

4.5 CSCI Dynamic Model

As part of the PDPS design process, the group of existing scenarios which were developed for the Release A PDR and CDR have been updated. As a result of the changes in the boundaries of Planning and Processing and the selection of AutoSys and AutoXpert products, the scenarios have undergone a great deal of modification. The basic scenario for initiation of a job as performed by AutoSys is presented in Section 4.1.4 and accompanying Figures 4.1-1 and 4.1-2. This information should be referenced as these other scenarios are reviewed. Before the scenarios, an introduction section has been provided which provide a list of assumptions used in developing the scenarios. These assumptions explain the interfaces and the given state of the ECS system and subsystems.

The Processing CSCI Dynamic Model is represented by a group of scenarios and event traces which have been developed to further refine the role, responsibilities and activities of the Processing CSCI during SDPS operations. The scenarios provide an abstract section to describe the system context of the processing, and a detailed scenario description of the required Processing CSCI activities. Event Traces have been developed for each scenario to show the interactions between the objects involved in the scenario. There may be more than one event trace for each scenario depending on the complexity of processing represented in the scenario. Because of the COTS-intensive aspects of the Processing CSCI design, the scenarios provide descriptions of events which occur in the custom code which is being developed to support the COTS' products. In the scenario, a brief description of the COTS activities is provided, but within the event traces, the COTS is treated as a black box. Each scenario provides a listing of the classes from the object model referenced in the scenario. The scenarios have been grouped around significant Processing CSCI functional areas. These groupings are as follows:

- a. Job Management Scenarios—These scenarios are presented to explain certain concepts to understand the integrated view between AutoSys and the Processing CSCI custom components. A scenario is provided which describes how information provided in a Data Processing Request is used to create the jobs required to execute a PGE. Also provided are scenarios which describe the Processing CSCI activities involved with the input of a daily schedule of Data Processing Requests from Planning, providing status information to Planning, activating jobs for execution, updating job information, suspension and resumption of jobs, and canceling of jobs.
- b. Data Management Scenarios—These scenarios describe the Processing CSCI activities which occur to support the following activities:
 1. Staging and destaging of data from and to the Data Server.
 2. Retaining of data on Science Production Hardware to support further production
 3. Deletion of data not required to support further production.
 4. Movement of data from one production resource to another to support further production.
 5. Support for pre-processing of data for production.
- c. PGE Execution Management Scenarios—These scenarios describe the activities which occur to initiate and manage the execution of a PGE.

- d. Resource Management Scenarios - These scenarios describe activities related to the management of Processing CSCI hardware resources.
- e. Quality Assurance Scenarios - These scenarios present high-level views of DAAC manual quality assurance and DAAC non-science quality assurance activities.

4.5.1 Scenario Assumptions

The following information details ECS system assumptions which should be followed while reviewing these scenarios. These assumptions define information on Processing CSCI interfaces as well as subsystem and system state information. Unless specified otherwise, this information remains consistent for each scenario. For the event traces, the Job Scheduling COTS is represented as a class. This is an abstract class and is used to represent the stimuli which result in Processing CSCI activities.

4.5.1.1 Interfaces

Table 4.5-1 shows the service of processing CSCI interfaces with other subsystems.

Table 4.5-1. Processing CSCI Interfaces With Other Subsystems

Subsystem	Service
CSS	Supplies the communication mechanisms used between ECS Applications. These mechanisms are based on OODCE services. Processing CSCI uses these mechanisms to communicate with MSS, Data Server, and Ingest.
MSS	1) Provides services to startup and shutdown ECS applications. MSS would initiate activities to startup and shutdown the PDPS Database, AutoSys' Database, AutoSys' Event Processor, and other required components needed at startup. Also, in the event of a shutdown, MSS would shutdown the components in the required order. 2) Provides services to log system events, i.e. Fault and exception handling information, in order to notify all impacted parties. 3) Retains accounting and configuration management data about resources used by the Processing CSCI.
Data Server	Provides services to stage (Acquire) and destage (Insert) data from Processing hardware. Also, the Q/A Monitor Operations position requires subscription services of the Data Server.
Ingest	Same as Data Server. Ingest is a specialized Data Server
Planning	Provides Data Processing Request information to Processing. The interface mechanisms occur through AutoSys provided command-line interfaces and APIs and through the PDPS Database.
Interoperability	Provides advertisement information for the Data Processing Subsystem to build Data Server Subscription information from.

4.5.1.2 System and Subsystem States

During these scenarios, all ECS application components are considered to be operating in a steady state fashion, unless specified otherwise. Some of the scenarios do deal with failure situations and the results of these failures.

4.5.2 Job Management Scenarios

These scenarios describe the activities associated with managing the AutoSys job schedule. The activities discussed include creation of the job schedule, cancellation of a job, suspension of a job, resumption of a job, and modification of a job or providing of status of a job. Also, the scenarios

explain the interface which exists between the Planning CSCI and the COTS (AutoSys) component of the Processing CSCI. Each of these activities is initiated by a Planning CSCI component. A brief overview of Planning CSCI activities follows.

The Production Planner (Operations Person) or the Planning CSCI is responsible for the creation of candidate production plans. These plans provide predicted views of science data production based on certain criteria, such as resource availability schedules and different standard production priorities. Once the Production Planner decides on the best candidate plan, this candidate plan is activated. (See the Planning Subsystem Design Specification for more information on these Planning activities). According to current DAAC Operations' strategies, a section of this active plan, known as the daily job schedule, will be fed into the AutoSys Database at the beginning of the day. These jobs are the jobs which will be processed for a particular day. Because of the selection of AutoSys and its capabilities to manage job dependencies, all Data Processing Requests will be fed into AutoSys at the beginning of the day. The Data Processing Requests which do not have all data dependencies fulfilled would be kept in a "HELD" state until the dependencies are fulfilled. Upon the meeting of all data dependencies, the Planning CSCI would release the job. To limit the reach of the AutoSys specific interfaces, a layer of software, sometimes referred to as Glue Software, has been developed to promote the integration of AutoSys with ECS custom software applications. This allows Planning and Processing CSCI component data structures and operation primitives to be associated with AutoSys specific primitives with limited impact to the overall implementation of these components. In the Processing CSCI, this Glue Software is represented by the classes which are part of the Job Management component. One of these classes, DpPrCotsManager, encapsulates the AutoSys specific command-line interfaces and APIs used to communicate with AutoSys.

4.5.2.1 Create a DPR Job

4.5.2.1.1 Abstract

This scenario describes the activities involved in creating a DPR job. A CreatedprJob is called when Planning Subsystem activates a plan. DpPrDataManager will initialize the data map for the data related with this DPR. A predictive data staging job, a job box and a resource allocation job will be created for this DPR. These jobs will be added to Autosys according to Job definition such as priority, predicted start time, and required resources. The start time of the predictive staging job will be calculated based on the staging time from PIPerformance and the predicted start time of the DPR.

4.5.2.1.2 Stimulus

Production Planner initiates daily job schedule process. The operation CreatedprJob in DpPrScheduler is called.

4.5.2.1.3 Desired Response

The predictive data staging job is created (if required) for this DPR. The job box and the resource allocation job are created for this DPR.

4.5.2.1.4 Participating Classes from the Object Model

- PIDPRB
 - DpPrScheduler
 - DpPrCotsManager
 - Cots
 - DpPrDataManager (DPR only)
 - PIPGE
 - PIPerformance

4.5.2.1.5 Description

When the CreateDprJob operation is called by DpPrScheduler, the following steps will occur:

a. The DpPrScheduler will retrieve information as needed from the PIDPRB to perform the following steps for each DPR:

(1) DpPrDataManager will initialize the data map for the DPR.

(2) Job Scheduler will query the predictive staging flag for this DPR through PIDPRB.

- a. If the flag is true, get predicted start time for the current job.
- b. Get the duration of previous predictive staging time for this DPR from PIPerformance
- c. Get the predicted start time from PIDPRB
- d. Get data processing request type
- e. Predict the staging time duration:

Depending on whether the data processing request type is routine or reprocessing, loop through either the routine or reprocessing predictive staging input list from PIPGE.

Get the unit staging time of the input data from PIPGE and compute the input data size from PI DataGranule. Set the predictive staging time duration to be the total of the input data size multiplied by the unit staging time of each input data granule.

f. Set the start time of this predictive staging to be the predicted start time minus the predictive staging time duration.

(3) Create the job box and the resource allocation job contained in the job box.

These jobs are considered "on hold" until Planning Subsystem releases the DPR.

END OF SCENARIO.

4.5.2.1.6 Event Traces

Figure 4.5-1 shows the create data processing request job event trace.

Figure 4.5-2 shows the create ground event job event trace.

