Model are also shown in Figure 4.5-4. The Planning Workbench UI is the key client within the Planning subsystem, this contains the planning algorithm responsible for generating the plan. The Planning algorithm is implemented within the framework of the Delphi System Scheduling Interface which encapsulates the protocol for planning algorithms.

The Planning Workbench interfaces with the Resource Planning Workbench to convert a Resource Plan into ground events in the Production Plan. These ground events are taken into consideration by the Planning Workbench and are planned around.

The Production Planning timeline display is shown in Figure 4.5-4 to represent the basic view available for the operators. This is the main graphical display used in the Planning Workbench showing DPRs planned on specific resources against time. This view is also available within the Resource Planning Workbench, but ground events are displayed instead of DPRs.

To complete the picture, Figure 4.5-4 also shows the Production Request Editor interfacing to the PDPS database. The production request editor is the originator of the Data Processing Requests objects within the database from which the planned activities are created.

The main areas of the Planning Workbench high level model that were presented as abstractions in Section 4.3 and are as follows:

| | |
|---|---|
| PlResource: | This class maps to a number of classes which are specializations of the Delphi classes. The PlDRResource class and PlDSResource class specialize from the RRsource SResource classes respectively. These classes are used within the Delphi System Resource Model as described in Section 4.5.5.4.2 |
| PlPlan: | This class maps to the PlDRPlan class, which is a specialization of the RPlan class within Delphi. The RPlan class contains basic operations to manage and copy plans, these are some of the key features that are implemented in Delphi which give considerable reuse advantage to the ECS Planning Subsystem. |
| PlActivity: | This class maps to the PlDRActivity, PlDRSimpleAct classes, specializations of the RActivity, RSimpleAct classes within Delphi. |
| PlResourceManager | This class again maps to a number of classes which are specializations of Delphi classes. The Planning subsystem specializations are PlDSsiScheduler, PlDSAl-locator, PlDSsiOppGen these classes are the key classes in implementing the Delphi System Scheduling Interface as described in Section 4.5.5.4.3 |

There are in addition a number of the utility classes for collections, lists etc. that are provided by Delphi, and the lower level HCL class libraries which will be reused and specialized within the Planning Workbench application.

In order to give a more complete picture of these methods and to illustrate the Delphi reuse this is described in more detail here, first by describing the System Resource Model, then the System Scheduling Interface within the Production Plan Editor responsible for performing the planning algorithm.

#### 4.5.5.4.2 Planning Workbench System Resource Model

In order to describe the Delphi scheduling approach the System Resource Model should first be described. The object model diagram for the System Resource Model is shown in figure 4.5-5.

The System Resource Model Srm class is responsible for maintaining the resource model for the Planning subsystem. It controls additions, deletions, and updates to the resource model and manages distribution of model changes to interested clients. When the Srm is initialized, all required pools are created: the resource pool, the scheduling resource pool, the activity pool, and the plan pool. The Srm then waits in the background for client messages which contain requests for information about or updates to the resource model pools.

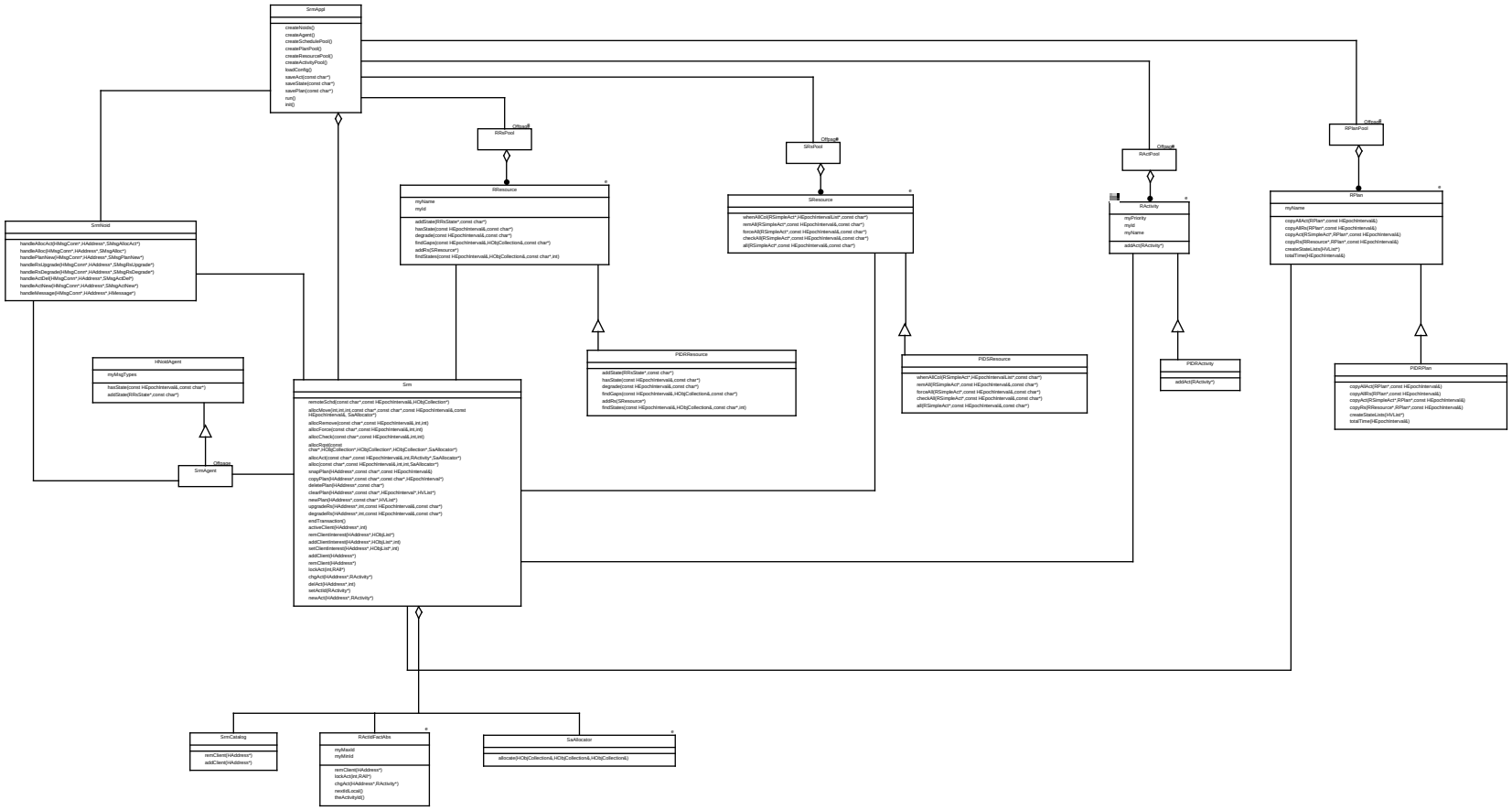



***Figure 4.5-5. Planning Workbench System Resource Object Model***

#### 4.5.5.4.3 Planning Workbench System Scheduling Interface

This model describes the key portion of the Production Plan Editor within the Planning workbench to describe the plan generation and resource allocation scheme. The System Scheduling Interface (SSI) tool provides an application framework for performing generic scheduling. The SSI contains a user interface which allows the user to select activities to schedule for a chosen plan. The SSI communicates with the System Resource Model (SRM) to request the current state of the resource model. The SSI maintains copies of the SRM pools for efficient local scheduling. After activities are scheduled on local copies of resources, messages are sent to the SRM to update the central resource model.

The object model diagram for the System Resource Model is shown in Figure 4.5-6. A generic scenario that shows how activities are allocated to resources is given below.

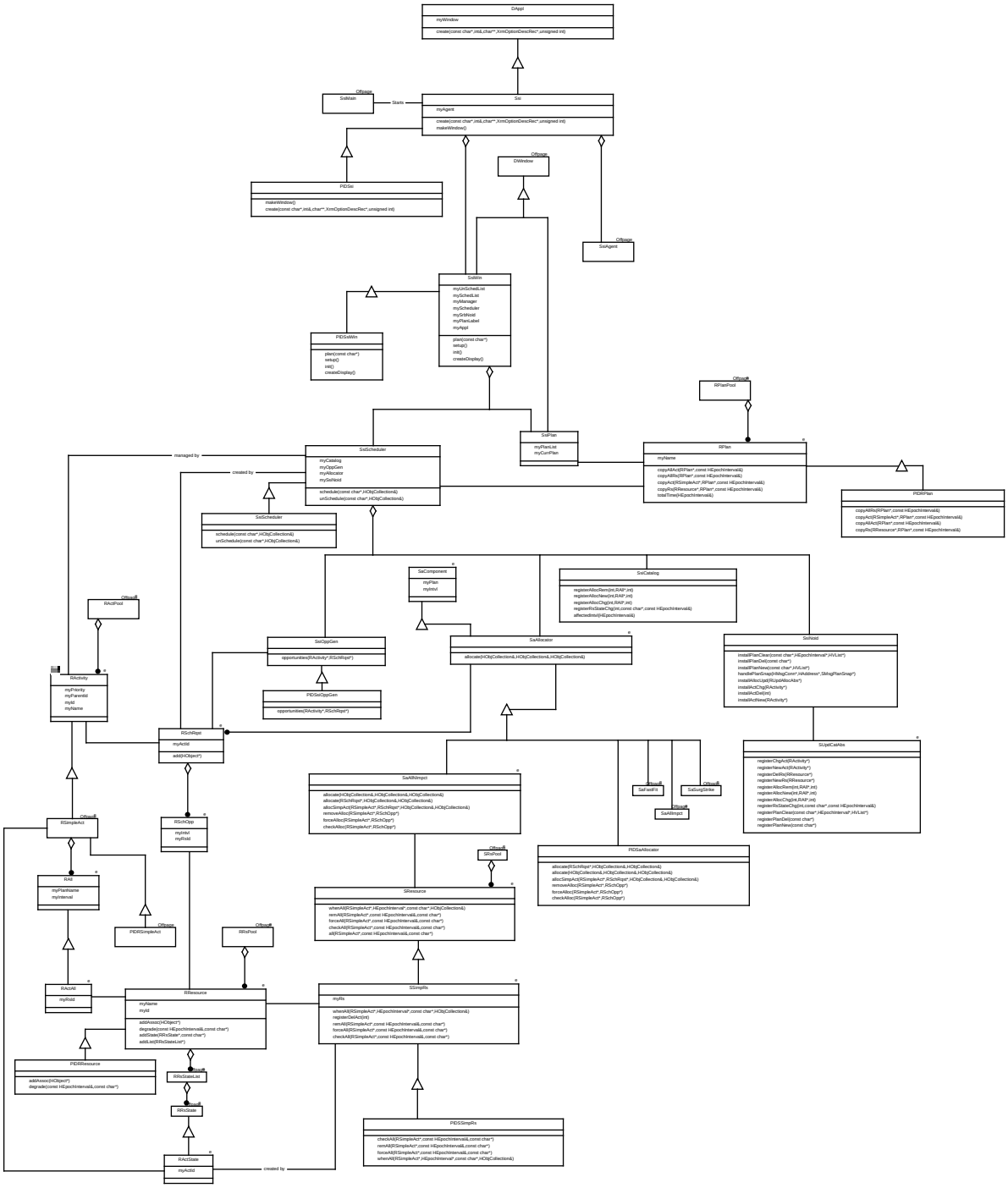



***Figure 4.5-6. Planning Workbench System Scheduling Object Model***

#### 4.5.5.4.4 Scheduling an Activity

##### 4.5.5.4.4.1 Abstract

This scenario describes the system response to a production planner scheduling an Activity (for example a Data Processing Request). After selecting a set of Production Requests the production planner will prompt the System Scheduling Interface (SSI) to schedule all the Data Processing Requests associated to the Data Processing. If possible, the activity will be scheduled on the appropriate resource and the System Resources Model will be notified of any changes to resources.

##### 4.5.5.4.4.2 Interfaces With Other Subsystems and Segments

##### 4.5.5.4.4.3 Stimulus

A production planner initiates the schedule function from the Production Planning Workbench.

##### 4.5.5.4.4.4 Participating Classes From the Object Model

The following are participating classes from the Object Model:

- PlDSsi
- PlDSsiScheduler
- RSchRqst
- PlDSsiOppGen
- PlDSaAllocator
- SsiNoid

##### 4.5.5.4.4.5 Beginning System, Segment and Subsystem State(s)

Steady state, up and running.

##### 4.5.5.4.4.6 Ending State

No change in the state of the system.

##### 4.5.5.4.4.7 Scenario Description

1. The production planner starts the planning workbench utility, the standard user authentication process applies (see scenario 4.4.17).
2. An PlDSsiScheduler is created
3. The PlDSsiScheduler iterates through an ordered list of activities to be scheduled.
4. The production planner selects an activity to be scheduled which is sent to PlDSsiScheduler. .
5. The PlDSsiScheduler creates an instance of the Scheduling Request (RSchRqst) which corresponds to the selected activity (RSimpleAct).
6. The PlDSsiScheduler creates an instance of an Opportunity Generator (PlDSsiOppGen).
7. The PlDSsiScheduler passes the Scheduling Request (RSchRqst) and the selected activity to the Opportunity Generator.

8. The Opportunity Generator finds a resource on which the selected activity can be scheduled. This resources information is stored in the Scheduling Requests (RSchRqst).
9. The PlDSsiScheduler creates an instance of a Resource Allocator (PlDSaAlloctor).
10. The PlDSsiScheduler passes the Scheduling Request to the Resource Allocator and asks the Resource Allocator to allocate the activity from the resource information stored in the Scheduling Request.
11. The PlDSsiScheduler notifies the System Resource Model (through the SsiNoid) of the changes to the resource on which the activity is scheduled.

#### 4.5.5.4.4.8 Event Trace

See Figure 4.5-7.