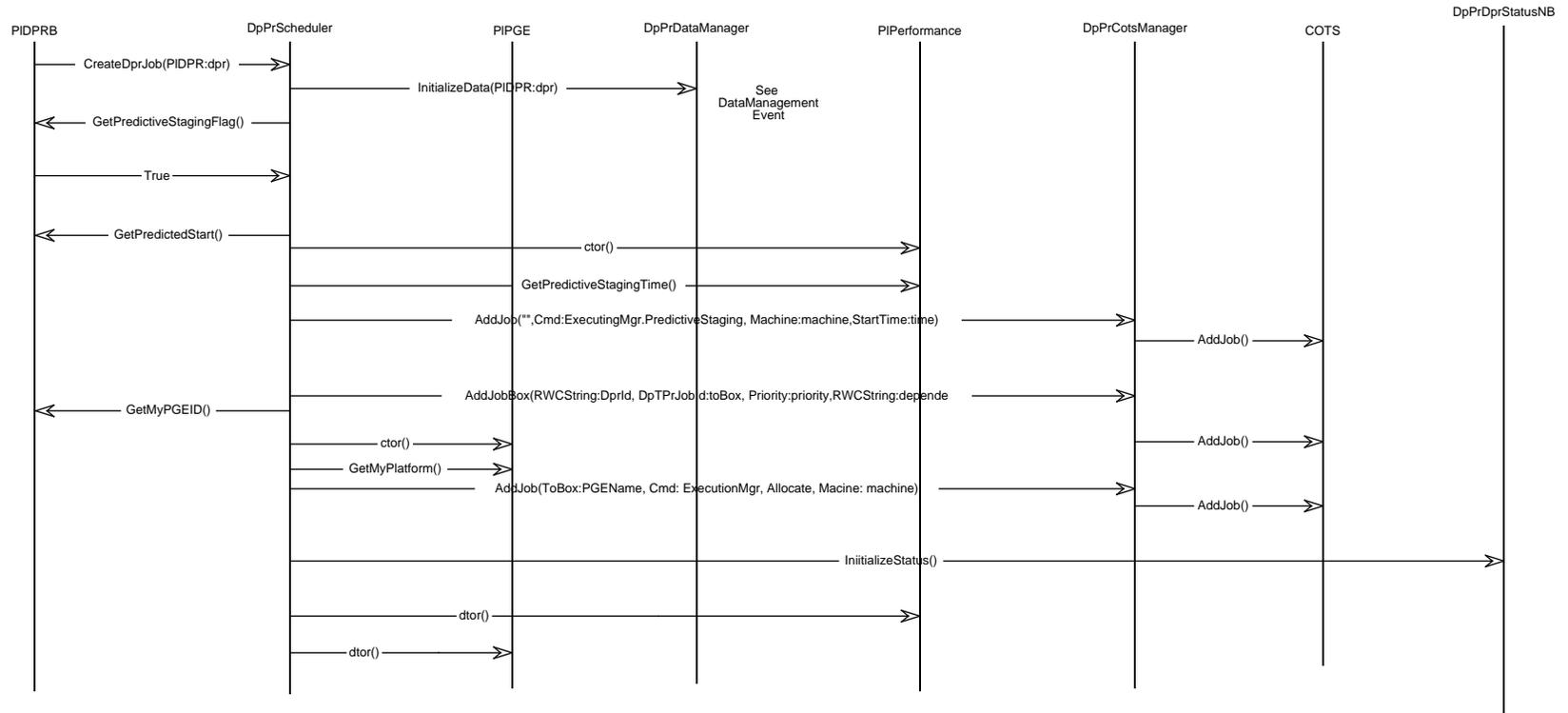


Figure 4.5-1. Create Data Processing Request

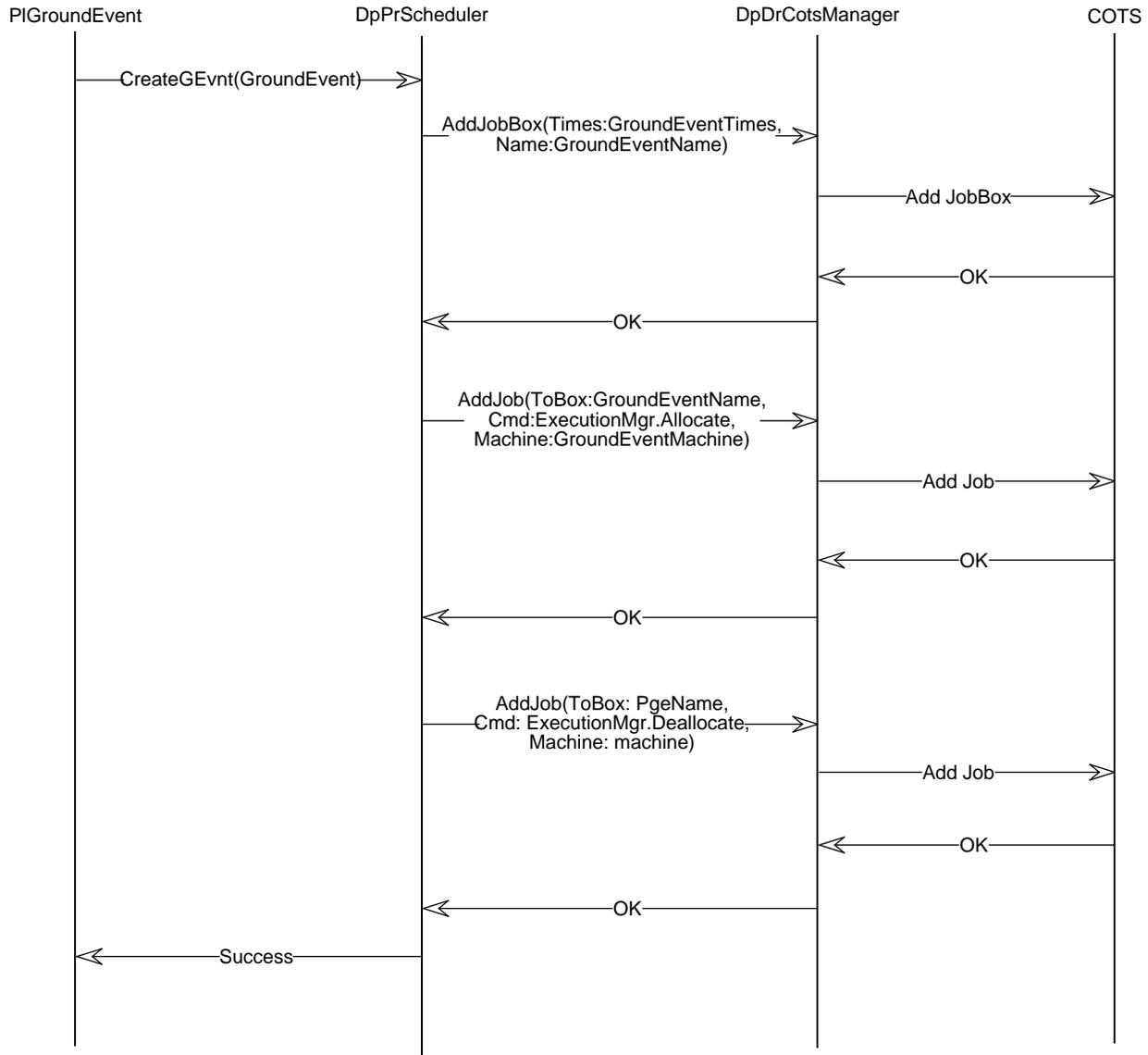


Figure 4.5-2. Create Ground Events

4.5.2.2 Release Data Processing Request (DPR) Job

4.5.2.2.1 Abstract

When a Data Processing Request is fed into AutoSys through the CreateDPRJob operation on the DpPrScheduler class, the resulting job box is set in the "HELD" state in the AutoSys Database. This results in the jobs not being processed, or managed, until the job box (referred to herein as the "DPR job") is released. The Planning CSCI will release a DPR job when all the data subscriptions for that DPR are fulfilled. Therefore, for all Data Processing Requests which have no outstanding data subscriptions, the DPR jobs will be initiated as "HELD" and then immediately released. Other DPR jobs which have outstanding data subscriptions will reside in the "HELD" state until Planning has been notified that the data is available from the Science Data Server.

When the Planning CSCI has determined that no outstanding data subscriptions exist for a DPR job, that DPR job is released by using the ReleaseDPRJob operation in the DpPrScheduler class. The result of this operation will be that the DPR job and the individual jobs within it become active in the AutoSys job stream, and will be processed accordingly. The Scheduling COTS product examines the released job to see if it has any dependencies on previous jobs; if so, and those dependencies have not been satisfied, the current DPR is prevented from running until they are. If those dependencies have been met, the jobs in the job box start executing. (See Figure 4.5-3.)

The series of steps in the following scenario description are repeated for each Data Processing Request which is scheduled for processing for on a particular day.

4.5.2.2.2 Stimulus

Upon receipt of the last outstanding Subscription Notification for a Data Processing Request, the Planning CSCI will use the ReleaseDPRJob operation in the DpPrScheduler class. The DpPrScheduler and related classes encapsulate the AutoSys specific command-line interfaces and APIs to used to communicate with AutoSys.

4.5.2.2.3 Desired Response

The specified DPR job and the individual jobs within it are activated in the AutoSys job stream and processed accordingly. Log Information to record the event will be collected and stored for review at a later time.

4.5.2.2.4 Participating Classes from the Object Model

- PIDPRB
- DpPrScheduler
- DpPrCotsManager
- Cots
- DpPrDataManager
- DpPrExecutionManager
- DpPrExecutable

4.5.2.2.5 Description

- 1) When the ReleaseDPRJob operation is initiated on the DpPrScheduler, the following steps will occur:
 - a) The DpPrScheduler will release the job in the AutoSys Database. Figure 4.5-3, Release Data Processing Request Event Trace, shows the events to needed to support this step.
- 2) If the DPR Job's dependencies on prior jobs have been fulfilled, the Job Box will begin to execute. Figure 4.5-29, Expand Jobs Event Trace, shows the events to needed to support this step.
- 3) END OF SCENARIO.

4.5.2.2.6 Event Traces

Figure 4.5-3 shows the release data processing request job event trace.

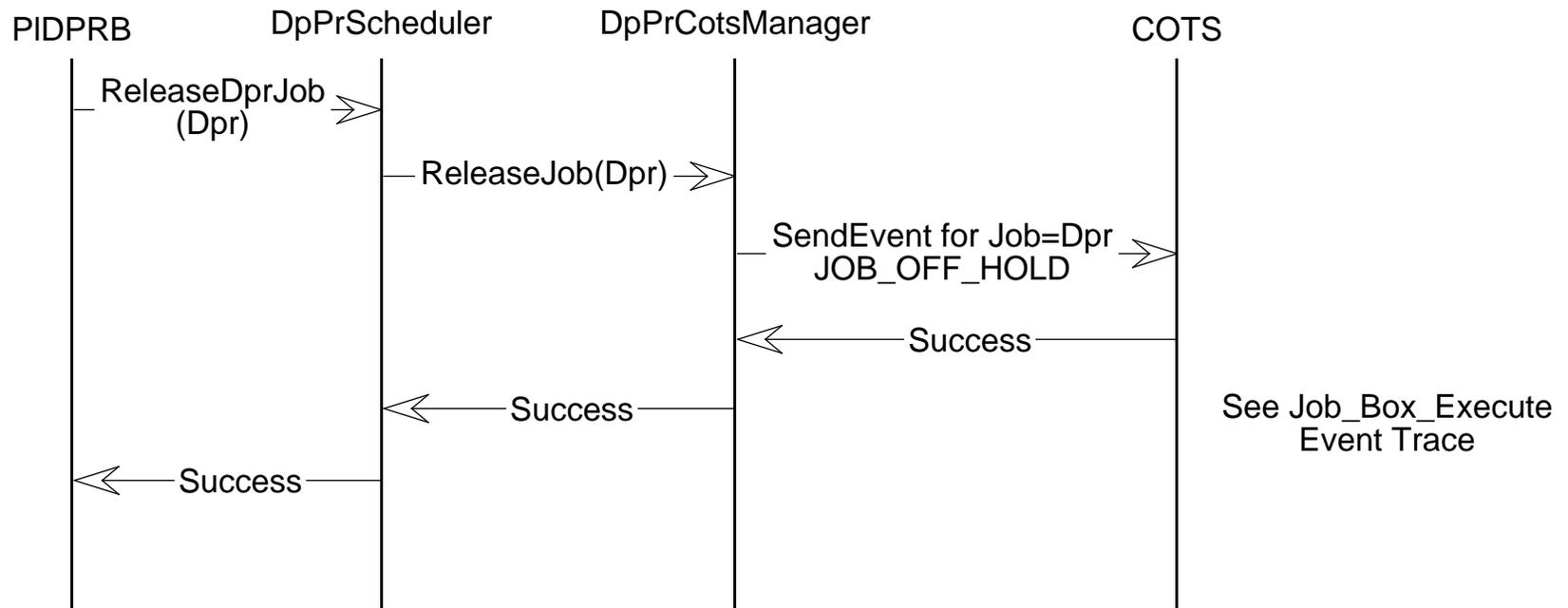


Figure 4.5-3. Release Data Processing Request

4.5.2.3 Cancel Processing Request or Ground Event Job

4.5.2.3.1 Abstract

This scenario describes the processing required to cancel further processing of a Data Processing Request or Ground Event. The Planning CSCI will cancel a DPR job/Ground Event job when the job's associated Request/Event is no longer relevant. This could occur when the Production Plan has been modified or when the processing of a particular Production Request has been discontinued. The result of the cancellation is that the processing of the job will be halted. No further processing for this job will be performed unless the job is re-created and added back into the AutoSys job stream.

For Data Processing Request jobs, if the PGE associated with the Data Processing Request was executing at the time of cancellation, this execution will be terminated. If the PGE was not executing, but the data staging process was in progress, this process will also be terminated, and all resources reserved to process this PGE will be freed.

A Ground Event Job is the method used by Planning to make resources unavailable to support production. A Ground Event is input by the Resource Planner who uses Planning CSCI software to make a Resource Availability Schedule. This schedule is made available to the Planning CSCI and is used as an input to the Production Plan. To schedule a resource for maintenance, system test, or some other defined ground event, a ground event job is input into the AutoSys Job Schedule.

4.5.2.3.2 Stimulus

The initiation of the CancelDPRJob or CancelGEvnt operation on the JobScheduler class.

4.5.2.3.3 Desired Response

The following actions will occur during Cancel Data Processing Request processing:

- a. The execution of the PGE associated with the Data Processing Request will be terminated, if executing, or canceled, if awaiting execution.
- b. The COTS operations display will be updated.
- c. If allocation of resources had occurred to support the execution of the PGE or the implementation of the Ground Event, the allocated resources will be freed.
- d. Log Information to record the event will be collected and stored for review at a later time

4.5.2.3.4 Participating Classes From the Object Model

- PIDPRB (DPR only)
- DpPrScheduler
- DpPrCotsManager
- Cots
- DpPrDataManager (DPR only)
- DpPrExecutionManager
- PIGroundEvent (Ground Event only)

4.5.2.3.5 Scenario Description

- a) When the CancelDPRJob or CancelGEvnt operation is initiated on the DpPrScheduler, the following steps will occur:
 - 1) The DpPrScheduler will cancel the job in the AutoSys Database.
 - 2) If the PGE associated with Data Processing Request is executing, the PGE is terminated. No attempt is made to complete execution. By indicating cancellation, Planning has indicated that the processing should not be accomplished.
 - 3) If resources (such as disk space or CPU time) have been reserved for the PGE or Ground Event, these will be deallocated so another job can use them.
- b) END OF SCENARIO.

4.5.2.3.6 Event Traces

Figure 4.5-4 Cancel Ground Event Job Event Trace, shows the steps required to cancel a Ground Event job. The Ground Event Job is used to take resources out of the active job schedule so that maintenance, system test, or special events can be supported.