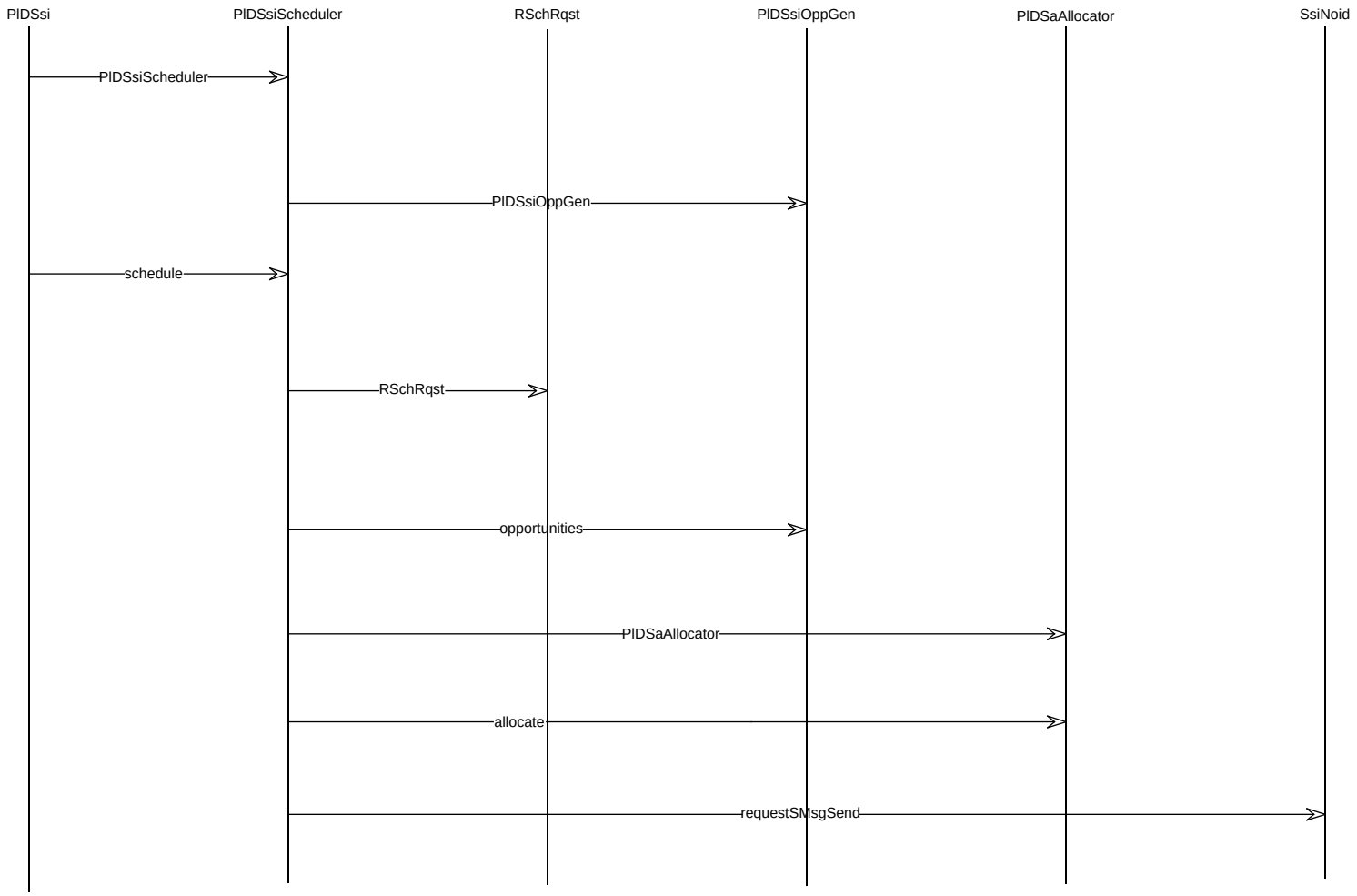



***Figure 4.5-7. Delphi Schedule Event Trace***

### 4.5.6 PDPS Database CSC

The Planning and Data Processing Subsystem (PDPS) requires that much of its data be stored in a repository in order for data processing and planning activities to be managed over time. This data is known as “persistent”, as it must "persist" beyond the termination of the programs that reference it. The repository for the persistent data is a relational database management system (RDBMS).

Objects in the OMT model having persistent attributes are manifested in the RDBMS as tables. The columns in a table represent the attributes of the corresponding object, and the rows correspond to individual instances of the object. In addition, operations that manipulate the attributes may be stored in the database as "stored procedures."

Associations between objects in the OMT model are manifested in the RDBMS as table relations. The most common type of table relation is the foreign key relation. Every table in a RDBMS has a primary key, i.e. one or more columns which uniquely determine a row in the table. A primary key of one table that appears as a column in some other table is called a foreign key in the second table. The presence of a foreign key creates an implicit link from the table in which it appears to the table in which it is the primary key. Such a link implies that operations affecting one table may also affect the other, thus reflecting the association in the OMT model.

The design of the PDPS RDBMS involves mapping the OMT model, which is a conceptual or “logical” model of the subsystem objects and functions, to a real-world or “physical” implementation of these objects and functions.

#### 4.5.6.1 Database Logical Design

The Logical Model of the PDPS, which uses OMT methodologies, is the model of record. This OMT model, however, is wholly logical. That is, it cannot alone serve as the blueprint for implementation-level work, as there is no direct correlation between the object-oriented concepts of classes, association, and methods to a RDBMS. Thus, a “physical,” or real-world, design equivalent of the object model must be specified.

The physical design equivalent of OMT in a RDBMS implementation is an Entity-Relationship Diagram (ERD). Persistent objects required by the PDPS subsystem are stored as tables in the RDBMS. Attributes of persistent objects are mapped to database table columns using an OMT-ER matrix. A catalogue of this mapping documents traceability from the object model to the ERD and includes any derived tables that are created in the process of this mapping.

The following logical rules are used during this mapping of OMT models to ERDs:

Rule 1 Objects without persistent attributes are not mapped to the ERD.

Rule 2 Objects with persistent attributes are mapped to tables.

Rule 3 Attributes associated with these objects are mapped to table columns. The maximum length of a column value must be specified. A column may be designated "null" or "not null," the latter indicating that a value must be supplied for it when a new row is inserted in the table. If a column is designated "null," then it may have a default value associated with it.

Rule 4 Every table has a primary key.

Rule 5 Any attributes having non-scalar data types, such as arrays or lists, are exploded into derived tables.

Rule 6 Any derived objects are annotated.

Rule 7 Associations in the OMT model are mapped to primary or foreign key relations.

#### 4.5.6.2 Transformation of the ERD to T-SQL

The information contained in the mapping of the OMT model to the ERD is entered into an appropriate ERD CASE tool, such as S-Designor Enterprise for Powerbuilder. The tool uses the information in the ERD to automatically generate low-level Data Definition Language (DDL) in Transact SQL (e.g. create table, create indexes, create primary/foreign key relationships, etc.).

#### 4.5.6.3 The Design of Normalized Tables

Normalized RDBMS tables are tables that meet widely-accepted database design criteria involving escalating levels of design constraint. At zero normal form all data resides in a single repository; there are zero constraints on how data is stored. Each level of normalization adds an additional constraint to information storage. For the PDPS database, the normalization rules in use will constrain the data to a third-normal form level of design.

A table is in third normal form if it satisfies the following three criteria:

1st NF: Any populated column in the table has a discrete value.

Result: A column in a table that is used to store information about input data can store information about one input data item or a pointer to a list of input items somewhere else in the database, but it cannot store a list of input data information.

2nd NF: Satisfies first normal form and has a primary key. Every column not part of the primary key is fully dependent on the entire primary key.

Result: Any row in a table can be uniquely identified by the column(s) constituting the primary key. If the primary key involves more than one column, then every part of the primary key is essential for distinguishing between any two rows in the table.

3rd NF: Satisfies second normal form and every column that is not part of the primary key depends directly on the primary key.

Result: There can be no partial or transitive dependencies for any non-key columns within a table. That is, it must not be possible to uniquely determine the value of a non-key column solely by referring to some other non-key column or columns without having to consult the primary key.

#### 4.5.6.4 Database Interface

The PDPS Database CSC includes interface classes responsible for mapping persistent data in object-oriented applications to the Relational Database Management System. These classes will be used to retrieve object information from the database, manipulate it, and store it back into the relational database. This interface also manages the connections between the client and server.

This interface is implemented using Rogue Wave's DBtools.h++, a C++ class library for interfacing with relational database management systems such as Sybase. A fuller description of this interface may be found in the Release B SDPS Data Processing Subsystem Design Specification for the ECS Project..

#### 4.5.6.5 Objects implemented by this component

PlDBMSProxyAgent

### 4.5.7 On-Demand Manager CSC

#### 4.5.7.1 Purpose and Description

The On-Demand Manager is used to manage the on-demand production requests received from the Data Server. Upon receipt of a production request the On-Demand Production Request class verifies that resource usage thresholds are not exceeded and that the data is available to generate the product. If all criteria is satisfied then the production request is added to the list of On-Demand requests maintained by the manager, the processing request is submitted to Processing subsystem and the product is produced. Based on certain threshold values a replan may be triggered. Additional possible On-Demand notifications are a request to modify or cancel a production request or a request for the status of a production request.

#### 4.5.7.2 Objects implemented by this component

PlDPRB

PlOnDemandExceed

PlOnDemandManagerNB

PlOnDemandPRNB

PlOnDemandReplanValues

PlPGE

PlPlanGenerationUIB

PlPopupMessage

PlPRCollectionNB

PlProductionRequestB

PlReplanCriteria

PlRescUseThreshNB

PlSubscriptionSubmitIF

#### 4.5.7.3 Candidate Products

This is a custom component required to interface with ECS specific services.

### 4.5.8 Resource Planning Workbench CSC

#### 4.5.8.1 Purpose and Description

The Resource Planning Workbench allows the Resource Planner to submit reservations for hardware resources and allocate and plan their use. Reservations may be made for activities that include major ECS data processing services as well as ground events. Ground events include activities such as maintenance, test, and training. The hardware resources include CPUs, storage devices, (sub)networks, as well as platforms that are made up of CPUs and storage devices. A list

of resources is configuration managed by MSS. Reports of planned versus actual resource usage can be generated and timelines for the resource plans may be displayed. This application is separate and distinct from the Production Planning Workbench. It's plan, however, is used as input to the Production Planning Workbench, so that production plans can take into account planned resource usage.

#### 4.5.8.2 Objects implemented by this component

PlComputer

PlDiskPartition

PlNetwork

PlResource

PlResourceConfigeration

PlResourceManager

PlRpActivityType

PlRpComplexResourceReservation

PlRpCPU

PlRpDevice

PlRpDiskPartition

PlRpExecutable

PlRpPlatform

PlRpPublishedPlan

PlRpResource

PlRpResourcePlan

PlRpResourcePlanningUI

PlRpResourcePool

PlRpResourceReservation

PlRpResourceReservationPool

PlRpResourceState

PlRpResourceTimeline

PlRpService

PlRpString

PlRpSubNetwork

PlService

PlString

#### 4.5.8.3 Candidate products

This is a custom component required to interface with ECS specific services.

## 4.6 PLANG CSCI Management and Operation

The following sections discuss the management and operation of the Planning CSCI. Specifically, the manner in which the CI is managed at the local and system levels is discussed. The approach to the development of operator interfaces for the Planning CI is then described. Finally, the approach to reporting for the Planning CI is described.

Before examining management and operation of the Planning CI at the system level, the local level management view of this CI is considered. That is, the operations and management of the CI with short range considerations. These include the subscription submittal activities, the production request editor, and the AI&T-developed PGE profile editor. These CSCs are used to create or update data structures local to the planning process. With longer range implications are the Subscription Manager, the On-Demand Manager, and the Planning Workbench.

### 4.6.1 PLANG CSCI Operation

A key consideration to remember is that operations personnel will not be required to routinely insert individual data processing requests as has been done in the past for some systems. That is, once the instructions for processing a type of data with a particular PGE have been inserted into the Planning database, the error-prone, repetitive entry of data processing instructions for each data product is avoided. Data processing will be primarily data driven. Plans are constructed based upon the data that is expected to arrive from external sources or based upon the intent to reprocess existing data with revised algorithms. As data arrives at the DAAC, the Planning subsystem will be made aware via subscriptions of the existence of the data and processing will begin when all necessary data files have arrived and are staged.

<u>Production Request Editor</u>

Typical of local management activities is the Production Request Editor CSC which provides the capability to define and modify PRs that are used as a part of the planning process. This interface is used by operations personnel to enter the production requests that are the starting point of the planning process. The PRs do not need to be added/created for each job to be run, but only for each job type or class to be run. The actual job stream or script is built up from this using information contained in the planning database, and submitted as a part of the planning and processing activity.

<u>Subscription Submittal</u>

The Subscription Submittal CSC permits the operations personnel to efficiently enter subscriptions to Data Server/Ingest for one of the types of data needed by a PGE for processing. Subscriptions for Planning Data Availability Schedules (PDAS) and FOS Detailed Activity Schedules (DAS) are also made. This CSC is a non-persistent or utility type application that is typically used during the algorithm integration and test phase for the PGE. At that time, the planning database is initialized with the information needed to plan and schedule the execution of the PGE during normal operations. After the subscription has been entered, no further operator interaction is ordinarily required to insure that the subscription notifications are transmitted to the Planning CI. This CSC allows operations to view current outstanding subscriptions, and to modify or delete them as required. This use of the CSC is no longer required for most Datatypes since the Production Request Editor and the On-Demand Manager automatically create and delete subscriptions as necessary. The only data that still need manual subscription entry are PDASs and DASs.

Planning Workbench

The Planning Workbench CSC includes a collection of utilities to enable the operations personnel to perform the planning function for the DAAC. These include capabilities to create candidate plans, activate a candidate plan, update and cancel an active plan. This utility allows the operations personnel to have control over the production. Operations personnel control on-demand production requests through resource usage thresholds. Priorities are assigned to data processing requests through production strategies, which are entered in the Planning Workbench CSC.

The Planning Workbench CSC allows operations personnel to devise plans for routine production data processing and reprocessing. The production scheduling personnel selects Production Requests from the Planning database and includes them into the planning process. Candidate plans are developed based on the predicted availability of the necessary input data, PGE processing characteristics, planned availability of resources at the DAAC, and priorities and policies established by the local DAAC management. The Planning Workbench CSC allows the production scheduling personnel to create multiple candidate plans for evaluation, select the candidate plan that conforms best with the DAAC system operations plans, and to retain the plan for future modification. The Planning Workbench provides the capability to activate the current plan which thereafter guides the activities of Processing. This CSC also includes the capabilities to update or modify an existing active plan or to cancel an active plan.

In addition to being able to view plans from the local DAAC, operations personnel can import plans from other DAACs. The Planning Workbench CSC will identify all data dependencies between the plans, so an operator can easily identify any conflicts.

A concept of the Planning Workbench CSC is to take advantage of the human capabilities to distinguish a 'good' plan from a less optimal plan. The capabilities provided here allow the operations personnel to generate multiple candidate plans for comparison and evaluations. The operations personnel can devise these plans by varying certain system planning strategy parameters such as priority allowing the planning manager to automatically construct several plans and select from among them the one that is most suitable for the needs of that DAAC.

Another feature of the Planning Workbench CSC is the Limited Automatic Replan capability where the operations personnel can be notified when an event occurs that might suggest a replan. Events such as notification of late data arrival, a change in a Resource Plan, or an excessive number of On-Demand Requests can trigger this notification to the operator. All of these notifications can be turned off, and all have operator-specified thresholds that can be set. The goal is to increase the automation of the Planning Subsystem so these events don't need to be constantly monitored.