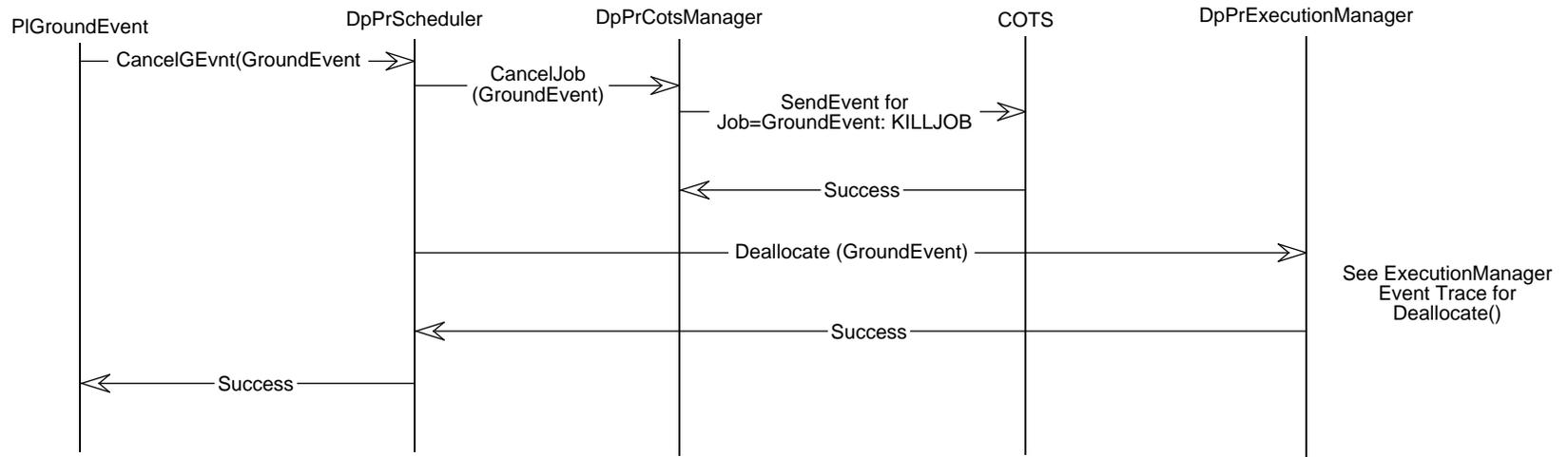


Figure 4.5-4. Cancel Ground Event

4.5.2.4 Update Data Processing Request Job

4.5.2.4.1 Abstract

This scenario describes the processing required to update the AutoSys job definition associated with a Data Processing Request. This process is initiated through the UpdateDPRJob operation in the DpPrScheduler class. The job definition information available for modification includes priority information and time information, such as start times, predicted execution times and ending time restrictions. Other Data Processing Request information can only be modified through a Planning CSCI interface.

4.5.2.4.2 Stimulus

The Planning CSCI will use the UpdateDPRJob operation in the DpPrScheduler class. The DpPrScheduler and associated classes encapsulate the AutoSys specific command-line interfaces and APIs to used to communicate with AutoSys.

4.5.2.4.3 Desired Response

The job definition of the Data Processing Request which resides in the AutoSys Database will be modified. AutoSys will add a log entry to its database to record this event.

4.5.2.4.4 Participating Classes from the Object Model

- PIDPRB
- DpPrScheduler
- DpPrCotsManager
- Cots

4.5.2.4.5 Description

- a) When the UpdateDPRJob operation is initiated on the DpPrScheduler, the following steps will occur:
 - 1) The DpPrScheduler will update the job definition and the predictive staging job associated with the Data Processing Request in the AutoSys Database.
- b) END OF SCENARIO.

4.5.2.4.6 Event Trace

Figure 4.5-5 shows the update data processing request job event trace.

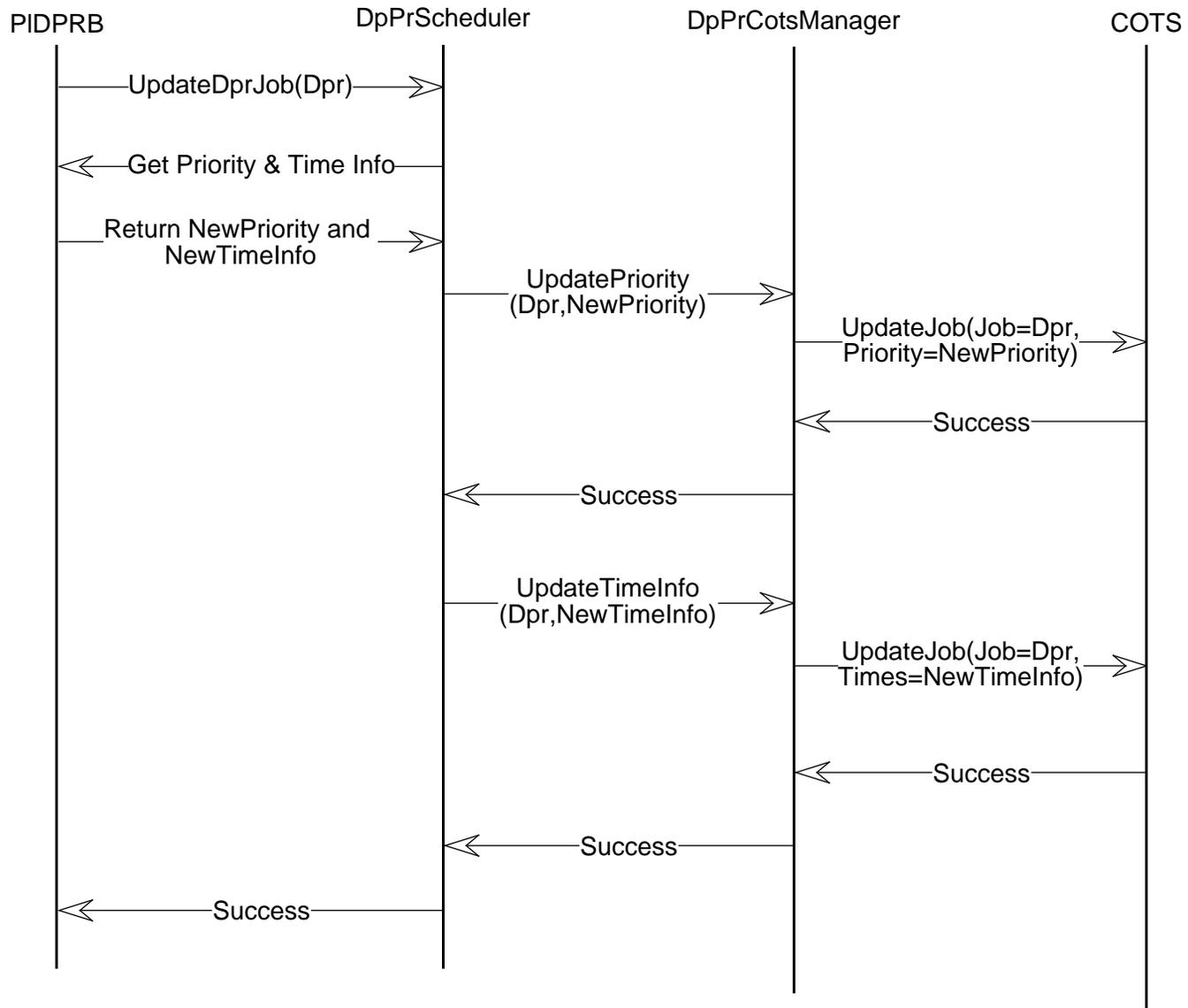


Figure 4.5-5. Update Data Processing Request Job

4.5.2.5 Get Data Processing Request Job Status

4.5.2.5.1 Abstract

This scenario describes the processing required to provide status information about the processing of a Data Processing Request to the Planning CSCI. The status of a Data Processing Request is provided during the AutoSys defined life of a Data Processing Request. This status information will provide the Planning CSCI a mechanism to judge the actual performance of the active plan compared to the predicted plan activities. Also, this will be used by Planning as a means of updating the Plan to determine the next group of activities to be provided to AutoSys.

4.5.2.5.2 Stimulus

The Planning CSCI will use the GetDPRJobStatus operation in the DpPrScheduler class. The DpPrScheduler and associated classes encapsulate the AutoSys specific command-line interfaces and APIs to used to communicate with AutoSys.

4.5.2.5.3 Desired Response

The status of the DPR job in the AutoSys job stream will be returned to Planning.

4.5.2.5.4 Participating Classes from the Object Model

- PIDPRB
- DpPrScheduler
- DpPrCotsManager
- Cots

4.5.2.5.5 Description

- a. When the GetDPRJobStatus operation is initiated on the DpPrScheduler, the following steps will occur:
 1. Retrieve job status information from the AutoSys database.
 2. Create Status Report.
 3. Return Status Report to the Planning CSCI.
- b. End of Scenario

4.5.2.5.6 Event Trace

Figure 4.5-6 shows the get data processing request job status event trace.

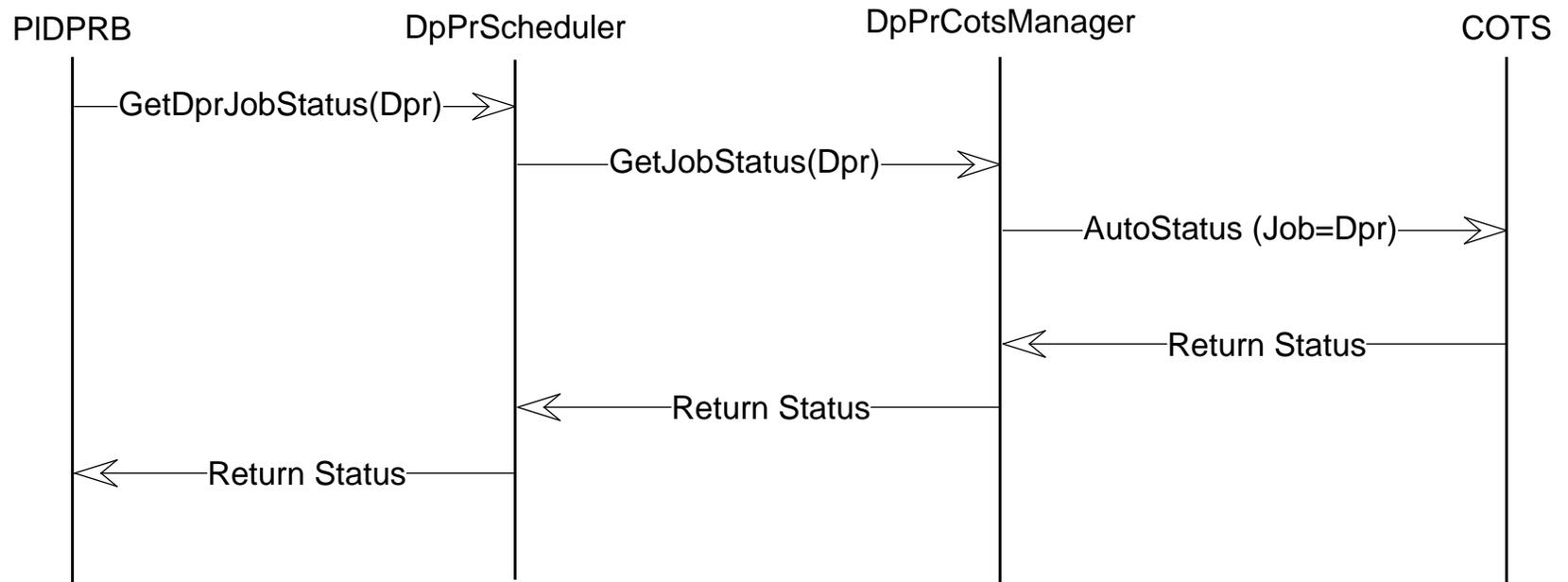


Figure 4.5-6. Get Data Processing Request Job Status

4.5.2.6 Suspend a DPR Job

4.5.2.6.1 Abstract

This scenario describes the processing of a Suspend DPR command issued to the Processing CSCI through the Planning CSCI by a user or operational staff member. The intention to suspend a job is to stop the job temporarily and either resume or cancel its normal processing at a later time. If a job is suspended, the storage space associated with it will not be deallocated. Therefore if too many jobs are suspend, the performance of the processing system will degrade because of tied-up disk resources. If a user tries to suspend a DPR which is already completed or does not exist, the Processing CSCI will ignore the command and return a warning status to the user.