A key feature here is that each DAAC may construct the planning procedures that most readily fit with existing operations procedures. For example, a planning cycle operations concept developed earlier suggested that a 30 day and a 10 day candidate plan could be prepared by the DAAC operations personnel to establish basic long and short term plans for processing objectives. The Active Plan is assumed to be prepared from the latest 24-hours of the 10 day plan. However, this concept of operations is only one possibility. The operations personnel at each DAAC can prepare the processing plans that meet their needs best, e.g., a sequence of one week plans might be desirable for another DAAC. In addition, the operations personnel may easily adjust the priorities associated with a type of processing (routine, reprocessing, or, for the AM-1 Release, On-Demand processing) a type of user, PGE type or with individual products. This capability allows the

operations personnel to quickly respond to changing priorities determined through NASA procedures.

During the planning activity, the Planning CI attempts to allocate data processing jobs to one of the several processors that it has knowledge of as being available for processing. The Planning CI may allocate a job to a specific processor or to one of a pool of processors, depending on the needs of the particular algorithm. In general, a pool of processors is available for data processing. However, processors may be removed from service occasionally for a maintenance or upgrade operation referred to as a “ground event”. The Production Planning Workbench is made aware of these ground events via the latest Resource Plan. This Resource Plan is created using the Resource Planning Workbench as described below. The Planning Workbench takes the ground events planned during resource planning into account when it creates its Production Plan, and plans around them.

Subscription Management

The Subscription Manager CSC manages the interface with Data Server/Ingest for the receipt of subscription notifications. These notifications are provided to Planning as a part of the process leading to the activation of PGEs. The notifications indicate to Planning that a granule of data needed by a PGE has become available at Data Server/Ingest, for example, as the result of a Level 0 data set arriving from SDPF or EDOS. When all data required for processing has arrived, the Planning CI will release the PGE to the Processing CI for activation.

This approach employs the common subscription notification scheme. It automates the process of communicating with Data Server/Ingest CI for potential data arrivals. The Subscription Management CSC is persistent - it is continually active and awaiting notifications. It isolates the interfaces to Data Server/Ingest to a single element for receipt of notifications, simplifying the management of the system. Predictions on the arrival time of required data are generated by Planning, using the FOS Detailed Activity Schedule, empirical EDOS processing times and resource management information. These are used to identify delayed data arrivals, allowing the operations personnel to react to and resolve the situation. In addition, in order to produce a realistic plan, the DAS is also used to extract mode information since certain PGEs may have different run characteristics based on the instrument mode.

On-Demand Manager

The On-Demand Manager CSC manages the handling of an on-demand production request from the Data Server, passing it on to processing for the production of the requested product. An on-demand production request is considered to be any non-routine production request for which the system is tasked. Upon receipt of a production request the On-Demand Manager verifies that the input data is either currently available or is scheduled to be available. The On-Demand Manager also checks that the resource usage time necessary to generate the product is not more than is allowed for a single on-demand production request, and further that the cumulative usage time for the resources used does not exceed a certain threshold for the entire processing period. If all the above tests are passed satisfactorily then the production request is submitted to processing and is added to the production queue to be produced. Additional requests can be made to modify or cancel a production request, or to status a request.

Resource Planning Workbench

The Resource Planning Workbench CSC provides the capabilities for allocating hardware resources to the major ECS data processing services and to other activities. These other activities include maintenance, test, etc. and are sometimes referred to as ground events. Hardware resources are platforms (made up of a list of CPUs and disks) , (sub)networks, storage devices. The list of possible hardware resources and their approved usages is maintained under configuration control in the MSS Configuration Management (CM) System. The Resource Planning Workbench copies this configuration from MSS into its own database and allows for configuration changes. In addition to providing functionality for allocating resources to activities, capabilities are provided to allow resources to be shared by multiple activities, to publish resource plans, to generate reports showing the planned versus actual resource usages, and to run executables (or scripts) as pre or postprocesses to the actual activities.

The planning / allocation of the resources to services and activities is the responsibility of the Resource Planner (operations staff). The Resource Planner determines the allocation using resource requests submitted by the various parties who have needs for DAAC resources, e.g. for training, test or routine production. When this process has been completed, the resource plan is committed. The commit process makes the plan available to the Planning Workbench CSC so that the activities impacting processing can be incorporated into the production plan.

A principal of operation underlying the planning of resources is that resources are routinely allocated to specific system services. For example, the processors that are used for production processing are normally allocated to that purpose on a long term basis. It is on an exception basis that they are allocated to other activities.

### 4.6.2 System Management Strategy

The system management strategy as revealed by the PLANG CI is discussed in the following paragraphs.

#### 4.6.2.1 System Management and Operations Philosophy

The primary design consideration for the Planning CI from the point of view of system management and operations is that operator interactions with Planning be simplified and automated as much as possible. In addition, the Planning CI provides operations with the needed flexibility to respond to unexpected tasks as they arise. In part, the objectives of simplification and flexibility are accomplished through well designed GUIs that aid in developing the needed timelines.

#### 4.6.2.2 PLANG CI and the System

The Planning CSCI is a part of the ECS System. It is integrated with the other ECS components via the common management interfaces to participate in the ECS enterprise. The following paragraphs address certain system management topics from the point of view of the Planning CSCI.

<u>Startup & Shutdown</u>

The Planning CSCI is integrated with the system management interfaces for startup and shutdown. Via these interfaces, the operations personnel will control the initialization of the Planning CSCI including planning database initialization. The controlled startup and shutdown insures that the

necessary coordination and synchronization of activities, within the Planning CSCI and with other subsystems, is enforced. Adhering to these procedures ensures the highest availability and reliability for this critical component of the ECS system.

Fault Handling and Recovery

The Planning CI is integrated with the system management components to address fault handling. By providing these fault messages to the appropriate operations personnel via a unified interface, the consistent handling of component and system faults may be addressed. This permits the personnel responsible for address faults to inspect the information at hand from across the system and to aid him/her in identifying the appropriate cause of the fault. This scheme insures that faults are addressed quickly and accurately, enabling the planning and processing of science data to proceed without unnecessarily long delays.

System Maintenance Operations and Ground Events

Of particular importance to the Planning CSCI from the point of view of system management is the scheduling and coordinating of ground events (e.g., maintenance activities, software upgrades, or for testing activities) with processing. The data processing resources are, from the point of view of system management, another resource that will occasionally be required for ground events. From the Planning CSCI point of view, these resources are typically dedicated to the science data processing activity and are therefore mostly reserved for planning and processing. The Planning CSCI supports these views of the data processing resources through the planning process. The system Resource Manager will identify (usually well in advance) the ground event that requires the resources that are typically dedicated to the planning and processing activities. The Resource Manager provides these requirements to the Planning CSCI for incorporation with its planning activities. These ground events are dealt with by Planning as high priority activities whose resource requirements are to be satisfied. After these events are planned for, the time allocations are locked-in, not to be changed by subsequent planning and scheduling actions.

In a similar manner, as new hardware resources are introduced into the system to support science data processing, these resources are identified to the Planning CSCI as potential data processing platforms but which are indefinitely scheduled out-of-service until the resources are fully integrated and tested and released for use by processing. During the I&T phase for these new platforms, they are scheduled accordingly. In this way, the Planning CI cooperates in the system-level management of the data processing resources.

#### 4.6.2.3 Interfaces

The interfaces of the PLANG CSCI and the support and coordination required for system management and operations is discussed in the following paragraphs.

MSS

The Planning subsystem interfaces with the MSS for management related activities. The Planning subsystem is responsible for sending MSS fault management data and resource utilization data. The Planning Subsystem will exchange mode management information, and will receive event notifications from MSS. In addition, for Resource Planning, the Planning Subsystem will receive a configuration management resource information with which to initialize it's database from MSS as well as the actual times of ground events from the MSS log.

Data Server/Ingest

The Planning subsystem interacts with the Data Server/Ingest via the subscription interface to achieve the system objectives of responsive and efficient data processing. The subscription notification scheme for interfacing with Data Server/Ingest is a very flexible and robust approach for signaling the presence of data to be processed. It removes operations effectively from the loop of coordinating transfers of data. This simplification greatly improves the reliability of an interface requiring operator interactions. An additional function of the Data Server interface is the transfer of On-Demand production requests for processing. Finally, the PLANG CI interface with the Data Server/Ingest for staging of data prior to processing is an automated approach that, again, removes operations from the roll of handling data transfers thereby making more reliable and timely the movement of data for data processing.

Processing

The major interfaces with the PLANG CI are to the Data Processing subsystem's PRONG CI which are in most cases implemented through the planning database. Interfacing through the database provides a simple and reliable scheme for the transfer of schedule information from one CI to the other. The separation between the planning activity and the processing management activity insures that as the requirements and technology evolve on either side of the separation, the impacts to the other side of the interface can be controlled and limited.

AI&T

The AI&T activities in relation to the Planning activity are limited. AI&T activities populate portions of the Planning database as a part of the installation of the science software at the DAAC. This pre-operational activity provides for the people intensive effort of understanding the planning and processing needs of the science software and then imbedding that extracted information into the planning databases.

### 4.6.3 Operator Interfaces

This subsection describes the operator user interfaces provided by the Production Planning CSCI to operations personnel. A general description of the framework and methodology employed for the development of these interfaces can be found in Section 6.5 of the detailed design overview (305-CD-020-002). This subsection augments that information with additional design information which is specific to the Production Planning CSCI.

The operator user interfaces will be developed from a combination of COTS graphical user interface (GUI) libraries and custom code. Hughes Delphi Scheduling Library will provide some components for reuse, such as the Timeline, but these will have to be customized with Production Planning code. In addition, custom graphical interfaces will be created with the aid of Integrated Computer Solutions' Builder Xcessory. Builder Xcessory enables the developer to manage Motif graphical user interface projects by providing a WYSIWYG, drag and drop, visual develop environment. Once an interface is constructed, Builder Xcessory will generate C++ code which represents the GUI and encapsulates the C-based Motif Widget set. The generated C++ code can then be combined with other Production Planning specific code.

#### 4.6.3.1 Off-The-Shelf Interfaces

As discussed in Section 4.5.5, the Production Planning CSCI will reuse part of the Hughes Delphi Class Library to develop graphical user interfaces for the Production Planning subsystem.

#### 4.6.3.2 Production Planning User Interfaces

This section is intended primarily to describe the data that may be displayed to facilitate the Operations of the Planning utilities. The precise determination of the GUI will be decided by surveying the wishes of the users of the interfaces and demonstrating prototypes. The discussion below does describe some interaction between the operator and interface, this discussion is only preliminary and used to illustrate one of many possible arrangements.

Production Request Editor

The Production Request Editor is a utility which provides operations personnel with the capability to define and modify Production Requests (PRs) that are used as part of the planning process. The GUI for the Production Request Editor will be constructed with the aid of Builder Xcessory and combined with custom code which interfaces to the Planning database for storage of PRs.

Upon invoking the Production Request Editor, the operator may be presented with a main window which serves to create new PRs and display stored PRs. The information to be displayed consists of two parts:

1) A product selection sub-section for choosing a unique product

   - Any of the information associated with the products that is stored within the PDPS database may be displayed to facilitate the selection. This information includes Product Name, Product Version, Instrument, Platform for the product. A table format may be suitable for this information. The default PGE that generates a selected product may also be displayed.

2) A Production Request sub-section for defining the PR to generate the selected product.

   - A production period which consists of starting and ending dates and times is required. In addition, an editable text field displays the priority of the PR. The operator can accept the default priority for the PR or enter a new value. Once the PR production period and priority are specified, the operator is encouraged to enter a comment to aid in uniquely identifying the PR for later recall.

3) A Production Request sub-section for defining Production Request Collections and determining the cost of the collection in terms of resource utilization.

   - This feature is especially useful for Reprocessing Requests since it will allow the operator to specify a set of Production Requests to include in a reprocessing campaign. The costing feature will help to determine the impact of a large (e.g., 6 months) Reprocessing Request on the system.

If the operator chooses, he or she can override the default PGE by selecting another version. The operator would be presented with a new window which provides a list of PGEs corresponding to the selected instrument and product type.

The operator can invoke a dialog for overriding specific parameters for the chosen PGE. A dialog can present the operator with a list of parameters and values for modification.

There is also a utility for reviewing, editing, and deleting previously defined PRs.

The second set of functions available from the Production Request Editor is to review / modify the Data Processing Request which fulfill the Production Request. This interface is being prototyped within the IR-1 development and may be used to illustrate the data that can be displayed.

Production Strategies User Interface

The production strategies user interface allows the operator to have control over the strategies used to determine priorities of the Data Processing Requests that are in a plan. Each plan will have a particular set of production strategies that are keyed to it, but the same production strategies can be used in multiple plans.

Production strategies work on two levels. First the operator can update lists of DPR attributes (such as PR type, user type or PGE) so that each value an attribute can have is tied to a particular priority. Next the operator can change the weight that each attribute's priority is given. A weight is also given to the priority selected by the user who entered the request. In this way, a priority can be calculated for each DPR. Finally, there are a number of deltas that can be added to the calculated priority based on certain conditions. For example, an operator may select to increase the priority of all jobs that produce data that is needed by other DAACs or that have been waiting in the Production Queue for more than a day. The final priority of the DPR is used when producing a candidate plan.

Resource Planning

There are two GUIs for the Resource Planning which include windows for entering resource reservations and a timeline for displaying resource plans. To initialize the resource configuration using MSS configuration management tools requires a simple button push from the operator using the Resource Planning Workbench.

When the operator invokes the Resource Reservation Editor application, a window appears which allows the operator to specify a new resource reservation. The operator fills in the event description, the activity type, the start and stop date/times, the request priority, any comments and the list of resources needed by this ground event; the system fills in the originator's name and the reservation identifier. The sponsor for this request uses this same tool to validate it; the systems fills in the sponsor's name. Finally, the resource manager approves the request again using this same tool.

The Resource Planning Workbench is again used to commit the approved requests and generate a timeline display of the Resource Plan. See the Production Planning Workbench for a more complete description of the Delphi Timeline displays. A report can be generated of the resource reservations in the Resource Plan. Like a Production Plan, the Resource Plan can be published to the Data Server to be viewed by outside parties.

Production Planning Workbench

The GUI for the Production Planning Workbench includes windows for creating plans and a timeline for displaying plans. When the operator invokes the application, a window appears which allows the operator to specify the name of a new plan or select a plan which was previously created. To create a new plan, a window is provided to define the plan parameters. The operator first enters the plan name and then enters the starting and ending dates and times. In addition the operator can enter comments to describe the plan.

After the plan parameters are initially defined, a window may be presented which allows the operator to select the Production Requests to be included in the plan. This window could include two lists: a list initially full of unscheduled Production Requests and a list initially empty of scheduled Production Requests. The operator can choose Production Requests and use arrow buttons to move the selected entries back and forth between the scheduled and unscheduled list.