4.5.2.6.2 Stimulus

If a user sends a suspend job request to the Planning CSCI, the Planning CSCI will pass the request to the Processing CSCI, and the suspend DPR job command is activated.

4.5.2.6.3 Desired Response

The Processing CSCI will examine all jobs constructed for this DPR, find the state of these jobs, and suspend any "normal" (not staging or destaging) jobs that are running. The state of any associated jobs that are not running is changed to suspended. Any staging or destaging operations for such a suspended job will be allowed to complete. If there is no job constructed for this DPR or all jobs constructed for this DPR are complete, a nonexistence status warning status is returned.

4.5.2.6.4 Participating Classes from the Object Model

- PIDPRB
- DpPrScheduler
- DpPrDprStatusNB
- DpPrCotsManager

4.5.2.6.5 Scenario Description

When a user or operational staff issues a suspend DPR command, the following activities will occur:

The SuspendDprJob operation of DpPrScheduler will query for all jobs constructed for this DPR through DpPrDprStatusNB. If a job has not been completed yet and is not staging or destaging job, the SuspendDprJob operation will construct a script to suspend this job and then add a job, which will run this script, to the Autosys database by calling AddJob operation of DpPrCotsManager.

4.5.2.6.6 Event Trace

Figure 4.5-7 shows the suspend data processing request event trace.

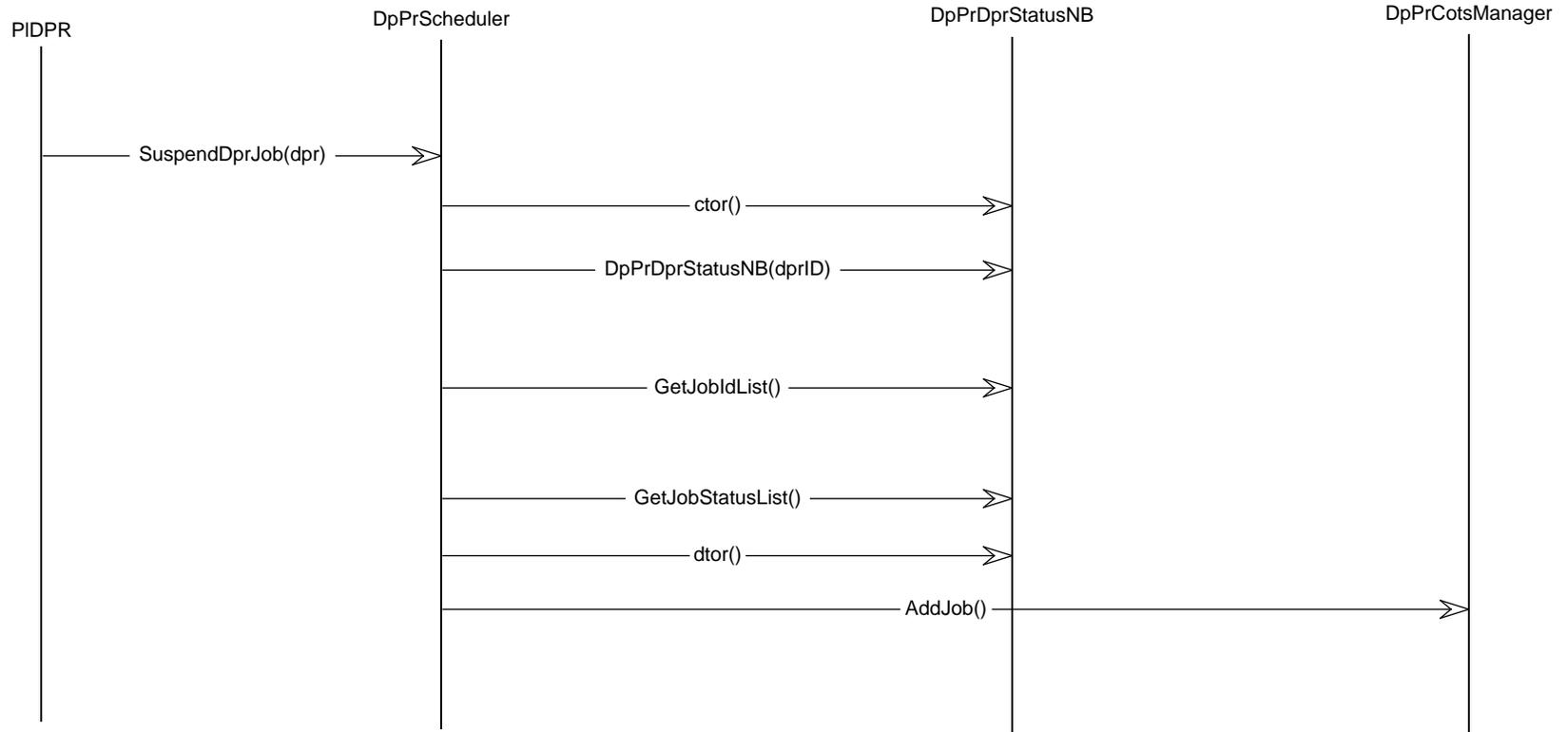


Figure 4.5-7. Suspend a Data Processing Request Job

4.5.2.7 Resume a DPR Job

4.5.2.7.1 Abstract

This scenario describes a Resume DPR command issued to the Processing CSCI through the Planning CSCI by a user or operational staff member. Resume a DPR job means that all suspended jobs constructed for this DPR should be resumed. If some of the jobs constructed for this DPR were completed (or allowed to complete in the case of data staging or destaging jobs), they will be in the completed state and resume action is not needed. If a user tries to resume a DPR which is already completed or does not exist, the Processing CSCI will ignore the command and return a nonexistence warning status to the user.

4.5.2.7.2 Stimulus

If a user sends a resume DPR job request to the Planning CSCI, it will pass the request to the Processing CSCI and the resume DPR job command is activated.

4.5.2.7.3 Desired Response

The Processing CSCI will examine all jobs constructed for this DPR, find the state of these jobs and resume the jobs that are suspended. If there is no job constructed for this DPR or all jobs constructed for this DPR are complete, a nonexistence status warning status is returned.

4.5.2.7.4 Participating Classes from the Object Model

- PIDPRB
- DpPrScheduler
- DpPrDprStatusNB
- Cots

4.5.2.7.5 Scenario Description

a. When a user or operational staff issues a resume DPR command, the following activities will occur:

The ResumeDprJob operation of DpPrScheduler will query for all jobs constructed for this DPR through DpPrDprStatusNB. If a job is suspended, this operation will construct a script to resume the suspended job and add a job, which will run this script, to the Autosys database by calling AddJob operation of DpPrCotsManager.

4.5.2.7.6 Event Trace

Figure 4.5-8 shows the resume data processing request event trace.

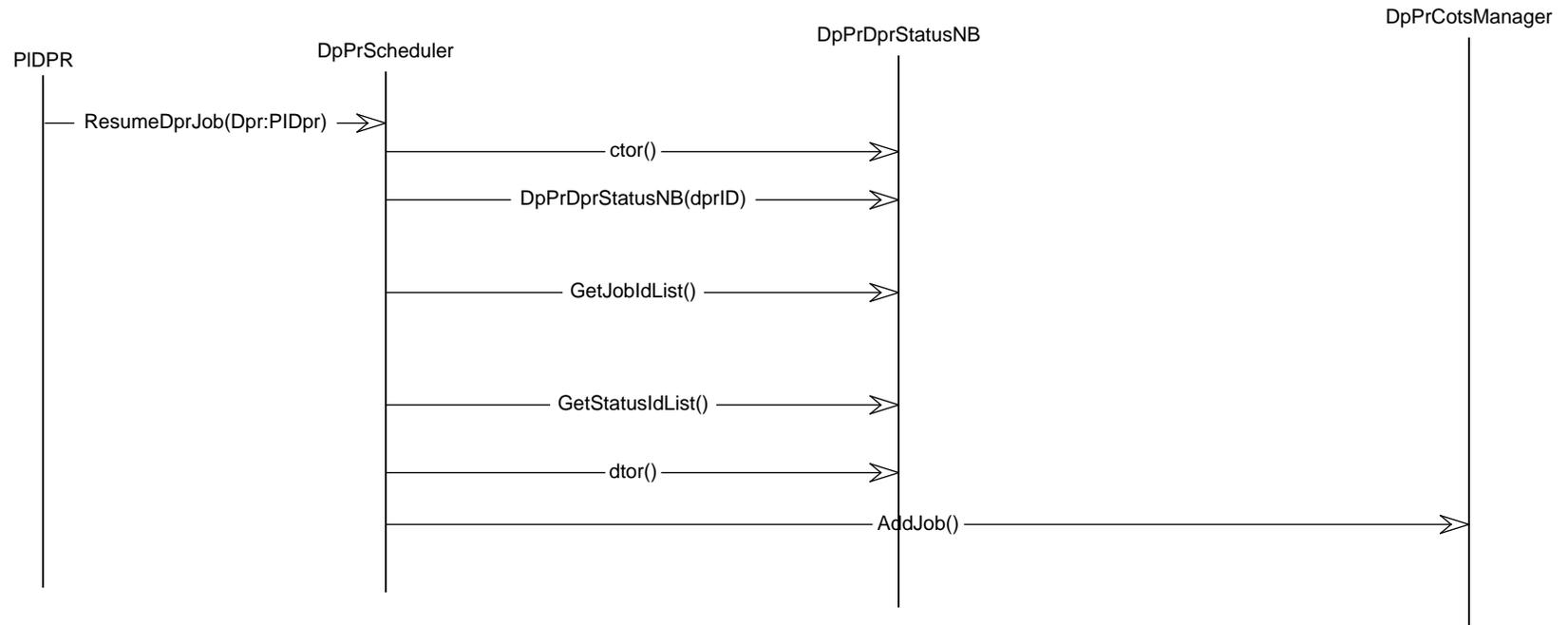


Figure 4.5-8. Resume a Data Processing Request Job

4.5.2.8 Report Status

4.5.2.8.1 Abstract

This scenario describes the process required to update status information of a DPR after the deallocation job is completed. The status information of a DPR is provided to Planning Subsystem whenever a job in the job box is completed.

4.5.2.8.2 Stimulus

When AutoSys finishes all the jobs of a PGE, it starts destaging data using DeallocateData operation in DpPrDataManager and updating the DPR status.

4.5.2.8.3 Desired Response

The status of a DPR is updated and returned to Planning.

4.5.2.8.4 Participating Classes from the Object Model

- Cots
- DpPrDataManager
- DpPrDprStatusNB
- PIDPRB

4.5.2.8.5 Description

When Autosys activates DeallocateData operation of DpPrDataManager, DeallocateDatae operation will construct DpPrDprStatusNB class, update the job status of this deallocation job and the status of the DPR to "completed". Finally it calls the ReportCurrentDprStatus operation to report the status change to PIDPRB.

4.5.2.8.6 Event Trace

Figure 4.5-9 shows the report status event trace.