A timeline window is available to display scheduled events in a graphical format. As described in previous sections, the timeline is built on the hierarchy established in the Hughes Delphi Timeline Class Library and is tailored to Production Planning needs. The timeline is a GANTT chart which displays resources and their use versus time. When the timeline appears, it is first populated by ground events which have previously been defined by the Resource Reservation Editor. As the operator schedules Production Requests for a plan, the corresponding DPRs appear in the timeline window. If the operator unschedules a Production Request, the corresponding DPRs will be removed from the timeline. See Section 4.5.5 for a further description of the timeline.

Several instances of the Production Workbench can be invoked to compare the results of different plans in order to select the best plan to submit for production. Furthermore, plans from other DAACs can be imported and displayed. The Production Planning Workbench will identify all data dependencies between these plans so that any conflicts can be resolved by the operator.

On-demand Resource Usage Threshold User Interface

This GUI allows the operator to update the resource usage thresholds for on-demand production requests. There will be different thresholds for different resources (such as CPU time or disk space). In addition, the operator can update the threshold for a single production request, or the cumulative threshold for all on-demand production requests received within a particular time period.

Subscription Request Editor

The Subscription Request Editor contains a window from which an operator can enter product subscriptions to the Data Server. Two lists could be provided: one list containing unsubscribed products and the second list contains subscribed products. With the use of arrow buttons, the operator would be able to select and move products between the subscribed and unsubscribed product lists. In the AM-1 Release, it will not be necessary for an operator to manually submit subscriptions. Subscriptions for production requests will be automatically generated after a new Production Request is entered into the Production Request Editor.

### 4.6.4 Reports

A variety of ad-hoc and canned reports will be available to the DAAC operations staff to assist in monitoring of the activities associated with the Planning CSCI. These reports are readily accessible given that Planning CSCI persistent data is maintained in the PDPS Database, a SYBASE RDBMS. Also, ECS application management information is maintained in the MSS database, which is used to log system events. The canned reports will include the following:

1) Candidate Plan Characteristics Report- This report will be associated with each generated candidate plan and will contain summary information to be used to establish the quality of the produced candidate plan. Information contained in this report will include the following:

a) Definitions of inputs, i.e. Resources, Production Requests, and Priority Information, and other planning data used to create the candidate plan.

b) Candidate Plan planned results. This will include the number of science data products produced, number of DPRs executed, estimated time to meet all planned activities, estimated resource utilization.

2) Production and Data Processing Request Status Reports - These reports will indicate the status associated with a Production Request and its associated Data Processing Requests. This information would indicate whether a given Production Request is active or inactive in the current activated Production Plan. If the Production Request is active, the last reported job status for the associated Data Processing Requests would be provided.

3) Planning Workload and Processing Turn-Around Reports - These reports will provide tracking information on planned vs. actual processing results. The information provided will include job statistics for a Data Processing Request to allow comparisons in planned vs. actual resource consumption, planned start and end time vs. actual start and end time, planned resource, i.e machine, allocation vs. actual resource, i.e. machine, allocation, etc.

4) Planning Management Reports - These reports will provide the operations staff information on Planning application software events which have occurred. This information will be available from the MSS database.

Other ad-hoc reports can be defined to assist the Production Planning Operations staff in performing their activities. The PDPS Database is the repository used to maintain information on Production Requests and associated Data Processing Requests, Data Subscriptions, PGE Profiles, etc. These reports can be used to track the modifications and provide historical information on these data objects. Because of the use of a consistent RDBMS throughout ECS, the sharing of information between different databases is simplified and will allow for consistent definitions for any number of reports.

# 5. Planning Subsystem Hardware CI

This section describes the functionality and architecture of the Planning Subsystem (PLS) Hardware Configuration Item (PLNHW CI). Because of the close coupling of scheduling/queuing functions with planning (i.e., use of the PDPS database), this section also addresses the scheduling/queuing hardware within the Data Processing Subsystem (DPS) Science Processing Hardware Configuration Item (SPRHW CI).

The PLNHW CI consists of one or more Production Planning/Management workstation(s) or server(s) which run the Planning Workbench software and the PDPS database. A Production Planning/Management workstation supports the planning operations staff in performing their routine production planning and management functions. A workstation is provided for operations personnel to access production planning GUIs via the Planning Workbench application. These functions include candidate plan creation, plan activation, entry of production request information and report generation.

The Planning Subsystem is responsible for maintaining the Planning and Data Processing (PDPS) database. This database serves both planning and scheduling. Planning uses the database in support of the Planning Workbench application used for production planning and resource planning activities. Scheduling uses the database in support of the AutoSys application which runs the production schedule.

The scheduling/queuing hardware in the SPRHW CI consists of one or more Queuing Server(s) which run AutoSys and the AutoSys database. Production Monitor Stations run the AutoSys AutoExpert display GUIs.

The scheduling/queuing portion of the DPS is responsible for maintaining the AutoSys database. An active plan is loaded into the AutoSys database from the Planning Subsystem. AutoSys then proceeds to execute this production schedule (the result of activating this plan). AutoSys uses both the PDPS database and its own database during its execution of the production schedule.

Both the PDPS and AutoSys databases are implemented using Sybase. The PDPS database is used for the persistent storage of critical data while the AutoSys database contains the current data for executing a daily production schedule.

## 5.1 Hardware Design Drivers

Drivers for sizing the hardware to support these functions include:

- database transaction rates,
- job throughput,
- volume of production status to display,
- the need for a fail-soft science processing architecture,
- replication of critical data.

The primary drivers for sizing the PDPS database are data criticality, support of science processing's fail-soft environment, and the volume of database transactions. The PDPS data is critical because it consists of invaluable PGE information and run-time statistics used for the creation of future plans. The PDPS database supports AutoSys in execution of the production schedule. If the database fails, the launching of production jobs by AutoSys will stop. The PDPS database also needs to support multiple database accesses from both planning and processing.

The primary drivers for sizing the AutoSys database are support of science processing's fail-soft environment, job throughput, and the display of production status. AutoSys launches jobs which stage data, run PGEs, report PGE status, and destage data. If AutoSys fails, no new production jobs will be executed and production will eventually stop. AutoSys must also be able to execute enough jobs (throughput) to process the daily production schedule within the allocated time frame (8 or 24 hours which varies by DAAC). AutoSys' production status also needs to be displayed without saturating the person monitoring production.

### 5.1.1 Key Trade-Off Studies and Prototypes

Most of the analysis with respect to scheduling/queuing hardware has focused on COTS suitability. Since these aspects are of primary importance to the planning and scheduling design on the whole, the studies and prototypes have not focused on hardware issues alone, but examine the planning and scheduling design options and requirements as an integrated system.

***Scheduling/Queuing Management Cots***-AutoSys COTS software comes bundled with Sybase, and supports direct query/command access to the database. A limited C language API structure implementation exists. "Job Boxes" allow grouping of smaller, related jobs. Both data-availability ("file watcher") and inter-job dependencies are handled easily. Reporting capabilities are limited (e.g., only start and stop time reported). The system has fault tolerance capacity with the ability to run backup or "shadow" processors. (Primary and secondary hardware capacity is discussed in Section 5.3 on fail-over and recovery.)

A simulation of a CERES chain of PGEs was run across three machines, representative of the class of systems expected to satisfy planning and scheduling execution requirements:

- Sun SPARC 10,
- HP 735, and
- SGI Indigo.

The following aspects were tested: inter-job dependencies, file-watchers, cross-platform execution, and the usefulness of the console GUI and the alert manager. All features performed as expected. A test of a series of two second sleep jobs was run on one platform to determine where overhead might be present in the system. Although some latency was found to occur in the early stages of job execution, the cost in time was found to be negligible when compared to the total estimated times for a real ECS job.

***Database*** **-** A study of relational and object-relational DBMS' was performed. As a result of this DBMS study, four DBMS' were selected for further evaluations: Illustra 2.3.1, Objectstore 3.1, Sybase SQL Server 10, and UniSQL. Hands-on experience was gained while testing and evaluating the products for functionality and maturity. Tests were conducted on a Sun/Sparc 20 running Solaris 2.3. Three sizes of input data, with a maximum size of 3 million granules (instances), were used to test the products' performance scalability. Subsequent evaluations have resulted in selection of the Sybase DBMS as the common COTS for the ECS (SDPS & CSMS).

### 5.1.2 Sizing and Performance Analysis

Database transaction rates were modeled for the PDPS database by estimating the number of updates, inserts, and reads made to the database when performing certain functions. These functions include:

- Creating Data Processing Requests to be added to AutoSys database (DPRs),
- Release of DPR jobs for execution,
- Updating Data Processing Request Job
- Getting DPR Job Status
- Data Initialization
- Local Data Management
- Data Staging
- Data Destaging
- Initiate PGE execution
- Monitor Execution
- Execute Post Processing
- Plan Creation/Activation
- Subscription Notification
- On-Demand Production Request.

Also accounted for in the PDPS database sizing are the added transactions due to database replication.

AutoSys job throughput was estimated based upon AutoSys' customer experience. The job throughput was then extrapolated for several different class machines assuming the use of AutoSys' high availability option which replicates its database. These machines were then matched to DAAC sites based upon production scheduling needs (PGE activations) and hours of operation. Each DAAC site has a different number of PGE activations. The five DAAC sites can be broken into two groups; sites with a lower number of PGE activations per day and sites with a higher number of PGE activations per day.

Workstations and X-terminals used to display production status were sized based on the number of PGE activations and the number of instruments associated with the PGE activations. RAM requirements were calculated based upon the number of jobs being displayed by the AutoSys AutoCons and AutoExpert GUIs.

The DAAC Specific Volumes contain full details on site configuration as well as sizing and rationale. The sizing and performance analyses apply to LaRC, GSFC, JPL, EDC, and NSIDC for Release B. (ASF and ORNL do not have ECS production operations, therefore, no hardware support for the Planning Subsystem is required at these sites.)

### 5.1.3 Scalability, Evolvability, Migration to Release B

With respect to the Planning Subsystem and the Queuing Server(s), general growth and evolvability strategies are not complex. The basic strategy, generally applied to client server configurations, is application of technology refresh through "swap-out" of selected components. Due to Planning's importance to the DAACs, this would have to be performed in parallel to DAAC operations, with initial application of new hardware technology (servers, workstations, disk, etc.) applied within the AI&T area (with later transition to operations support). With respect to salability and evolvability issues outside of general technology refresh, the following growth paths are applicable to the Planning and Queuing Servers and supported by the current design:

- *SMP Servers*-The number of CPUs populating the server(s) depends upon DAAC specific performance requirements. The benefits to this approach would be "rightsizing" of the server platform at a site to meet initial requirements, with an in place upgrade capability to meet out-year performance requirement growth due to reprocessing demands.
- *Disk Requirement Growth*-The current Planning Subsystem design allows for both standard and RAID disk application without rework of the core design. This can be accomplished through the application of channel adapted disk capacity upgrades (e.g., larger RAID units), or through network adapted disk servers addressable by the DBMS servers.

## 5.2 HWCI Structure

A generic block diagram of the Planning hardware is shown in Figure 5-1.

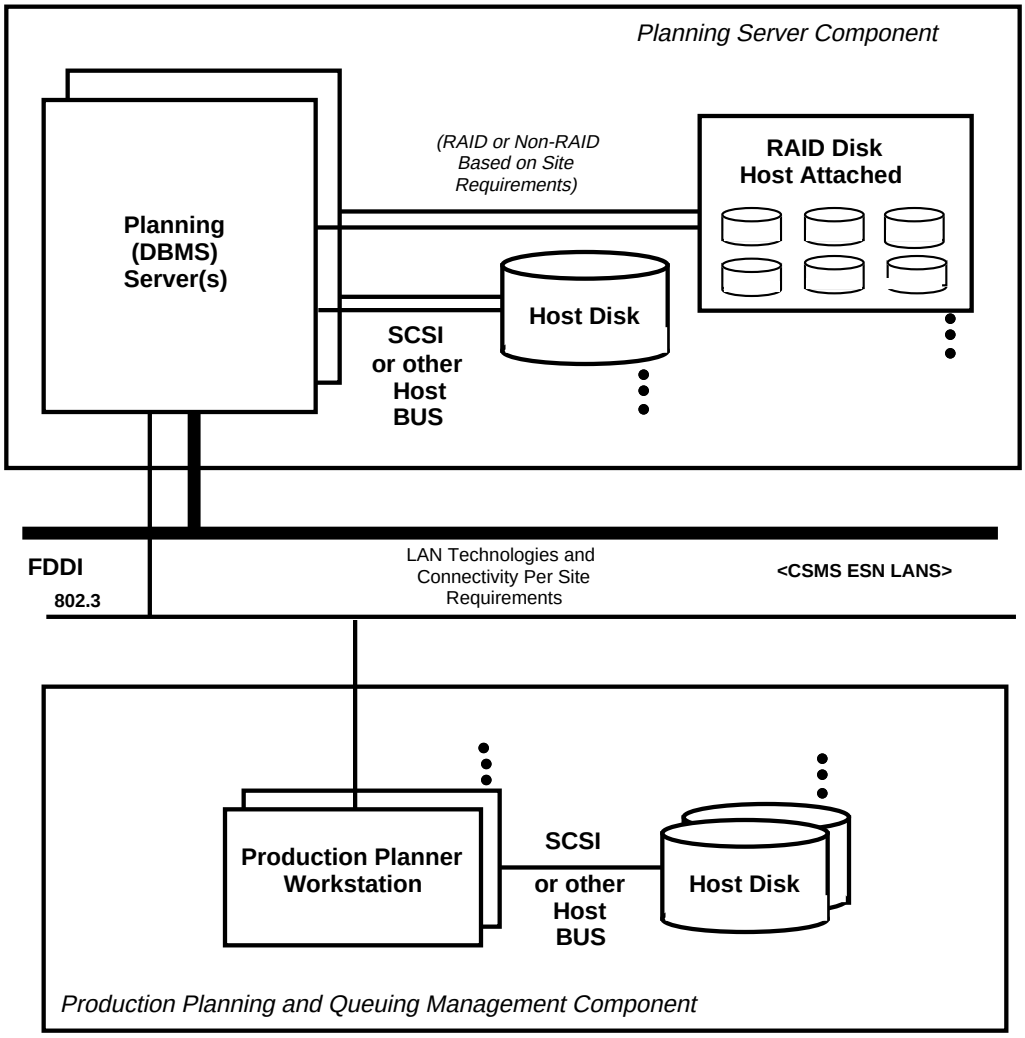



***Figure 5-1. Planning Block Diagram***

The Planning Subsystem components maintain status and database information on: production plans, ground resources, resource status, resource load, etc. The Planning Subsystem coordinates the Data Processing Subsystem activities through data processing requests. The Release B Planning Subsystem hardware configuration consists of SMP servers and workstations.

### 5.2.1 Connectivity

The intra-DAAC data interfaces will be implemented as follows: data interfaces will be of the channel type (e.g., SCSI II); Control interfaces will be of the network type (e.g., FDDI, Ethernet).