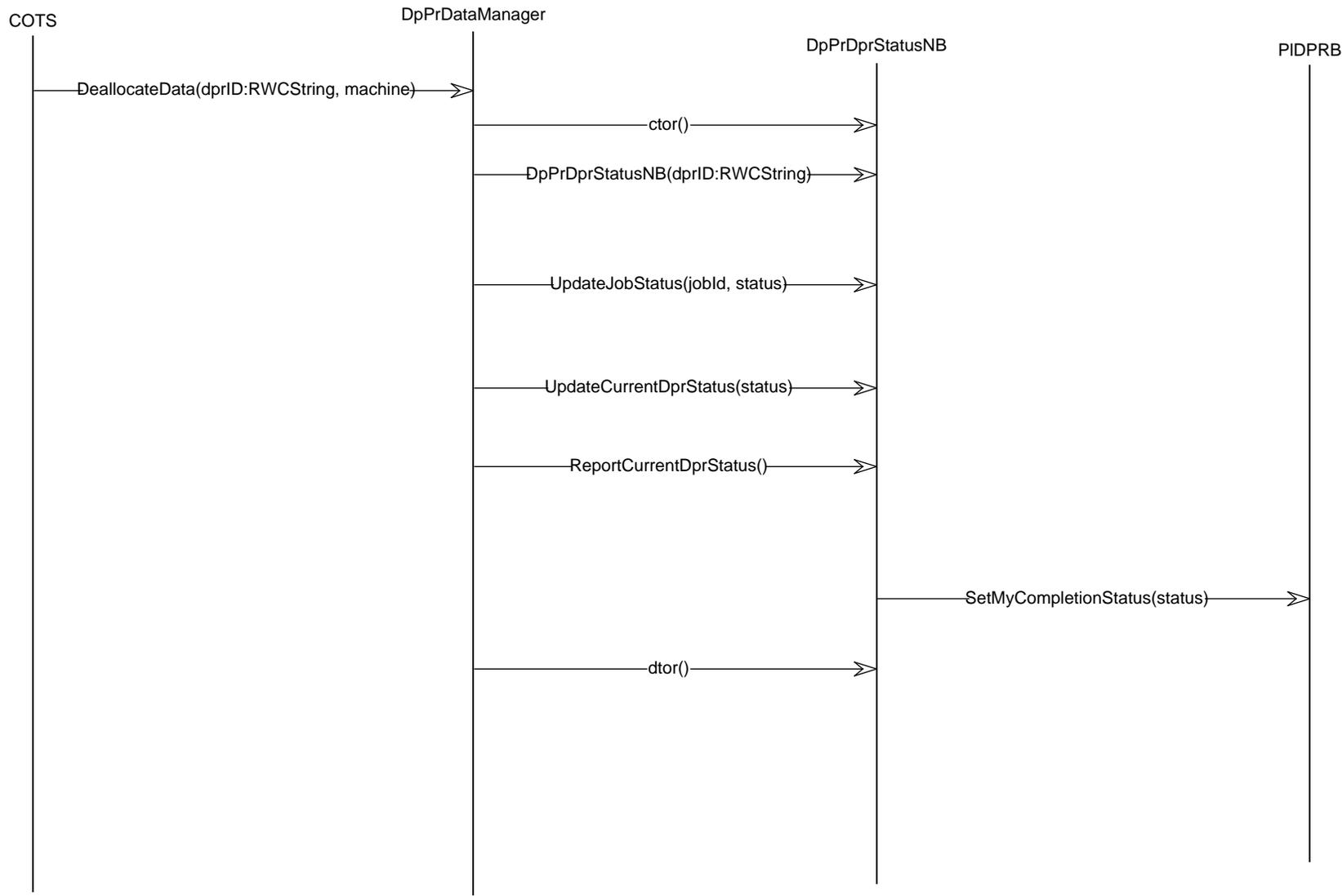


Figure 4.5-9. Report Job Status

4.5.3 Data Management Scenarios

These scenarios describe the activities associated with the Data Management component of the Processing CSCI. These scenarios describe the Processing CSCI activities which occur to support the following activities:

1. Staging and destaging of data from and to the Science Data Server CSCI.
2. Retaining of data on Science Production Hardware to support further production
3. Deletion of data not required to support further production.
4. Movement of data from one production resource to another to support further production.

These scenarios explain the interface which exists between the Processing CSCI and the Science Data Server CSCI. Data Staging and Destaging are services used to interface with the Science Data Server to coordinate the transfer of data. Data Staging defines the transfer of data from the Science Data Server CSCI to the Processing CSCI, and Data Destaging defines the transfer of data from the Processing CSCI to the Science Data Server CSCI. The protocol used to provide these services consists of the Processing CSCI requesting the transfer of data from the Science Data Server to Processing or from Processing to the Science Data Server. The Science Data Server is responsible for the movement of the data. Any data requiring transfer from the Science Data Server to Processing, or from Processing to the Science Data Server will be transferred using this approach. This data includes PGE scripts, algorithm executables, PGE status message files, PGE process control files, metadata, calibration data files, ancillary data products, or ECS Data Products. At this time, the Processing design is based on the assumption that all data staging occurs prior to the start of PGE execution. Also, PGE execution is not considered complete until destaging of the output data products occurs.

As a result of the initiation of data staging by Processing, the Science Data Server will initiate a file transfer operation to copy the data to the science processing hardware. As a result of the initiation of data destaging by Processing, the Science Data Server will initiate a file transfer operation to copy the data from the science processing hardware to the Data Server subsystem hardware.

Please note that the Processing CSCI uses the Science Data Server CSCI provided services to stage data from the Ingest subsystem as well as the Data Server Subsystem. Also, the Processing CSCI interfaces with Data Servers at different DAAC locations to stage and destage data.

4.5.3.1 Data Initialization

4.5.3.1.1 Abstract

The Processing CSCI has the responsibility to manage data that is currently residing on the science processing hardware resources and to retain data that is required to support the execution of multiple PGEs. To manage this data, the Processing CSCI must retain knowledge of the current location of data. The location may be at the Data Server or on some local science Processing hardware resource. To retain knowledge of the current location of data and other characteristics that are used to define a data object, the Processing CSCI has established a persistent class referred to as the DpPrDataMap. The persistence of this class is retained through the use of the PDPS Database. When this class is required to support a Processing CSCI operation, this class is created and information is extracted from the PDPS Database. More information on the PDPS Database

and the extraction of data is contained in the Planning Subsystem Detailed Design Specification. The information retained in this class is used during different Processing CSCI operations to determine if staging or destaging of a given data object is required.

The initialization of the class DpPrDataMap occurs when the CreateDPRJob operations of DpPrScheduler class is activated. Part of the CreateDPRJob operation will consist of creating the DpPrDataMap for a Data Processing Request.

4.5.3.1.3 Stimulus

When the Planning CSCI adds a Data Processing Request to AutoSys' daily job schedule, the DpPrScheduler will abstract data from the Data Processing Request which will be retained in the DpPrDataMap class. The persistence of this data will be managed through the PDPS Database.

4.5.3.1.4 Desired Response

The initialization of the DpPrDataMap class which contains information on the characteristics of data that a PGE associated with a Data Processing Request requires for execution.

4.5.3.1.5 Participating Classes From the Object Model

- DpPrScheduler
- DpPrDataManager
- DpPrDataMap
- PIDPRB
- PIDataGranule
- GIUR

4.5.3.1.8 Scenario Description

- a) When the CreateDPRJob operation is initiated on the DpPrScheduler, the following steps will occur:
 - 1) The DpPrScheduler will retrieve information as needed from the PDPS database to perform the following steps for each job:
 - (a) Create a DpPrDataMap used to retain information on the input data required to support the execution of the PGE associated with the Data Processing Request.
- b) END OF SCENARIO.

4.5.3.1.9 Event Trace

Figure 4.5-10 shows the DpPrDataMap Initialization.

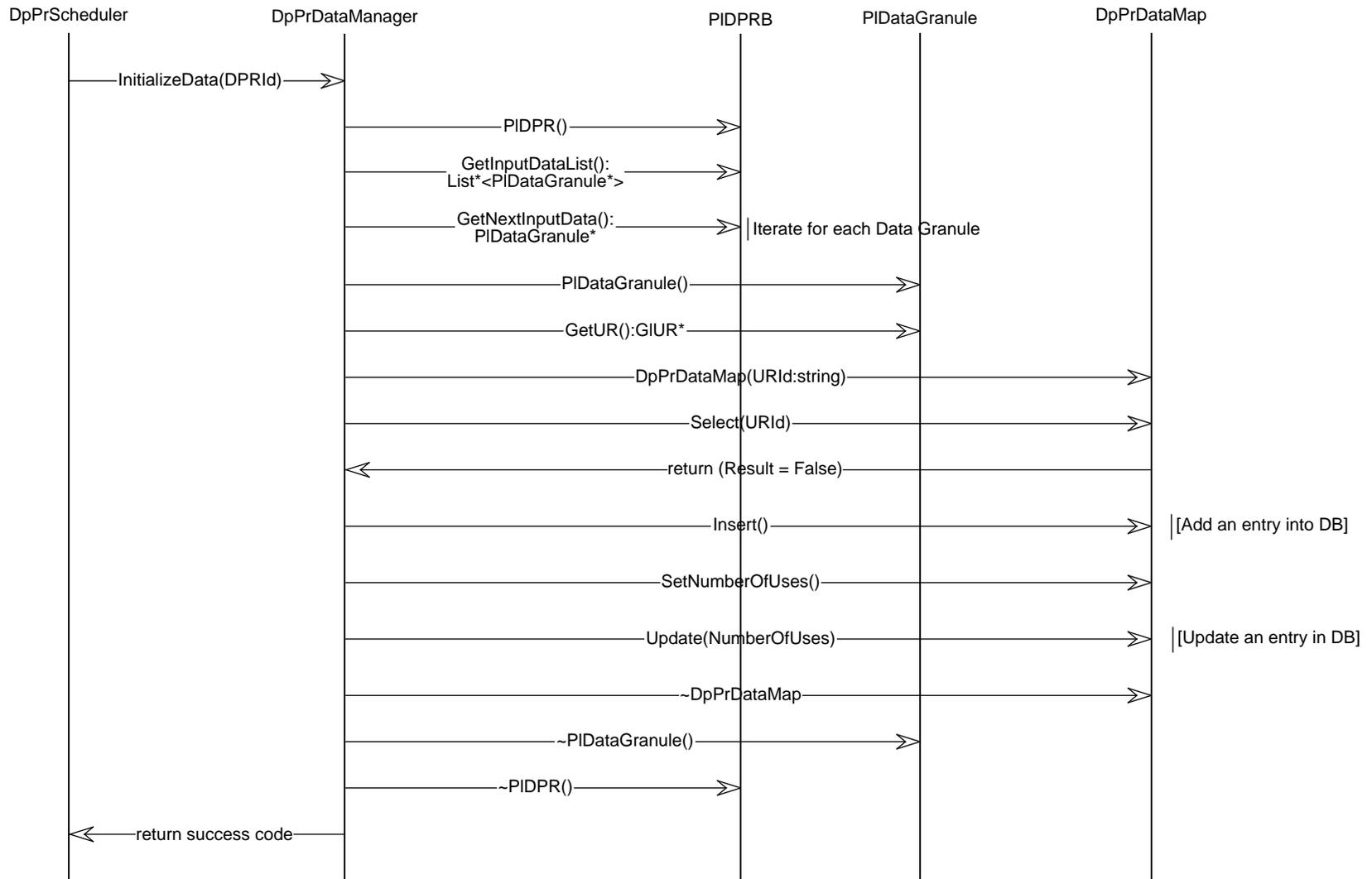


Figure 4.5-10. Data Initialization

4.5.3.2 Local Data Management

4.5.3.2.1 Abstract

The Processing CSCI has the responsibility to ensure that all input data are available and reside on the science processing hardware resources before the execution of PGEs. The Processing CSCI allocates sufficient resources to support output data that will be produced by a PGE even before ensuring that all input data are local. If our science Processing hardware resource can accommodate all output data, then it proceeds with determining the availability of input data. To manage this task, the Processing CSCI must determine if data is located locally on the platform that the PGE is using for execution, or on some other local science Processing hardware resource, or data is not available on our science Processing hardware resource at all. If the data resides on this platform, no further actions need to be done. If the location of data is on some other local science Processing hardware resource, then a location is allocated by Resource Management and the data will be copied from some other resource platform to this platform at the new location provided by Resource Management. This new location and other characteristics of this data are stored in PDPS Database. In the case that the data is not available on our science Processing hardware resource at all, the Resource Management allocates space for this data on this platform. The processing CSCI then builds a Data Server request including ACQUIRE command(s) that will be used to request the staging of data to a science processing resource. Also this new location provided by Resource Management and other characteristics of this data are stored in PDPS Database. When the start time set for the data staging (data localization) job equals the current time, AutoSys will start the data staging job and data localization occurs.

4.5.3.2.2 Stimulus

After the Planning CSCI releases a Data Processing Request to AutoSys' daily job schedule and the start time set for the predictive data staging (data localization) job equals to the current time, AutoSys will kick off a job which invokes MakeDataLocal of DpPrDataManager class.

4.5.3.2.3 Desired Response

This activity will result in the initiation of a persistent data structure used to retain knowledge about the location of data on the Science Processing Hardware resources.