The Planning Subsystem network connectivity is illustrated in Figure 5-2. The Planning Subsystem will connect directly to the FDDI production network. This figure highlights the primary components of the Planning HWCI. The server will contain a dual attached station (DAS) card and the workstation will contain single attached station (SAS) card. All hosts will be connected to FDDI concentrators, which will in turn be connected to the FDDI switch via a physically wired FDDI ring. Refer to Volume 0 for a general description of DAAC networks, and to the DAAC Specific volumes for LaRC, GSFC, JPL, EDC, and NSIDC for details on specific CSMS topologies.

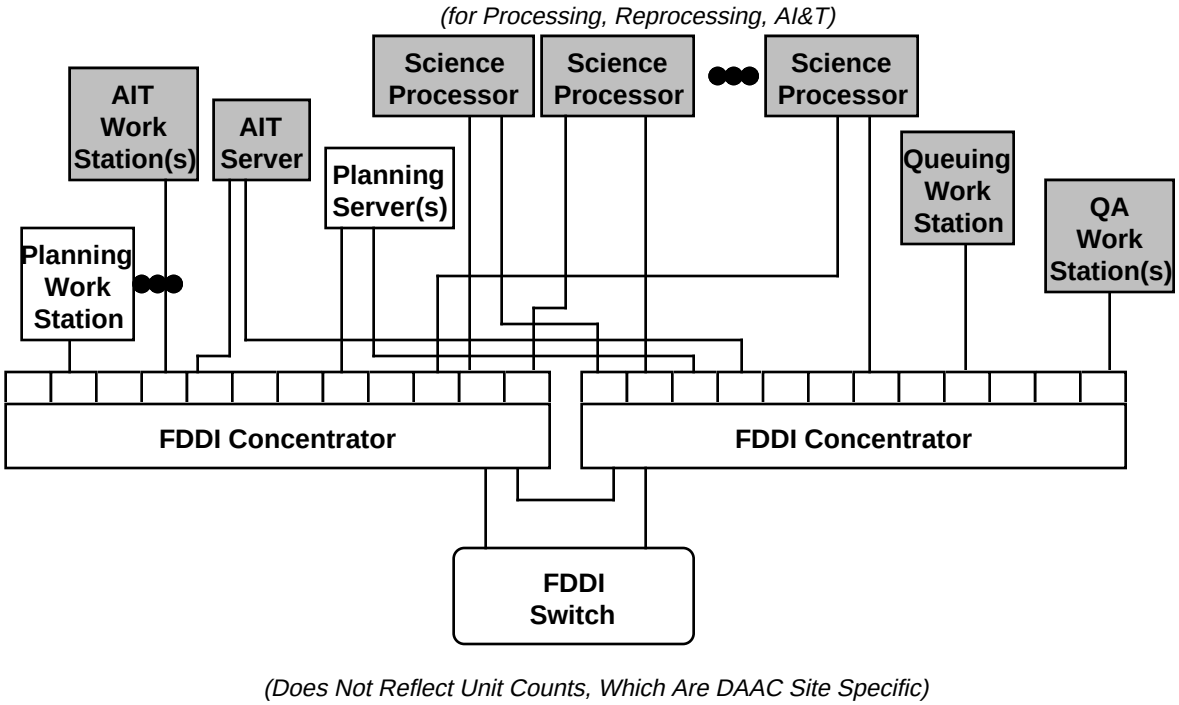



***Figure 5-2. Planning Generic Network Connectivity***

### 5.2.2 HWCI Components

The planning and scheduling functions are implemented on the following hardware:

- PDPS DBMS Server
- Queuing Server
- Production Planning/Management workstation
- Production Monitor Station

Each of these components is discussed in the paragraphs that follow and are elaborated on in Table 5-1.

***PDPS DBMS Server Hardware Component-***The PDPS DBMS Server consists of a server class machine maintaining the PDPS database. Sets of PDPS DBMS servers (one primary, the other backup) are provided for each DAAC site whose complement requires science data processing. The relative size of these servers is DAAC specific and time frame specific. For purposes of discussing this design generically, two servers are assumed. One provided by the PLNHW CI, acting as the primary DBMS server, and the second provided by the SPRHW HWCI acting as the production queue and/or secondary DBMS server as necessary. Note that this redundant configuration is consistent with a DAAC site with larger numbers of PGE activations per day.

***PDPS DBMS Server Hardware Component-***The Queuing Server consists of a server class machine maintaining the AutoSys database. Sets of Queuing servers (one primary, the other backup) are provided for each DAAC site whose complement requires science data processing. The size of these servers is DAAC specific and time frame specific. For purposes of discussing this design generically, two servers are assumed. One provided by the scheduling/queuing hardware in the SPRHW CI, running the AutoSys Event Processor, and the second provided by the PLNHW CI running the AutoSys Shadow Event Processor. Note that this redundant configuration is consistent with a DAAC site with larger numbers of PGE activations per day.

*Table 5-1. PLNHW Logical Components and Equipment Classes*

| Component Name | Class/Type | Comments |
|---|---|---|
| PDPS DBMS Server<br><br>Queuing Server | SMP Server or SMP Workstation | • For lower volume job activation sites:<br>SMP workstations with 2 CPUs.<br>One pair of workstations provided to Data Processing Subsystem (Queuing Server) to assure fail-soft operation.<br>One pair of workstations provide to Planning Subsystem (PDPS DBMS Server) to ensure data integrity.<br>• For higher volume job activation sites:<br>SMP Server with 2 to 12 CPUs. Varies by site depending on production volume (job activations).<br>One server each are provided to Data Processing Subsystem (Queuing Server) and the Planning Subsystem (PDPS DBMS Server) to assure fail soft operations and data integrity.<br>• Interfaces with other subsystems via the ESN LAN. |
| | Disk Storage Array (Host Attached) | • Disk storage of candidate and active plans, as well as PDPS DBMS tables and working space and AutoSys database.<br>For lower volume job activation sites:<br>Queuing Server SMP workstation pair share disk array. PDPS DBMS SMP workstation pair share disk array.<br>For higher volume job activation sites:<br>Queuing Server and PDPS DBMS Server share disk array. |
| Production Planning/ Management<br><br>Production Monitor Station | Workstation and/or X-terminal | • Production Planning/Management Workstation and/or X-terminal support display of Planning Workbench GUIs.<br>• Interfaces with the PDPS DBMS Server component via ESN LAN interfaces or runs concurrently with local PDPS DBMS server at lower volume job activation sites.<br>• Production Monitor Workstation and/or X-terminal support display of AutoSys AutoExpert GUIs.<br>Interfaces with the Queuing Server component via ESN LAN interfaces or runs concurrently with local Queuing Server at lower volume job activation sites. |

***Production Planning/Management Workstation Hardware Component-***This component contains the hardware necessary to support DAAC operations users staff performing routine production planning and management operations. This component consists of a workstation or X-terminal.

## 5.3 Fail-over and Recovery Strategy

The Product Generation function has the following RMA requirements:

(1)Availability: 0.96

(2)Mean Down Time: < 4 hours.

This function is supported by the Production Planning/Management and PDPS workstation(s) and server(s) along with other subsystem hardware including Data Processing's, Queuing server(s) and X-terminal displays, and the science processors as well. The RMA requirement is readily met (reference: Availability Models / Predictions; 515-CD-002-002). However, the PDPS DBMS Server's data and functionality is critical and requires additional backup capability.

***PDPS DBMS Server Architecture***-The basic approach to Planning Subsystem fail-over and recovery strategy for Release B is to provide Sybase replication for the PDPS DBMS server. Sybase replication provides a form of redundancy to protect against hardware failure and to provide a degree of fault tolerance. Replication is the capability to maintain a replicate of all data stored on a database device. Replication provides non-stop recovery in the event of disk failure. A Sybase procedure is executed to cause an SQL Server database device to be duplicated, i.e., all writes to the device are copied to a separate physical device. If one of the devices fail, the other contains an up-to-date copy of all transactions. RAID mirroring of databases could be used to replicate the database in order to maximize disk resources and performance but this requires manual intervention to restart the Sybase SQL server upon a failure of the primary Sybase SQL server. Replicating all devices such as the master device, user databases, and transaction logs provides a non-stop recovery from hardware failure, but requires more disk space and provides a reduction in overall performance. In the event of media failure, the replicated database and SQL backup server can take over, typically without any downtime or, in this application, interruption of processing. When the damaged device is repaired or replaced, it is resynchronized with the undamaged copy. Replication ensures the safety of PDPS database. The Sybase replication of the PDPS DBMS allows the secondary server to continue with the PDPS DBMS responsibilities should the primary PDPS DBMS fail, thereby supporting a fail-soft science processing environment.

The ECS system also provides the capability of tape backup for all critical information (software configuration files, database configuration information, etc.) and will be performed at regular intervals.

***Queuing Server Architecture***-The fail-over and recovery strategy for the scheduling/queuing functions is handled by AutoSys. AutoSys runs in a Dual Data Server Mode which replicates AutoSys' database on separate disk. When a disk or database failure occurs in the primary database, AutoSys continues by using the backup database. Also, the primary Queuing Server runs the AutoSys Event Processor while the secondary Queuing Server runs the AutoSys Shadow Event Processor. The Shadow Event Processor remains dormant and periodically checks the Event Processor's status. When the Shadow Event Processor detects that the Event Processor has malfunctioned, the Shadow Event Processor takes charge of the execution of the production schedule. This allows the continued launching of production jobs and supports the science processing's fail-soft requirement.

Figure 5-3 functionally shows the primary and secondary server concept. The gray areas indicate the primary server and the primary disk.

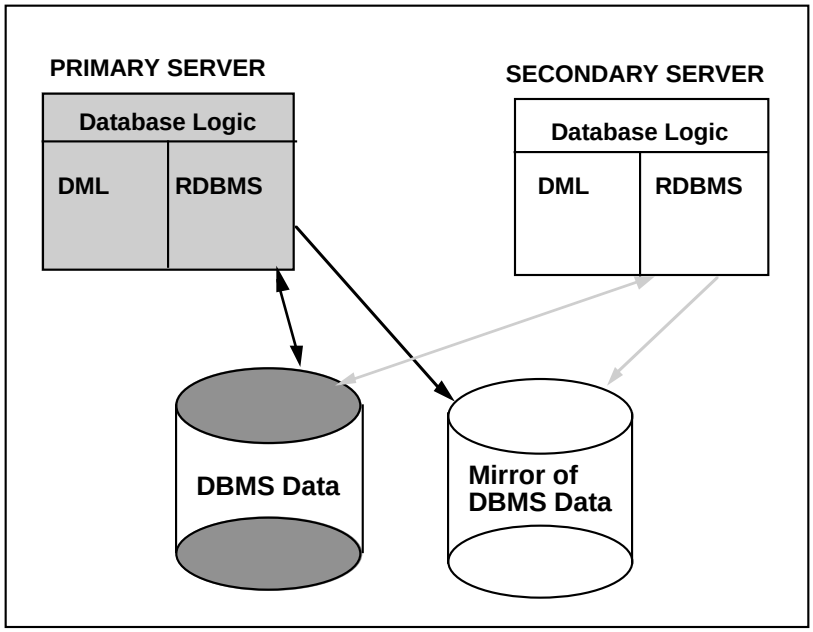



***Figure 5-3. Primary and Secondary Server Concept Overview***

# Appendix A. Requirements Trace

The Interim Release 1 (Ir1), TRMM Development (Release A) and AM-1 Development (Release B) Level 4 requirements listed in Table A-1 reflect the RTM requirements database RELB_CDR_030196. The Object Classes listed are from the CDR version of OMT and are described throughout this document., as are the CSCs , CIs and subsystem listed.

***Table A-1. Requirements Trace (1 of 14)***

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-00005 | The PLANG CI shall accept priority Production Requests for the generation of specific Data Products. | PlDataTypeCatalogue<br>PlProductionRequestB<br>PlProductionRequestUI |
| S-PLS-00010 | The PLANG CI shall accept Production Requests for specific Data Products with associated time windows that are to be routinely generated. | PlProductionRequestB<br>PlDataTypeCatalogue<br>PlProductionRequestUI |
| S-PLS-00020 | The PLANG CI shall generate Data Processing Requests from Production Requests. | PlDataGranule<br>PlDataTypeReq<br>PlPGE<br>PlDataTypeB<br>PlDataTypeCatalogue<br>PlProductionRequestB<br>PlProductionRequestUI<br>PlDPRB |
| S-PLS-00040 | The PLANG CI shall reject a Production Request if an invalid product identifier has been specified. | PlDataTypeCatalogue<br>PlProductionRequestB<br>PlProductionRequestUI |
| S-PLS-00050 | The PLANG CI shall reject a Production Request if an unauthorized User Identifier is specified. | PlUser<br>PlProductionRequestUI<br>PlProductionRequestB<br>PlDataTypeCatalogue |
| S-PLS-00060 | The PLANG CI shall support the capability to display a response message to the operations staff, indicating the acceptance / rejection status of Production Requests and the reasons for rejection (if applicable). | PlProductionRequestB<br>PlProductionRequestUI |
| S-PLS-00070 | The PLANG CI shall accept Production Requests for reprocessing of Data Products from currently available input data. | PlReprocessingStrategyNB<br>PlReprocPRNB |
| S-PLS-00100 | The PLANG CI shall accept Production Requests for On-Demand Data Products. | PlOnDemandManagerNB<br>PlOnDemandPRNB |
| S-PLS-00110 | The PLANG CI shall reject a Production Request for On-Demand Data Products if the processing completion deadline (specified in the Production Request) cannot be met. | PlOnDemandPRNB |

*Table A-1. Requirements Trace (2 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-00120 | The PLANG CI shall validate Production Requests for On-Demand Data Products against a pre-approved list of acceptance criteria. | PlOnDemandManagerNB<br>PlOnDemandPRNB |
| S-PLS-00130 | The PLANG CI shall send a response message to the Data Server confirming the acceptance status of the received Production Request for On-Demand Data Products (" accepted", "rejected", "deferred") and reason for rejection of a request (if applicable). | PlOnDemandPRNB |
| S-PLS-00140 | Upon acceptance of a Production Request for an On-Demand Data Product, the PLANG CI shall immediately forward its corresponding Data Processing Requests to the PRONG CI if predefined resource thresholds are not exceeded and if the input data is available. | PlOnDemandManagerNB<br>PlOnDemandPRNB |
| S-PLS-00150 | The PLANG CI shall defer On-Demand Production Requests for future plan generation consideration when these On-Demand Production Requests are estimated to exceed a predefined resource threshold. | PlOnDemandPRNB<br>PlOnDemandManagerNB |
| S-PLS-00160 | If a Production Request for an On-Demand Data Product exceeds a predefined resource usage threshold, the PLANG CI shall notify the operations staff that the Production Request has been deferred. | PlOnDemandPRNB<br>PlOnDemandManagerNB |
| S-PLS-00165 | The PLANG CI shall allow the operator to specify the resource usage thresholds used to accept or defer On-Demand Production Requests. | PlRescUseThreshNB<br>PlOnDemandPRNB<br>PlOnDemandManagerNB |
| S-PLS-00170 | The PLANG CI shall accept updates (modifications/ cancellations) to Production Requests for On-Demand Data Products. | PlOnDemandPRNB |
| S-PLS-00180 | The PLANG CI shall validate updates (modifications / cancellations) to existing Production Requests. | PlPGE<br>PlDPRB<br>PlDataGranule<br>PlDataTypeB<br>PlDataTypeReq<br>PlProductionRequestB<br>PlProductionRequestUI<br>PlDataTypeCatalogue |
| S-PLS-00190 | The PLANG CI shall forward a response message to the Data Server indicating acceptance / rejection status of the updates to the Production Request for On-Demand Data Products . | PlOnDemandPRNB |