4.5.3.2.4 Participating Classes From the Object Model

- COTS
- DpPrDataManager
- DpPrResourceManager
- PIDPRB
- DpPrDataMap
- PlDataGranule

4.5.3.2.5 Scenario Description

- a) When Autosys invokes MakeDataLocal of DpPrDataManager class, the following steps will occur:

- 1) The AutoSys will kick off a job that invokes MakeDataLocal operation of DpPrDataManager class to perform the following steps for each job:
 - (a) Iterate through the output data granule list, call DpPrResourceManager to allocate resource for each output data.
 - (b) If all output data granules are allocated successfully, iterate through the input data granule list, determine if data is available at science processing hardware resource or not.
 - (c) If data is resident on this local science processing hardware resource, the PDPS Database entry will be updated.
 - (d) If data is not resident on this local science processing hardware resource, but is located on some other local science processing hardware resource, then ask DpPrResourceManager to allocate space for this data on this local resource and copy data from nearby resource to this local resource at the new location. Go to PDPS Database and add an entry for this data with new location and other characteristic information.
 - (e) If data is not located anywhere on our local science processing hardware resource, then ask DpPrResourceManager to allocate space for this data on this local resource, and request Data Server to stage data to our local science processing hardware resource (see Data Staging section). Go to PDPS Database and add an entry for this data with new location, status is set to STAGING, and other characteristic information.
- b) END OF SCENARIO.

4.5.3.2.6 Event Traces

Figures 4.5-11 through 4.5-13 show the Local Data Management event traces.

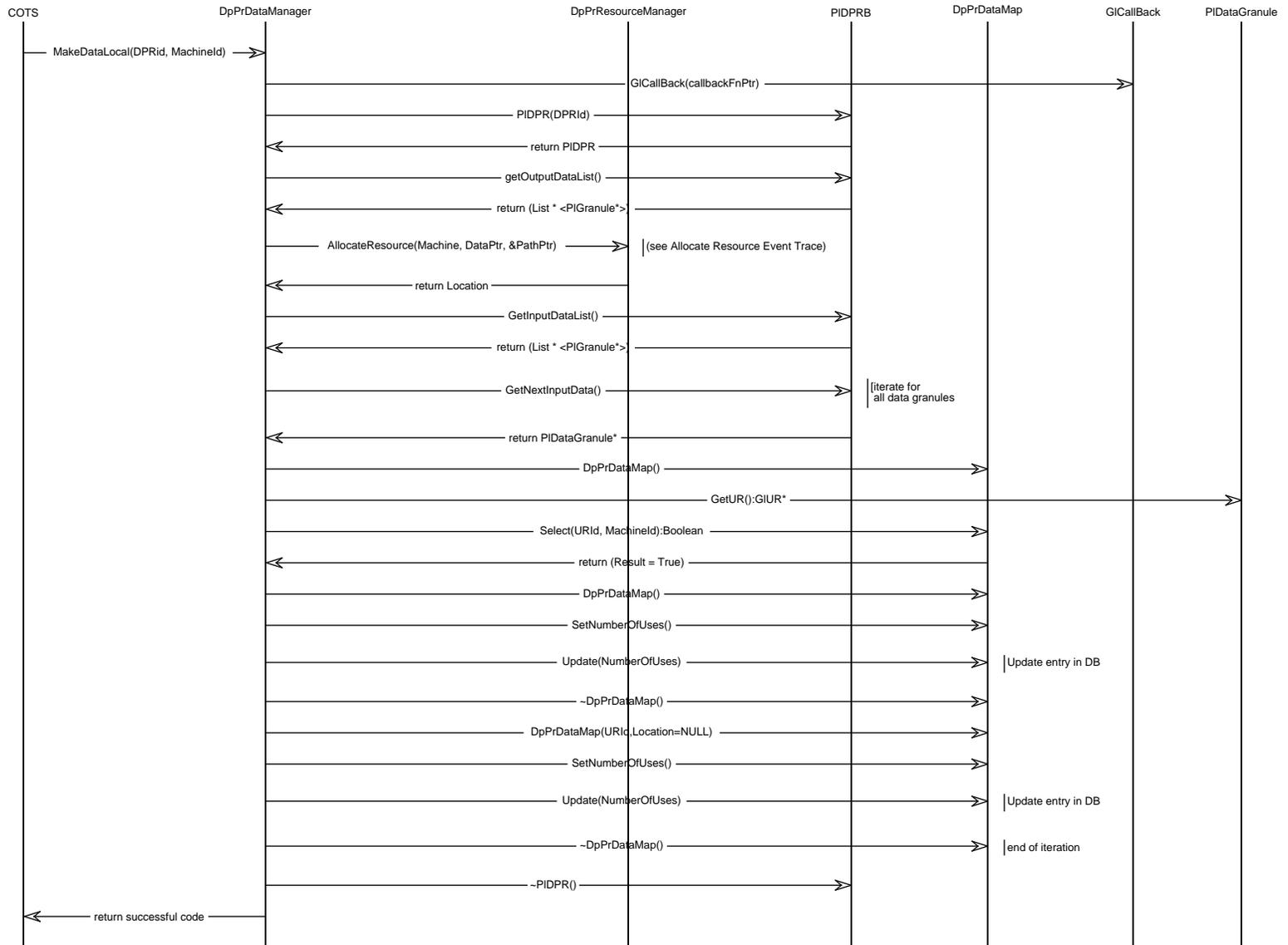
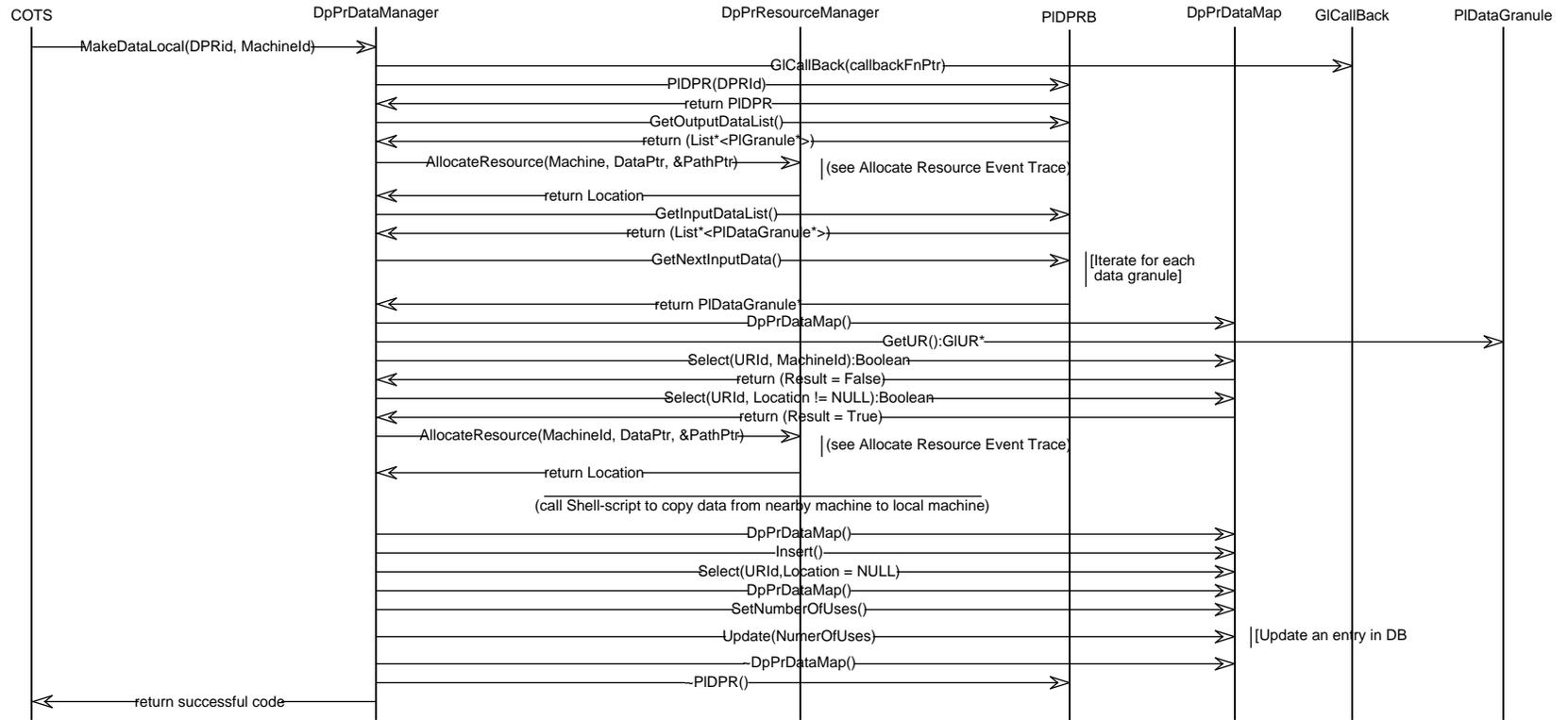


Figure 4.5-11. Local Data Management (Data Resides on Science Processing Resource)



[Add entry into DB table]

Figure 4.5-12. Local Data Management (Local Data Movement)

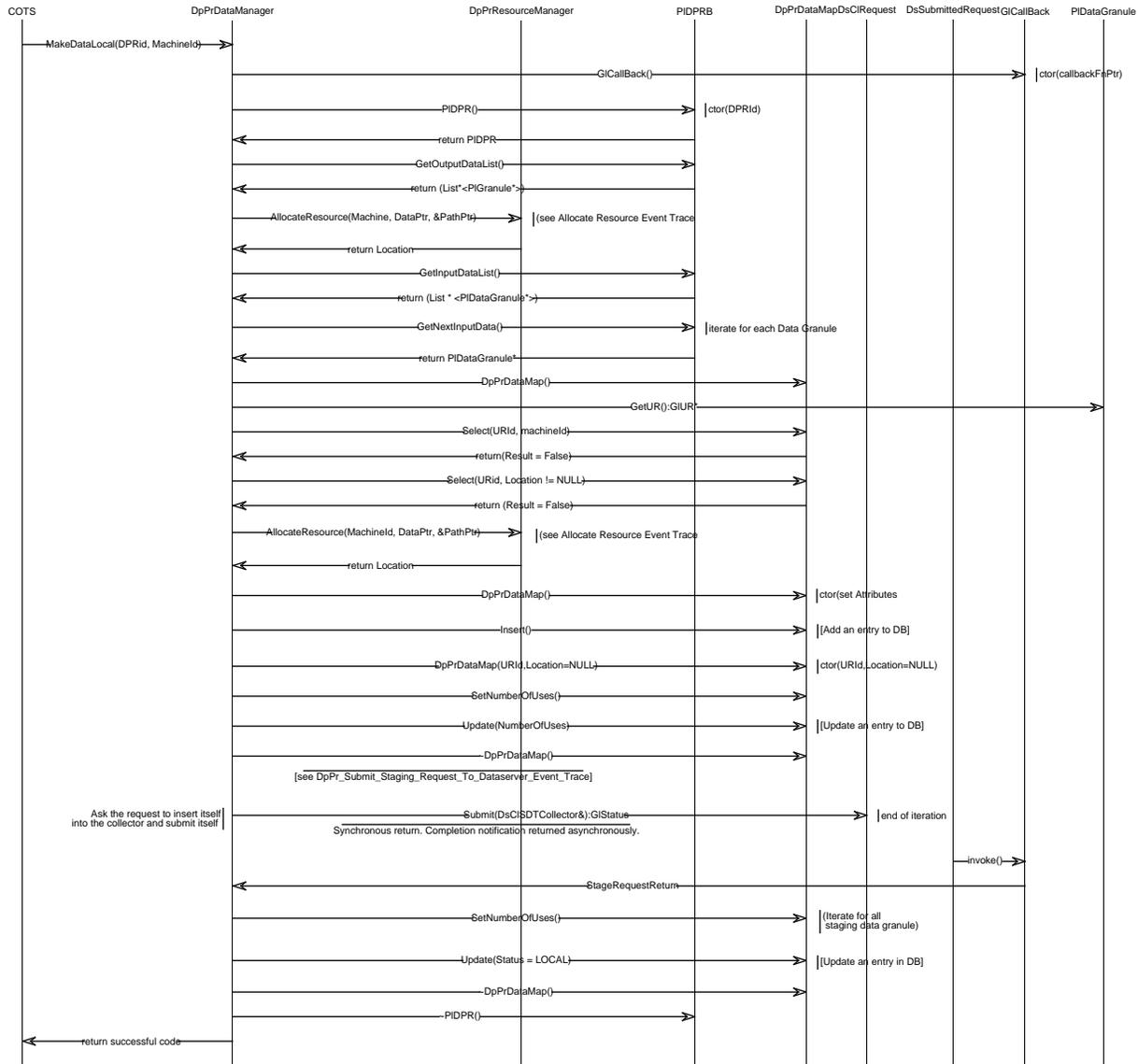


Figure 4.5-13. Local Data Management (Data Staging Required)

4.5.3.3 Data Staging

4.5.3.3.1 Abstract

Data Staging is an internally generated Data Processing process which must occur prior to PGE execution. Data Staging will be initiated when the COTS has initiated the Data Management Job associated with the AutoSys job box created from a Data Processing Request. After determining that there are sufficient resources required to support the execution of the PGE, the Data Management services will initiate the data staging process. The Data Management services will use Science Data Server CSCI provided public classes to request data staging. The Data Staging process is the series of steps followed to transfer data from the Data Server subsystem to the Data Processing subsystem. The staging of all data required by a PGE is completed prior to the execution of a PGE.

4.5.3.3.2 Stimulus

As the jobs which are part of the job box associated with a Data Processing Request are executed, one of the jobs which performs Data Management services will request data staging to be initiated upon determining that sufficient resources are available to support data staging and PGE execution.

4.5.3.3.3 Desired Response

The data staging process will be initiated by issuing a request to the Science Data Server CSCI. Part of this request will consist of an Acquire command which will be used to request the staging of data to a science processing resource. Also provided with this request is a Callback Mechanism which is provided by the Processing CSCI to the Science Data Server CSCI to return the success or failure status of the staging operations.

4.5.3.3.4 Participating Classes From the Object Model

- DpPrDataManager
- SDSRV
- PIDPRB
- PIDataGranule

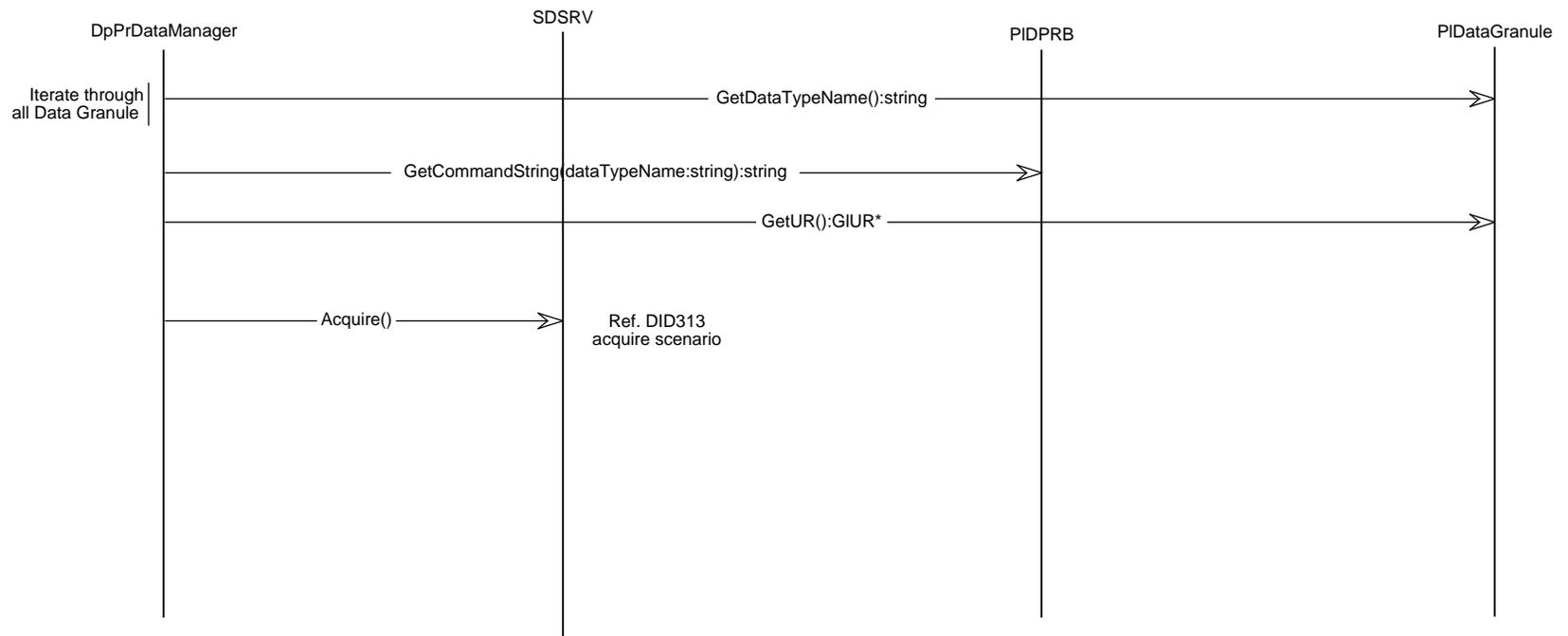
4.5.3.3.5 Scenario Description

- a. When AutoSys determines that a Data Management job can be initiated, i.e., all the job dependencies defined for this job have been fulfilled, the Data Management Job is executed and the following activities occur:
 1. Build a request to provide to the Science Data Server CSCI. This request will consist of a series of commands which will be the Acquire commands needed to stage all of the data required for the PGE associated with this Data Processing Request. The Acquire commands are actually obtained from the PDPS Database and added to the Science Data Server CSCI public class. After completing the building of the request, the request is submitted to the Science Data Server CSCI. The Data Management Job will wait until a notification is received back from the Science Data Server CSCI that this request was fulfilled successfully or unsuccessfully.

2. When the Science Data Server CSCI activates the Callback mechanism to inform the Processing CSCI of the success or failure of the request, the Data Management Job will map the success or failure of the data staging operation to a return status code and gracefully terminate. The return status code will be used by AutoSys to determine the success or failure of this job. If the return code indicates success, AutoSys will initiate the next dependent job, i.e., the PGE Preparation Job. If the return code indicates failure, AutoSys will take appropriate error recovery activities, such as alerting this operations staff. More information on the failure of data staging is contained in the Section 4.5.4, Failure of Data Staging scenario.
 3. Logging of all information related to these events will occur through AutoSys and MSS provided services used by the Data Management component of the Processing CSCI.
- b. End of Scenario.

4.5.3.3.6 Event Trace

Figure 4.5-14 shows the data staging event trace.



4-254

Figure 4.5-14. Data Management (Data Resides on Data Server)

305-CD-027-002

4.5.3.4 Failure of Data Staging

4.5.3.4.1 Abstract

The failure of data staging can occur for numerous reasons which are internal or external to the domain of the Processing CSCI. In all cases, the Science Data Server CSCI will inform the Processing CSCI of the failure of data staging. If the cause of the problem is within the science processing resources, appropriate error recovery actions will be initiated which may lead to the re-initiation of staging, informing Operations and awaiting manual intervention, or taking appropriate termination activities.

4.5.3.4.2 Stimulus

The stimulus which triggers a Data Staging Failure occurs during the interaction of the Data Server with the staging resources and is therefore an external event. This scenario will address the outcome on Processing due to such an event. Using a callback mechanism which is provided by the Processing CSCI, the Science Data Server CSCI will inform the Processing CSCI of the failure of data staging. This failure information will be transferred into a status return code which is used to inform AutoSys of the failure of the Data Management job.

4.5.3.4.3 Desired Response

When alerted to the failure of the Data Management Job through the status return code mechanism, AutoSys will activate appropriate error recovery actions. Initially, this amounts to alerting the Operations staff, logging information about the failure to AutoSys job log as well as the system log retained by MSS.

4.5.3.4.4 Participating Classes from the Object Model

- GICallBack
- DpPrDataManager
- COTS

4.5.3.4.5 Scenario Description

- a. When the Science Data Server CSCI activates the Callback mechanism to inform the Processing CSCI of the success or failure of the request, the Data Management Job will map the success or failure of the data staging operation to a return status code and gracefully terminate. The return status code will be used by AutoSys to determine the success or failure of this job. If the status return code indicates failure, AutoSys will take appropriate error recovery activities, such as the following:
 1. Activate the alert mechanisms to inform the operations staff.
 2. Check the local resources for problems which may have caused the staging failure.
 3. Log fault information to MSS.
 4. Delete the input data that has been staged thus far, unless subsequent, near-term, processing requires it.

5. Update the operational mode of the failed resource to "Off-Line" so that these resources do not get allocated for subsequent processing until the resource problem is corrected.
 6. Deallocate the remaining resources initially allocated for this run of the PGE.
- b. End of Scenario.

4.5.3.4.6 Event Trace

Figure 4.5-15 shows the failure of data staging event trace.

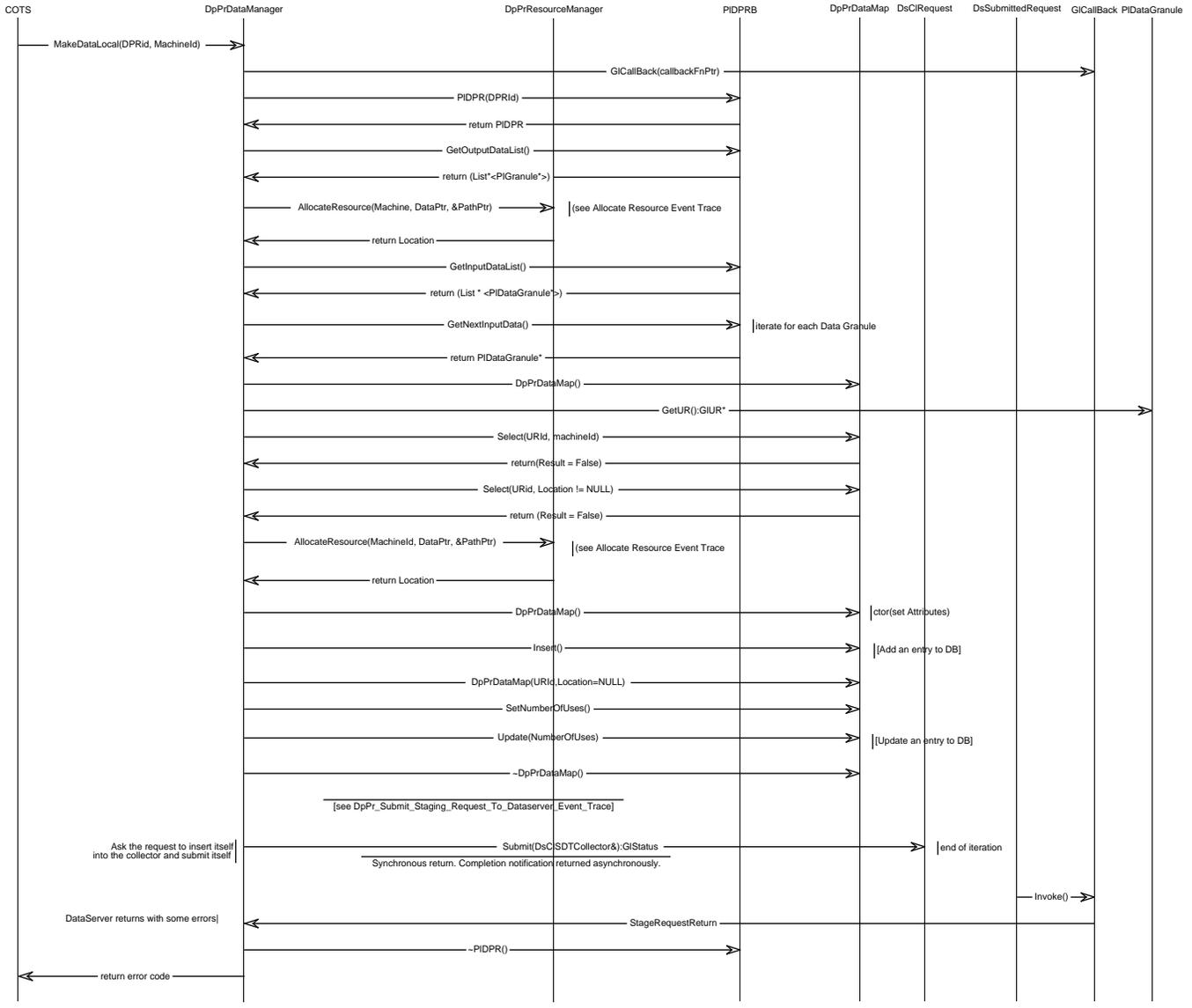


Figure 4.5-15. Failure of Data Staging

4.5.3.5 Data Destaging

4.5.3.5.1 Abstract

Data Destaging is an internally generated Data Processing process which must occur after completion of PGE execution. Data Destaging will be initiated when the COTS has initiated the PGE Post-Processing Job associated with the AutoSys job box created from a Data Processing Request. The Data Management services will use Science Data Server CSCI provided public classes to request data destaging. The Data Destaging process is the series of steps followed to transfer data from the Processing CSCI to the Science Data Server CSCI. The destaging of all output data produced by a PGE is completed after the execution of a PGE.