*Table A-1. Requirements Trace (3 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-00200 | The PLANG CI shall accept updates (modifications / cancellations) to Production Requests entered by the operations staff. | PlDataTypeB<br>PlDPRB<br>PlDataGranule<br>PlDataTypeCatalogue<br>PlPGE<br>PlProductionRequestB<br>PlProductionRequestUI<br>PlDataTypeReq |
| S-PLS-00220 | The PLANG CI shall support the display of a response message to the operations staff, indicating the acceptance /rejection status of updates to a Production Request. | PlProductionRequestUI |
| S-PLS-00230 | The PLANG CI shall provide production rules (via GUI) to break up large reprocessing Production Requests. | PlReprocPRNB<br>PlReprocessingStrategyNB |
| S-PLS-00260 | For each Production Request being processed, the PLANG CI shall interact with the appropriate instance of the SDSRV CI to determine whether the Granules needed to satisfy the request exist. | PlProductionRequestUI<br>DsClESDTReference<br>DsClESDTReferenceCollect or<br>PlDataTypeB<br>PlProductionRequestB<br>DsClQuery |
| S-PLS-00300 | The PLANG CI shall accept ground events to describe the allocation of data processing resources to non-production tasks. | PlRpResourceReservation<br>PlGroundEvent |
| S-PLS-00310 | The PLANG CI specification of ground events shall include priorities, dependencies, and estimated duration. | PlRpResourceReservation<br>PlGroundEvent |
| S-PLS-00400 | The PLANG CI shall maintain Product Generation Executives (PGEs) information that identifies the Science Software, the order of execution, the conditions for execution, the processing environment, and the input / output data types and locations. | PlUserParameters<br>PlPGE<br>PlDataTypeReq<br>PlDataTypeB<br>PlResourceRequirement<br>PlPGEProfile |
| S-PLS-00405 | The PLANG CI shall allow the conditions for execution of Product Generation Executives (PGEs) to include intermediate results such as metadata fields of input data. | PlMetadataChecks |
| S-PLS-00407 | The PLANG CI shall maintain Product Generation Executives (PGEs) information necessary to support the production of tile or spatial-based output Granules. | PlTileScheduledNB<br>PlOrbitScheduledNB |
| S-PLS-00410 | The PLANG CI shall support the capability to display (via GUI) a list of PGEs maintained in its PGE information database. | PlPGEProfile |
| S-PLS-00420 | The PLANG CI shall support the capability to browse (via GUI) the information maintained on a specific PGE. | PlPGEProfile |

## *Table A-1. Requirements Trace (4 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-00430 | The PLANG CI shall support the capability to (a) allow (authorized) operations staff updates (enter / modify / delete) of PGE information in the Planning PGE information database, (b) maintain a record of updates made. | PlPGEProfile |
| S-PLS-00440 | The PLANG CI shall maintain Production Rules that define the production strategy (rules defining production priorities and preferences) to be used when preparing a Production Plan. | PlProductionRequestB<br>PlDPRB |
| S-PLS-00445 | The PLANG CI shall maintain multiple Production Strategies defined by sets of Production Rules to be used when preparing a Production Plan. | PlPGEPriorityNB<br>PlUserPriorityNB<br>PlProdStratUINB<br>PlProdStratNB |
| S-PLS-00450 | The PLANG CI shall support the capability that allows the operations staff to update (enter/ modify/ delete) the Production Rules (via GUI). | PlProductionRequestB<br>PlProductionRequestUI |
| S-PLS-00455 | The PLANG CI shall support the capability that allows the operations staff to update (enter/ modify/ delete) the Production Strategies (via GUI). | PlPGEPriorityNB<br>PlProdStratNB<br>PlUserPriorityNB<br>PlProdStratUINB |
| S-PLS-00457 | The PLANG CI GUI shall conform to the guidelines in version 5.1 of the ECS User Interface Style Guide. | PLANG |
| S-PLS-00458 | To the extent possible, the PLANG CI COTS GUI shall be configured to conform to the guidelines in version 5.1 of the ECS User Interface Style Guide. | PLANG |
| S-PLS-00460 | The PLANG CI shall maintain lists of Granules needed to satisfy Production Requests. | PlProductionRequestB<br>PlDataTypeB<br>PlDataTypeReq<br>PlDPRB<br>PlDataGranule<br>PlPGE |
| S-PLS-00465 | The PLANG shall maintain lists of input Granules in order to support the production of tile or spatial-based output Granules | PlTileScheduledNB<br>PlOrbitScheduledNB |
| S-PLS-00470 | The PLANG CI shall maintain information on the following: a. current processing status of all Production Requests received, b. current processing status of all Data Processing Requests generated, c. detected processing fault data. | PlDataGranule<br>PlDataTypeB<br>PlDataTypeCatalogue<br>PlProductionRequestB<br>PlPGE<br>PlDataTypeReq<br>PlDPRB |

*Table A-1. Requirements Trace (5 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-00475 | The PLANG CI shall maintain information on all Candidate and Active Plans generated. | PlGroundActivity<br>PlDPRB<br>PlDPRs<br>PlGroundEvents<br>PlPGEActivity<br>PlPlanB<br>PlPlanningWorkbenchUI<br>PlResourceManager<br>PlActivePlan<br>PlGroundEvent |
| S-PLS-00490 | The PLANG CI shall maintain Planning system fault data using fault isolation tools provided by the LSM. | EcEvent |
| S-PLS-00604 | The PLANG CI shall receive advertisements from the IOS. | PlDataTypeB |
| S-PLS-00606 | The PLANG CI shall send advertisement subscriptions to the IOS. | To be deleted, see table A-2. |
| S-PLS-00611 | The operations staff shall manually submit (to the Data Server) Data Subscriptions for the Data Availability Schedules (DAS) of any remote ECS sites, any IP and any ODC that makes a DAS available | PlDASNB |
| S-PLS-00615 | The operations staff shall manually submit (to the Data Server) Data Subscriptions for FOS plans and schedules. | PlFOSDASFile |
| S-PLS-00631 | The PLANG CI shall receive Data Availability Schedule Notices indicating arrival of Data Availability Schedules (DAS) for any remote ECS site, any IP, and any ODC that makes a Data Availability Schedules available. | PlDASNB |
| S-PLS-00635 | The PLANG CI shall receive Data Availability Schedule Notices indicating arrival of FOS plans and schedules | PlFOSDASFile |
| S-PLS-00651 | The PLANG CI shall accept Data Availability Schedules (DAS), for remote ECS sites, IPs, and ODCs, based on the Data Availability Schedule Notices received. | PlDASNB |
| S-PLS-00652 | The PLANG CI shall support the capability to retrieve FOS plans and schedules from the Data Server. | PlFOSDASFile |
| S-PLS-00654 | The PLANG CI shall create a Data Availability Schedule (DAS) for EDOS based on FOS plans and schedules. | PlDataTranferHistory |
| S-PLS-00656 | The PLANG CI shall send a response message to Data Server upon receiving FOS plan and schedule, confirming the receiving of the data | PlFOSDASFile |
| S-PLS-00665 | The PLANG CI shall notify the operations staff (via GUI), about the arrival of any Data Availability Schedule Notice corresponding to a DAS. | PlDASNB |
| S-PLS-00670 | The PLANG CI shall provide (to the operations staff) the capability to enter, via GUI, "plan creation requests" that initiate creation of Candidate Plans. | PlPlanningWorkbenchUI<br>PlPlanB |

*Table A-1. Requirements Trace (6 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-00680 | The PLANG CI shall provide the capability to generate multiple Candidate Plans. | PlPlanB<br>PlPlanningWorkbenchUI |
| S-PLS-00690 | The PLANG CI shall create a Candidate Plan specifying a timeline for PGE execution that will satisfy Production Requests for Standard Products. | PlGroundActivity<br>PlPlanB<br>PlPGEActivity<br>PlPlanningWorkbenchUI<br>PlProductionRequestB<br>PlDPRs<br>PlDPRB<br>PlGroundEvents<br>PlGroundEvent |
| S-PLS-00700 | The PLANG CI shall create a Candidate Plan specifying a timeline for PGE execution that will satisfy Production Requests for Reprocessing and On-Demand Data Products. | PlPRCollectionNB |
| S-PLS-00710 | The PLANG CI shall create a Candidate Plan based on the following:<br>1. Outstanding production requests, their priorities and estimated runtimes,<br>2. Ground events, their priority and estimated duration,<br>3. Planning production rules,<br>4. Mutual PGE accessibility of shared data,<br>5. Completion notification status messages from Data Processing. | PlGroundEvents<br>PlResourceManager<br>PlProductionRequestB<br>PlPlanningWorkbenchUI<br>PlPGEActivity<br>PlGroundEvent<br>PlGroundActivity<br>PlDPRs<br>PlDPRB<br>PlPlanB |
| S-PLS-00720 | The PLANG CI shall create a Candidate Plan based on the data availability schedules for remote ECS sites, EDOS, the IPs, and ODCs, as needed. | PlDASNB |
| S-PLS-00730 | The PLANG CI shall have the capability to plan algorithm and calibration coefficient test time in the test environment. | PlGroundEvent<br>PlPlanningWorkbenchUI<br>PlPlanB |
| S-PLS-00740 | The PLANG CI shall have the capability to schedule algorithm test Data Processing Requests that do not interfere with the operational production environment. | PlResourceRequirement<br>PlResourceManager<br>PlPlanningWorkbenchUI<br>PlPlanB<br>PlPGEActivity<br>PlDPRs<br>PlDPRB |
| S-PLS-00741 | The PLANG CI shall seperate AI&T activities from the operational production environment. | PlPGEActivity<br>PlDPRB<br>PlPlanB<br>PlPlanningWorkbenchUI<br>PlResourceRequirement<br>PlDPRs |

*Table A-1. Requirements Trace (7 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-00760 | The PLANG CI shall send electronic copies of the Candidate Plans and corresponding metadata to the designated local Data Server for storage and distribution. | PlPlanningWorkbenchUI<br>PlPlanMetadataFile<br>PlPlanBinaryReportFile<br>PlPlanB<br>PlPlanASCIIReportFile<br>PlPublishedPlan |
| S-PLS-00770 | The PLANG CI shall provide (to the operations staff) the capability to enter, via GUI, a "Plan cancellation" request, indicating cancellation of the currently Active Plan. | DpPrScheduler<br>PlPlanningWorkbenchUI<br>PlPGEActivity<br>PlActivePlan<br>PlDPRB |
| S-PLS-00780 | The PLANG CI shall generate Data Processing Request cancellations against previously submitted Data Processing Requests (if so directed by the operations staff), or upon activation of a new plan that no longer requires those requests. | DpPrScheduler<br>PlPlanningWorkbenchUI<br>PlPlanB<br>PlDPRB<br>PlPGEActivity |
| S-PLS-00790 | The PLANG CI shall send a Data Processing Request cancellation to the instance of the PRONG CI that received the original Data Processing Request. | PlPlanningWorkbenchUI<br>PlPlanB<br>PlDPRs<br>PlDPRB |
| S-PLS-00800 | The PLANG CI shall provide to the operations staff the capability to enter, via GUI, a "plan activation request" that identifies which Candidate Plan is to be activated. | PlDPRs<br>PlPlanningWorkbenchUI<br>PlPGEActivity<br>PlResourceManager<br>PlPlanB<br>PlDPRB |
| S-PLS-00811 | The PLANG CI shall reconcile any outstanding Data Processing Requests in the current Active Plan with the Data Processing Requests in the Candidate Plan to be activated. | PlActivePlan<br>PlDPRB |
| S-PLS-00825 | The PLANG CI shall have the capability to identify all available input data (as specified in the Active Plan) that is currently awaiting quality assurance information. | PlSubscriptionManager |
| S-PLS-00827 | The PLANG CI shall update the quality assurance status of input data (if applicable) to reflect an expired QA timeout period if its quality assurance information has not been received within specified time periods._ | PlSubscriptionManager |
| S-PLS-00830 | The PLANG CI shall send Data Processing Requests (specified in an Active Plan) to a processing resource that can perform the processing, if the following applies:<br>a. All required input data (including metadata) is available<br>b. Its input data has passed quality assurance (if applicable) | PlDataGranule<br>PlDPRB<br>DpPrScheduler<br>PlSubscriptionManager |

*Table A-1. Requirements Trace (8 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-00840 | The PLANG CI shall send electronic copies of the Active Plan and corresponding metadata to the designated local Data Server for storage and distribution. | PlPlanB<br>PlPlanASCIIReportFile<br>PlPlanBinaryReportFile<br>PlPlanMetadataFile<br>PlPlanningWorkbenchUI<br>PlPublishedPlan |
| S-PLS-00845 | The PLANG CI shall support the capability to retrieve stored plans and their corresponding metadata from the Data Server based on specific queries. | PlPlanGenerationUIB |
| S-PLS-00850 | The PLANG CI shall have the capability to generate data availability schedules (and the corresponding metadata) that reflect the Data Products expected to be generated in the Production Plan. | PlDASNB |
| S-PLS-00860 | The PLANG CI shall send the data availability schedules and the corresponding metadata to the designated Data Server. | PlDASNB |
| S-PLS-00870 | The operations staff shall manually submit Data Subscriptions for PGE input data to the appropriate Data Servers. | PlSubscriptionSubmitIF<br>PlDataTypes<br>PlDataTypeB<br>IoAdAdvertisingSrv_C<br>IoAdServiceAdvertisement_C<br>IoAdServiceCollection_C |
| S-PLS-00872 | The operations staff shall manually submit Data Subscriptions for L0 data to the Ingest Subsystem. | PlDataTypeB<br>PlDataTypes<br>PlSubscriptionSubmitIF |
| S-PLS-00875 | The PLANG CI shall receive Subscription Notices indicating availability of subscribed data. | PlDataGranule<br>PlDataTypes<br>DsClESDTReference<br>EcMpMsgPsngCtrl<br>EcUrUR |
| S-PLS-00880 | The operations staff shall manually cancel Data Subscriptions for input data to PGEs that are no longer used, once they determine that the input data is not required by any other PGE. | DsClSubscription<br>PlDataTypeB<br>PlDataTypes<br>PlSubscriptionSubmitIF |
| S-PLS-01000 | The PLANG CI shall receive a Data Processing Request Response message, acknowledging acceptance of the Data Processing Request forwarded to the PRONG CI. | PlActivities<br>DpPrScheduler<br>PlDPRB |
| S-PLS-01010 | The PLANG CI shall receive "Complete Notification" status messages, indicating the completion status of Data Processing Requests. | DpPrScheduler<br>PlDPRB<br>PlActivities |
| S-PLS-01020 | The PLANG CI shall receive responses to Data Processing Request cancellations indicating the completion status of the cancellation requests. | DpPrScheduler<br>PlActivities<br>PlDPRB |

*Table A-1. Requirements Trace (9 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-01030 | The PLANG CI shall update the Active Plan with the current processing status of each Data Processing Request listed. | PlPlanB<br>DpPrScheduler<br>PlPGEActivity<br>PlPlanningWorkbenchUI<br>PlDPRB<br>PlActivities |
| S-PLS-01040 | The PLANG CI shall send the current processing status of Production Requests (for On-Demand Data Products) to the originating Data Server. | PlOnDemandManagerNB |
| S-PLS-01200 | The PLANG CI shall provide the operations staff with the capability to perform the following on-line functions, via GUI:<br>a. Entry of product requests for standard products,<br>b. Query / update / cancellation of production requests for standard products,<br>c. Query status of production requests,<br>d. Query / update of production rules and PGE information,<br>e. Entry of plan creation requests,<br>f. Entry of plan activation requests,<br>g. Entry of plan cancellation requests,<br>h. Query candidate / active plans and corresponding status,<br>i. Entry of requests for processing log reports / production and data processing request status reports / resource utilization reports / planning workload status reports / management reports, | PlPlanningWorkbenchUI |
| S-PLS-01210 | The PLANG CI shall provide the operations staff with the capability to perform the following on-line functions, via GUI: a. Entry/query/update/ cancellation of Production Requests for Reprocessing, b. Query/update/cancellation of Production Requests for On-Demand Data Products. | PlProductionRequestB<br>PlProductionRequestUI |
| S-PLS-01220 | The PLANG CI shall have the capability to accept a request from the operations staff for scheduling algorithm and calibration coefficient test time in the test environment. | PlGroundEvent<br>PlPlanB<br>PlPlanningWorkbenchUI |
| S-PLS-01230 | The PLANG CI shall support the display (via GUI) of warning messages to the operations staff indicating revised completion times if processing will not complete per original schedule. | PlPlanningWorkbenchUI |
| S-PLS-01240 | The PLANG CI shall support the display (via GUI) of Planning hardware and software detected faults to the operations staff. | EcEvent |

*Table A-1. Requirements Trace (10 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-01245 | The PLANG CI shall provide capability to make available (for review by all affected instrument teams) information related to product generation delays and production faults. | PlPlanMetadataFile<br>PlPlanningWorkbenchUI<br>PlPlanBinaryReportFile<br>PlPlanASCIIReportFile<br>PlPublishedPlan<br>PlPlanB |
| S-PLS-01250 | The PLANG CI shall record detected hardware and software errors in a Planning processing log. | EcEvent |
| S-PLS-01260 | The PLANG CI shall support the capability to generate Planning processing log reports (periodically and on request) for a specified time period. | DpPrDbInterface |
| S-PLS-01270 | The PLANG CI shall support the generation of Data Processing Request Status reports (upon request) that will provide Data Processing Request information based on the report generation parameters and the time period specified. | PlPlanB<br>PlPlanningWorkbenchUI<br>PlDPRs<br>PlDPRB |
| S-PLS-01280 | The PLANG CI shall support the generation of Production Request Status reports (upon request) that will provide Production Request information based on the report generation parameters and the time period specified. | PlProductionRequestB<br>PlProductionRequestUI<br>PlDataGranule<br>PlDataTypeB<br>PlDataTypeCatalogue<br>PlDataTypeReq<br>PlPGE<br>PlDPRB |
| S-PLS-01290 | The PLANG CI shall support the generation of resource utilization reports (periodically and on request). | MsRgManagerB |
| S-PLS-01300 | The PLANG CI shall support the capability to generate PLANG CI processing workload and processing turnaround time reports (periodically and on request). | MsRgRepWriterB |
| S-PLS-01320 | The PLANG CI shall make all reports generated available for review. | MsRgRepWriterB |
| S-PLS-01330 | The PLANG CI shall restrict the functions available to operators depending on operations role, (e.g., to permit the resource manager role to enter ground events, but to restrict that role from being able to enter production requests). | ECSSecurity |
| S-PLS-01400 | The PLANG CI shall accept the fault isolation tools for the PLANG CI. | PLANG |
| S-PLS-01410 | The PLANG CI shall forward faults detected in the Planning system to MSS. | PLANG |
| S-PLS-01430 | The PLANG CI shall send to MSS product scheduling, processing status and data quality information. | PLANG |
| S-PLS-01440 | The PLANG CI shall collect Fault Management Data and provide it to the MSS. | PLANG |

*Table A-1. Requirements Trace (11 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-01450 | The PLANG CI shall collect Configuration Management Data and provide it to the MSS. | PLANG |
| S-PLS-01460 | The PLANG CI shall collect Accounting Management Data and provide it to the MSS. | PLANG |
| S-PLS-01470 | The PLANG CI shall collect Accountability Management Data and provide it to the MSS. | PLANG |
| S-PLS-01480 | The PLANG CI shall collect Performance Management Data and provide it to the MSS. | PLANG |
| S-PLS-01490 | The PLANG CI shall collect Security Management Data and provide it to the MSS. | PLANG |
| S-PLS-01500 | The PLANG CI shall collect Scheduling Management Data and provide it to the MSS. | PLANG |
| S-PLS-01600 | The PLANG CI design and implementation shall have the flexibility to accommodate Planning expansion up to a factor of 3 in its capacity with no changes to its design, and up to a factor of 10 without major changes to its design. Such expansion in capacity or capability shall be transparent to existing algorithms or product specifications. | PLANG |
| S-PLS-01610 | The PLANG CI shall be developed with configuration controlled APIs that will be capable of supporting development and integration of new algorithms developed at DAAC to support DAAC value-added production. | PLANG |
| S-PLS-02000 | The PLANG CI shall be able to accept scheduling information on external events which affect processing resources and operations | PlRpResourceReservation |
| S-PLS-02010 | The PLANG CI shall identify scheduling conflicts. | PlPlanB |
| S-PLS-02020 | The PLANG CI shall be able to provide operations personnel priorities and planned execution times of jobs causing scheduling conflicts within and between DAACs. | PlTimeLineDisplay |
| S-PLS-02030 | The PLANG CI shall identify conflicts in plans caused by cross-DAAC data dependencies. | PlPlanB |
| S-PLS-02040 | The PLANG CI shall be able to display (via GUI) cross-DAAC data dependencies. | PlPlanB |
| S-PLS-02050 | The PLANG CI shall be able to provide plans to PLANG CIs at other sites. | PlExportedPlanNB |
| S-PLS-02060 | The PLANG CI shall be able to account for cross-DAAC data dependencies in the plans it generates. | PlPlanB |
| S-PLS-02070 | The PLANG CI shall be able to integrate multiple DAAC plans to produce a coordinated plan. | PlTimelineDisplay |

*Table A-1. Requirements Trace (12 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-02080 | The PLANG CI shall provide the operations and management staff at a site the ability to send routine scheduling information to other sites. | PlExportedPlanNB |
| S-PLS-02090 | The PLANG CI shall able to receive routine scheduling information from other sites. | PlExportedPlanNB |
| S-PLS-02100 | The PLANG CI shall provide the operations and management staff at a site the ability to send scheduling request information to other sites. | PLANG |
| S-PLS-02110 | The PLANG CI shall able to receive scheduling request information from other sites. | PLANG |
| S-PLS-02120 | The PLANG CI shall provide the operations and management staff at a site the ability to send schedule conflict alert information to other sites. | PLANG |
| S-PLS-02130 | The PLANG CI shall able to receive schedule conflict alert information from other sites. | PLANG |
| S-PLS-02140 | The PLANG CI shall provide the operations and management staff at a site the ability to send emergency scheduling information to other sites. | PLANG |
| S-PLS-60010 | The PLNHW CI shall support the hardware resource requirements of the PLANG CI and its interface requirements with the operations staff performing planning functions. | PLNHW |
| S-PLS-60200 | The maximum down time of the PLNHW CI shall not exceed twice the required MDT in 99 percent of failure occurrences. | PLNHW |
| S-PLS-60320 | The PLNHW CI shall support transactions per day, as specified for each release and corresponding DAAC sites in Appendix E, Table E-1 of the current version of 304-CD-002 for Release A and Table E-1 of Appendix E of the current version of 304-CD-005 for Release B. | PLNHW |
| S-PLS-60330 | The PLNHW CI shall provide local storage in support of the DAAC-specific requirements as specified in Appendix E, Table E-9 of the current version of 304-CD-005. | PLNHW |
| S-PLS-60380 | The PLNHW CI design and implementation shall have the flexibility to accommodate planning workload expansion up to a factor of 3 in its capacity with no changes in its design and up to a factor of 10 without major changes to its design. | PLNHW |
| S-PLS-60410 | The PLNHW CI shall be capable of operating in a 24 hour per day, 7 days a week mode. | PLNHW |
| S-PLS-60420 | PLNHW CI functions shall have an operational availability of .96 as a minimum and Mean Down Time of < 4 hours during times of staffed operations. | PLNHW |

*Table A-1. Requirements Trace (13 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-60450 | The PLNHW CI elements and components shall include the on-line (operational mode) and off-line (test-mode) fault detection and isolation capabilities required to achieve the specified operational availability requirements. | PLNHW |
| S-PLS-60610 | The PLNHW CI shall interface with the ISS. | PLNHW |
| S-PLS-60630 | The PLNHW CI shall provide maintenance interfaces to support the function of System Maintenance. | PLNHW |
| S-PLS-60640 | The PLNHW CI shall provide operations interfaces to support the function of System Maintenance. | PLNHW |
| S-PLS-61010 | The PLNHW CI shall support test activities throughout the development phase. | PLNHW |
| S-PLS-61020 | The following testing shall be performed on the PLNHW CI:<br>a. Unit testing<br>b. Subsystem testing<br>c. Integration & Testing<br>d. End-to- End testing | PLNHW |
| S-PLS-61040 | Internal testing shall be performed on the PLNHW CI which includes tests of hardware functions, and integration testing with other SDPS subsystems. | PLNHW |
| S-PLS-61050 | Internal testing shall be performed on the PLNHW CI to verify the internal interfaces to the Data Server, and Ingest subsystems. | PLNHW |
| S-PLS-61080 | The PLNHW CI shall be capable of supporting end-to-end test and verification activities of the EOS program including during the pre-launch, spacecraft verification, and instrument verification phases. | PLNHW |
| S-PLS-61150 | The PLNHW CI shall be capable of being monitored during testing. | PLNHW |
| S-PLS-61210 | The operating system for each Unix platform in the PLNHW CI shall conform to the POSIX.2 standard. | PLNHW |
| S-PLS-61220 | Each PLNHW CI POSIX.2 compliant platform shall have the following utilities installed at a minimum: perl, emacs, gzip, tar, imake, prof, gprof, nm. | PLNHW |
| S-PLS-61230 | Each PLNHW CI POSIX.2 compliant platform shall have the following POSIX.2 user Portability Utilities installed at a minimum: man, vi. | PLNHW |
| S-PLS-61240 | Each PLNHW CI platform shall have the following POSIX.2 Software Development utilities installed: make, imake. | PLNHW |

*Table A-1. Requirements Trace (14 of 14)*

| L4 Rqmt ID | L4 Requirement Text | Object Class, CSC, or CI |
|---|---|---|
| S-PLS-61260 | Each PLNHW CI POSIX.2 compliant platform shall have the following Unix shells installed at a minimum: C shell, Bourne shell, Korn shell. | PLNHW |
| S-PLS-61350 | Each PLNHW CI POSIX.2 compliant platform shall have a screen capture utility. | PLNHW |
| S-PLS-61530 | The PLNHW CI shall contain the processing, storage, and interface resources to support the planning functions for the TRMM mission instruments of CERES and LIS. | PLNHW |
| S-PLS-61610 | Each PLNHW CI workstation platform shall provide a hard media device with a capacity of TBD GB for software and system maintenance and upgrade support. | PLNHW |

Table A-2 identifies proposed/pending requirements changes which have not been approved and/or were not yet in the RELB_CDR_030196 version of the requirements database.

### *Table A-2. Intended Changes to CDR Requirements Baseline (1 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| ~~S-PLS-00110~~ | ~~The PLANG CI shall reject a Production Request for On-Demand Data Products if the processing completion deadline (specified in the Production Request) cannot be met.~~ | Not currently receiving completion deadline as part of On-Demand Production Requests. | PlOnDemandPRNB |
| ~~S-PLS-00120~~ | ~~The PLANG CI shall validate Production Requests for On-Demand Data Products against a pre-approved list of acceptance criteria.~~ | All identified acceptance criteria checked prior to receipt of On-Demand Production Requests by Planning Subsystem | PlOnDemandManagerNB<br>PlOnDemandPRNB |
| S-PLS-00180 | The PLANG CI shall ~~validate updates (modifications / cancellations) to existing Production Requests~~ provide the capability to create new production requests from existing production requests. | This modification and deletion of S-PLS-0200 reflect how system will actually work. Modification of PR is actually the deletion of old PR and entry of a new PR. | PlProductionRequest,<br>PlProductionRequestUI |
| ~~S-PLS-00200~~ | ~~The PLANG CI shall accept updates (modifications / cancellations) to Production Requests entered by the operations staff.~~ | See above comment for S-PLS-00180. | PlDataTypeB<br>PlDPRB<br>PlDataGranule<br>PlDataTypeCatalogue<br>PlPGE<br>PlProductionRequestB<br>PlProductionRequestUI<br>PlDataTypeReq |
| ~~S-PLS-00230~~ | ~~The PLANG CI shall provide production rules (via GUI) to break up large reprocessing Production Requests.~~ | This functionality is redundant with that of decomposing Production Requests (PRs) to Data Processing Requests (DPRs). | PlReprocPRNB<br>PlReprocessingStrategyNB |
| ~~S-PLS-00300~~ | ~~The PLANG CI shall accept ground events to describe the allocation of data processing resources to non-production tasks.~~ | Not consistent with current resource planning concept. | PlRpResourceReservation<br>PlGroundEvent |

*Table A-2. Intended Changes to CDR Requirements Baseline (2 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| New | The PLANG CI shall provide the capability to display a site resources plan covering an operator specified time interval, in text report and timeline display format, to a resolution of one minute which describes the exclusive allocation of planned site resources to default activities and ground events. | Implement resource planning. | PlRpResourcePlan, PlRpResourceReservation PlRpTimeline |
| New | The PLANG CI shall provide the capability to display a site resources plan covering an operator specified time interval, in text report and timeline display format, to a resolution of one minute which describes the shared allocation of planned site resources to default activities and ground events. | Implement resource planning. | PlRpResourcePlan, PlRpResourceReservation, PlRpTimeline |
| New | The PLANG CI shall, when a ground event is scheduled, identify and display (via GUI) conflicts with previously scheduled ground events. | Implement resource planning. | PlRpResourceReservation, PlRpResourceState, PlRpResource |
| New | The PLANG CI shall provide the capability to receive MSS site configuration management information for resource planning. | Implement resource planning. | PlRpResource, Baseline Manager |

*Table A-2. Intended Changes to CDR Requirements Baseline (3 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| New | The PLANG CI shall provide the capability to maintain a list of planned site resources via GUIs with the specific capabilities to add, update, delete, and query on site resources. | Implement resource planning. | PlRpResourcePlanningUI, PlRpResource |
| New | The PLANG CI shall provide the capability to associate a default activity to a planned site resource. | Implement resource planning. | PlRpResource |
| New | The PLANG CI shall be capable of setting up dependencies between services and hardware resources. | Implement resource planning, allow more extensive use of resoruce plan by production planning. | PlRpService, PlRpResource |
| New | The PLANG CI shall provide the capability to enter via GUIs site resource requests, to support ground events. | Implement resource planning. | PlRpResourceReservation |
| New | The PLANG CI shall provide the capability to maintain site resource requests, via GUIs, with the specific capabilities to add, update, delete, and query on site resource requests. | Implement resource planning. | PlRpResourceReservation |
| New | The PLANG CI shall provide the capability to review site resource requests, via GUIs, for the purpose of validation and to set or update the site resource request validation status field. | Implement resource planning. | PlRpResourceReservation |

*Table A-2. Intended Changes to CDR Requirements Baseline (4 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| New | The PLANG CI shall provide the capability to review site resource requests, via GUIs, and to set the status of the site resource request in one of the following states:<br>1) New<br>2) Validated<br>3) Approved<br>4) Rejected. | Implement resource planning. | PlRpResourceReservation |
| New | The PLANG CI shall provide the capability to generate a planned site resource report. | Implement resource planning. | PlRpResourceReservation, PlRpResourcePlan |
| New | The PLANG CI shall provide the capability to generate site resource request reports. | Implement resource planning. | PlRpResourcePlan |
| New | The PLANG CI shall provide the capability to generate reports providing a comparison of planned vs. actual resource usage. | Implement resource planning. | PlRpResourceReservation, PlRpResourcePlan |
| New | The PLANG CI shall be able to provide site resource plans to PLANG CI's at other sites | Implement resource planning. Enable cross-DAAC planning. | PlRpPublishedPlan |
| New | The PLANG CI shall be able to import saved site resource plans. | Implement resource planning. | PlRpPublishedPlan |
| New | The PLANG CI shall be able to save site resource plans to a file. | Implement resource planning. | PlRpPublishedPlan |
| New | The PLANG CI shall provide the capability to initiate a site ground event script associated with a resource request in the resource plan at the planned for time. | Implement resource planning. | PlRpExecutatable, PlGroundEventExecutable, PlResourceReservation |
| New | The PLANG CI shall log the start time of ground events it executes. | Implement resource planning. | PlResourceReservation |

### *Table A-2. Intended Changes to CDR Requirements Baseline (5 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| New | The PLANG CI shall log the end time of ground events it executes. | Implement resource planning. | PlResourceReservatio |
| S-PLS-00405 | The PLANG CI shall allow the conditions for execution of Product Generation Executives (PGEs) to include the values of ~~intermediate results such as~~ metadata fields of input data. | Clarification. | PlMetadataChecks |
| ~~S-PLS-00410~~ | ~~The PLANG CI shall support the capability to display (via GUI) a list of PGEs maintained in its PGE information database.~~ | Redundant with report generation capabilities. | PlPGEProfile |
| ~~S-PLS-00420~~ | ~~The PLANG CI shall support the capability to browse (via GUI) the information maintained on a specific PGE.~~ | Capabilities provided by AITTL. | PlPGEProfile |
| ~~S-PLS-00430~~ | ~~The PLANG CI shall support the capability to (a) allow (authorized) operations staff updates (enter / modify / delete) of PGE information in the Planning PGE information database, (b) maintain a record of updates made.~~ | Capabilities provided by AITTL. | PlPGEProfile |
| ~~S-PLS-00465~~ | ~~The PLANG shall maintain lists of input Granules in order to support the production of tile or spatial-based output Granules~~ | Not consistent with current design. | PlTileScheduledNB<br>PlOrbitScheduledNB |
| S-PLS-00490 | The PLANG CI shall ~~maintain Planning system fault data using fault isolation tools provided by the LSM~~ log planning system faults to MSS. | Clarification. | EcEvent |

### *Table A-2. Intended Changes to CDR Requirements Baseline (6 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| ~~S-PLS-00611~~ | ~~The operations staff shall manually submit (to the Data Server) Data Subscriptions for the Data Availability Schedules (DAS) of any remote ECS sites, any IP and any ODC that makes a DAS available~~ | Prodcedural requirement, not linked to any functional requirements. | PlDASNB |
| ~~S-PLS-00615~~ | ~~The operations staff shall manually submit (to the Data Server) Data Subscriptions for FOS plans and schedules.~~ | Prodcedural requirement, not linked to any functional requirements. | PlFOSDASFile |
| ~~S-PLS-00606~~ | ~~The PLANG CI shall send advertisement subscriptions to the IOS.~~ | Interface is no longer needed or used. | |
| S-PLS-00690 | The PLANG CI shall create a Candidate Plan specifying a timeline for PGE execution that will satisfy Production Requests for Standard Products consistent with available and allocated processing resources. | Clarification. | PlGroundActivity<br>PlPlanB<br>PlPGEActivity<br>PlPlanningWorkbenchUI<br>PlProductionRequestB<br>PlDPRs<br>PlDPRB<br>PlGroundEvents<br>PlGroundEvent |
| S-PLS-00700 | The PLANG CI shall create a Candidate Plan specifying a timeline for PGE execution that will satisfy Production Requests for Reprocessing and On-Demand Data Products consistent with available and allocated processing resources. | Clarification. | PlPRCollectionNB |
| New | The PLANG CI shall be able to provide a high-level, aggregate view of production plans. | Increase tool usability. | |

### *Table A-2. Intended Changes to CDR Requirements Baseline (7 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| S-PLS-00741 | The PLANG CI shall be capable of seperateing AI&T activities from the operational production environment. | Clarification. | PlPGEActivity<br>PlDPRB<br>PlPlanB<br>PlPlanningWorkbenchUI<br>PlResourceRequirement<br>PlDPRs |
| ~~S-PLS-00870~~ | ~~The operations staff shall manually submit Data Subscriptions for PGE input data to the appropriate Data Servers.~~ | Prodcedural requirement, not linked to any functional requirements. | PlSubscriptionSubmitIF<br>PlDataTypes<br>PlDataTypeB<br>IoAdAdvertisingSrv_C<br>IoAdServiceAdvertisement_C<br>IoAdServiceCollection_C |
| ~~S-PLS-00872~~ | ~~The operations staff shall manually submit Data Subscriptions for L0 data to the Ingest Subsystem.~~ | Prodcedural requirement, not linked to any functional requirements. | PlDataTypeB<br>PlDataTypes<br>PlSubscriptionSubmitIF |
| ~~S-PLS-00880~~ | ~~The operations staff shall manually cancel Data Subscriptions for input data to PGEs that are no longer used, once they determine that the input data is not required by any other PGE.~~ | Prodcedural requirement, not linked to any functional requirements. | DsClSubscription<br>PlDataTypeB<br>PlDataTypes<br>PlSubscriptionSubmitIF |
| S-PLS-01030 | The PLANG CI shall update the Active Plan with the ~~current processing~~ received status of each Data Processing Request listed. | Clarification based on current design. | PlPlanB<br>DpPrScheduler<br>PlPGEActivity<br>PlPlanningWorkbenchUI<br>PlDPRB<br>PlActivities |

*Table A-2. Intended Changes to CDR Requirements Baseline (8 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| S-PLS-01200 | The PLANG CI shall provide the operations staff with the capability to perform the following on-line functions, via GUI:<br>a. Entry of product requests for standard products,<br>b. Query / update / cancellation of production requests for standard products,<br>c. Query status of production requests,<br>d. Query / update of production rules and PGE information, | Clarification made in light of resource planning design. | PlPlanningWorkbenchUI |
| S-PLS-01240 | The PLANG CI shall support the display (via GUI) of Planning ~~hardware and~~ software detected faults to the operations staff. | Only software faults are logged with this mechanism. Hardware faults are logged via HP Openview. | EcEvent |
| ~~S-PLS-01450~~ | ~~The PLANG CI shall collect Configuration Management Data and provide it to the MSS.~~ | Interface is no longer needed or used. | PLANG |
| S-PLS-02010 | The PLANG CI shall be able to identify scheduling conflicts in site production plans. | Clarification | PlPlanB |
| S-PLS-02030 | The PLANG CI shall identify conflicts in site production plans caused by cross-DAAC data dependencies. | Clarification | PlPlanB |
| S-PLS-02040 | The PLANG CI shall be able to display (via GUI) cross-DAAC data dependencies in production plans. | Clarification | PlPlanB |
| S-PLS-02050 | The PLANG CI shall be able to provide site production plans to PLANG CIs at other sites. | Clarification | PlExportedPlanNB |

### *Table A-2. Intended Changes to CDR Requirements Baseline (9 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| S-PLS-02060 | The PLANG CI shall be able to account for cross-DAAC data dependencies in the site production plans it generates. | Clarification | PlPlanB |
| S-PLS-02070 | The PLANG CI shall be able to integrate multiple DAAC site production plans to produce a coordinated plan. | Clarification | PlTimelineDisplay |
| S-PLS-02200 | The PLANG CI shall have the capability to ~~automatically~~ extract ~~pertinent scheduling information based on operator supplied criteria~~ temporal subsets from a production or resource plan and save them to a file, | Clarification based on design. | PlPublishedPlan |
| New | The PLANG CI shall have the capability to extract subsets based on user selected Production Requests and save them to a file. | Clarification based on design. | PlPublishedPlan |
| New | The PLANG CI shall provide a list of replan events which will cause the user to be notified and given the option to replan. | Implementing Limited automatic replan capability. | PlReplanCriteriaUI, PlReplanCriteria |
| New | The PLANG CI shall consider the creation of a new resource plan to be a replan event if it changes the availability of hardware resources less than a configurable amount of time in the future. | Implementing Limited automatic replan capability. | PlResourceChange |

*Table A-2. Intended Changes to CDR Requirements Baseline (10 of 10)*

| L4 Rqmt ID | L4 Requirement Text Intended Change | Reason for Change | Object Class, CSC, or CI |
|---|---|---|---|
| New | The PLANG CI shall consider the arrival of a new Predicted Data Availability Schedule to be a replan event if it indicates a delay in the predicted arrival of data by more than a configurable (for that particular data type) amount of time. | Implementing Limited automatic replan capability. | PlDASDifferent, PlDASDelta |
| New | The PLANG CI shall consider the submission of an On-Demand Prodution Request to be a replan event if the resource requirements exceed predefined thresholds. | Implementing Limited automatic replan capability. | PlOnDemandExceed, PlOnDemandReplanValues |
| New | The PLANG CI shall have the capability of providing an estimate of the resource usage for a production request prior to the inclusion of that request in a production plan. | Implementing Limited automatic replan capability. | PlProductionRequestUI |
| New | The PLANG CI shall require confirmation of the selection of a version of a PGE for which a more recent version exists. | New capability to minimize human error in entry of Production Requests. | PlProductionRequestUI |

# Abbreviations and Acronyms

| | |
|---|---|
| ADSRV | advertising service CSCI |
| AHWGP | Ad Hoc Working Group on Production |
| AI&T | algorithm integration and test |
| AITTL | algorithm integration and test tools (CSCI) |
| AM-1 | EOS AM Project (morning spacecraft series) |
| ASCII | American Standard Code for Information Exchange Interchange |
| CASE | computer aided software engineering |
| CDR | Critical Design Review |
| CERES | Clouds and Earth's Radiant Energy System |
| CI | configuration item |
| COTS | commercial off-the-shelf (hardware or software) |
| CPU | central processing unit |
| CSC | computer software component |
| CSCI | computer science configuration item |
| CSMS | Communications and Systems Management Segment (ECS) |
| DAAC | distributed active archive center |
| DAS | Detailed Activity Schedule (created by FOS) |
| DAT | Data Availability Times |
| DBMS | database management system |
| DDSRV | document data server |
| DEV | developed code |
| DPR | data processing request |
| ECS | EOSDIS Core System |
| EDF | ECS development facility |
| EDOS | EOS Data and Operations System |
| EOS | Earth Observing System |
| EOSDIS | EOS Data and Information System |
| ESDT | Earth science data types |
| ESN | EOSDIS Science Network (ECS) |
| FDDI | fiber distributed data interface |
| GB | gigabyte ($10^9$) |
| GC | global change |

| | |
|---|---|
| GUI | graphic user interface |
| FOS | Flight Operations Segment (ECS) |
| HCL | Hughes class library |
| HTML | Hyper-Text Markup Language |
| HWCI | hardware configuration item |
| I/O | input/output |
| L0 | Level 0 data |
| LAN | local area network |
| LaRC | Langley Research Center (DAAC) |
| MB | megabyte ($10^6$) |
| MSFC | Marshall Space Flight Center (DAAC) |
| MSS | Management Subsystem |
| NOAA | National Oceanic and Atmospheric Administration |
| OO | object oriented |
| OODCE | Object Oriented Distributed Computing Environment |
| OPR | On-demand Production Requests |
| PDAS | Planning Data Availability Schedule (created by PLANG) |
| PDPS | planning and data processing system |
| PDR | Preliminary Design Review |
| PGE | Product Generation Executive |
| PLANG | production planning CSCI |
| PR | production request |
| RAID | redundant array of inexpensive disks |
| RDBMS | relational database management system |
| RMA | reliability, maintainability, availability |
| SCSI | Small Computer System Interface |
| SDPF | Sensor Data Processing Facility (GSFC) |
| SDPS | Science Data Processing Segment (ECS) |
| TRMM | Tropical Rainfall Measuring Mission (joint US-Japan) |
| U/I | user interface |
| UNIX | POSIX operating system |
| WAIS | Wide Area Information Server |
| WWW | World Wide Web |