4.5.3.5.2 Stimulus

As the jobs which are part of the job box associated with a Data Processing Request are executed, the PGE Post-Processing job will request data destaging to be initiated.

4.5.3.5.3 Desired Response

The data destaging process will be initiated by issuing a request to the Science Data Server CSCI. Part of this request will consist of an Insert command which will be used to request the destaging of data to a science processing resource. Also provided with this request is a Callback Mechanism which is provided by the Processing CSCI to the Science Data Server CSCI to return the success or failure status of the destaging operations.

4.5.3.5.4 Participating Classes From the Object Model

- DpPrDataManager
- DpPrResourceManager
- DsCIESDTReferenceCollector
- DsCIRequest
- DsCICommand
- PIDPRB
- PIDataGranule
- GICallBack

4.5.3.5.5 Scenario Description

- a. When AutoSys determines that a PGE Post-Processing job can be initiated, i.e., all the job dependencies defined for this job have been fulfilled, the PGE Post-Processing Job is executed and the following activities occur:
 1. Build a request to provide to the Science Data Server CSCI. This request will consist of a series of commands which will be the Insert commands needed to destage all of the data required for the PGE associated with this Data Processing Request. The Insert commands are actually obtained from the PDPS Database and added to the Science Data Server CSCI public class. After completing the building of the request, the request is submitted to the Science Data Server CSCI. The Data Management Job will wait until

- a notification is received back from the Science Data Server CSCI that this request was fulfilled successfully or unsuccessfully.
2. When the Science Data Server CSCI activates the Callback mechanism to inform the Processing CSCI of the success or failure of the request, the PGE Post-Processing Job will map the success or failure of the data destaging operation to a return status code and gracefully terminate. The return status code will be used by AutoSys to determine the success or failure of this job. If the return code indicates success, AutoSys will conclude the processing for this job and move on to initiate the next job awaiting execution. If the return code indicates failure, AutoSys will take appropriate error recovery activities, such as alerting this operations staff. More information on the failure of data staging contained in the Section 4.5.6, Failure of Data Destaging scenario, or Section 4.5.7, Failure of Data Server Communication scenario.
 3. Logging of all information related to these events will occur through AutoSys and MSS provided services used by the Data Management component of the Processing CSCI.
- b. End of Scenario.

4.5.3.5.6 Event Trace

Figure 4.5-16 shows the deallocate data event trace. Figure 4.5-17 shows the data destaging event trace.

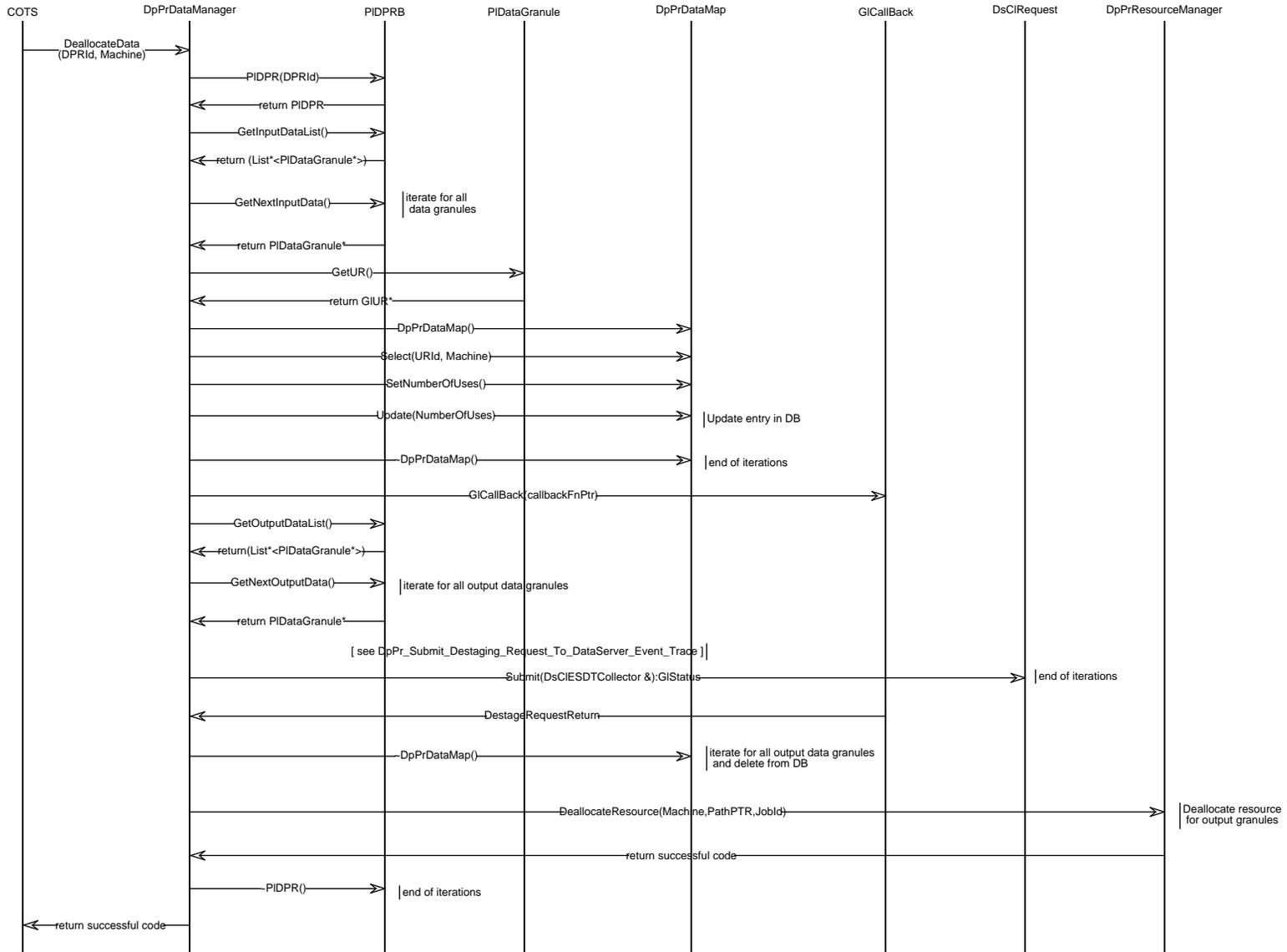


Figure 4.5-16. Deallocate Data

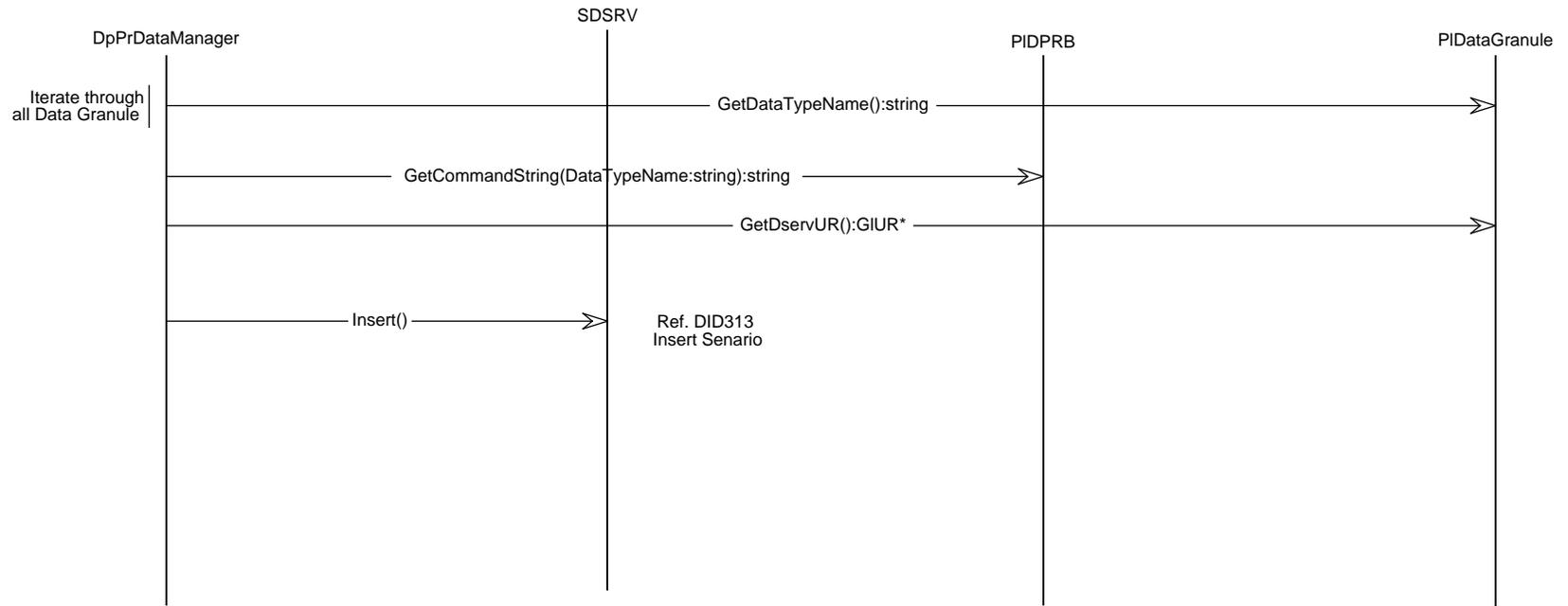


Figure 4.5-17. Data Destaging

4.5.3.6 Failure of Destaging

4.5.3.6.1 Abstract

The failure of data staging or destaging can occur for numerous reasons which are internal or external to the domain of the Processing CSCI. In all cases, the Science Data Server CSCI will inform the Processing CSCI of the failure situation. If the cause of the problem is within the science processing resources, appropriate error recovery actions will be initiated which may lead to the re-initiation of staging or destaging, informing Operations and awaiting manual intervention, or taking appropriate termination activities.

4.5.3.6.2 Stimulus

The stimulus which triggers a data staging or destaging failure occurs during the interaction of the Science Data Server CSCI with the staging resources and is therefore an external event. This scenario will address the outcome on the Processing CSCI due to such an event. Using a callback mechanism which is provided by the Processing CSCI, the Science Data Server CSCI will inform the Processing CSCI of the failure of data staging or destaging. This failure information will be transferred into a status return code which is used to inform AutoSys of the failure of the PGE or Post-Processing job.

4.5.3.6.3 Desired Response

When alerted to the failure of the PGE Post-Processing Job through the status return code mechanism, AutoSys will activate appropriate error recovery actions. Initially, this amounts to alerting the Operations staff, logging information about the failure to AutoSys job log as well as the system log retained by MSS.

On failing to destage all of the output data produced by the execution of the current PGE, the Operations should attempt to determine if the problem lies with the local resources and if so, should take steps to identify the failed resources in order to prevent similar problems during subsequent processing. In any event, deallocation of resources should not occur for this process until the data are recovered and destaging has been performed.

4.5.3.6.4 Participating Classes from the Object Model

- DpPrDataManager
- GICallBack
- COTS

4.5.3.6.5 Scenario Description

- a. When the Science Data Server CSCI activates the Callback mechanism to inform the Processing CSCI of the success or failure of the request, the Data Management Job will map the success or failure of the data destaging operation to a return status code and gracefully terminate. The return status code will be used by AutoSys to determine the success or failure of this job. If the status return code indicates failure, AutoSys will take appropriate error recovery activities, such as the following:
 1. Activate the alert mechanisms to inform the operations staff.

2. Check the local resources for problems which may have caused the destaging failure.
 3. Log fault information to MSS.
 4. Update the operational mode of the failed resource to "Off-Line" so that these resources do not get allocated for subsequent processing until the resource problem is corrected and the generated outputs of the PGE are destaged successfully.
- b. End of Scenario.

4.5.3.6.6 Event Trace

Figure 4.5-18 shows the failure of a data destaging event trace.

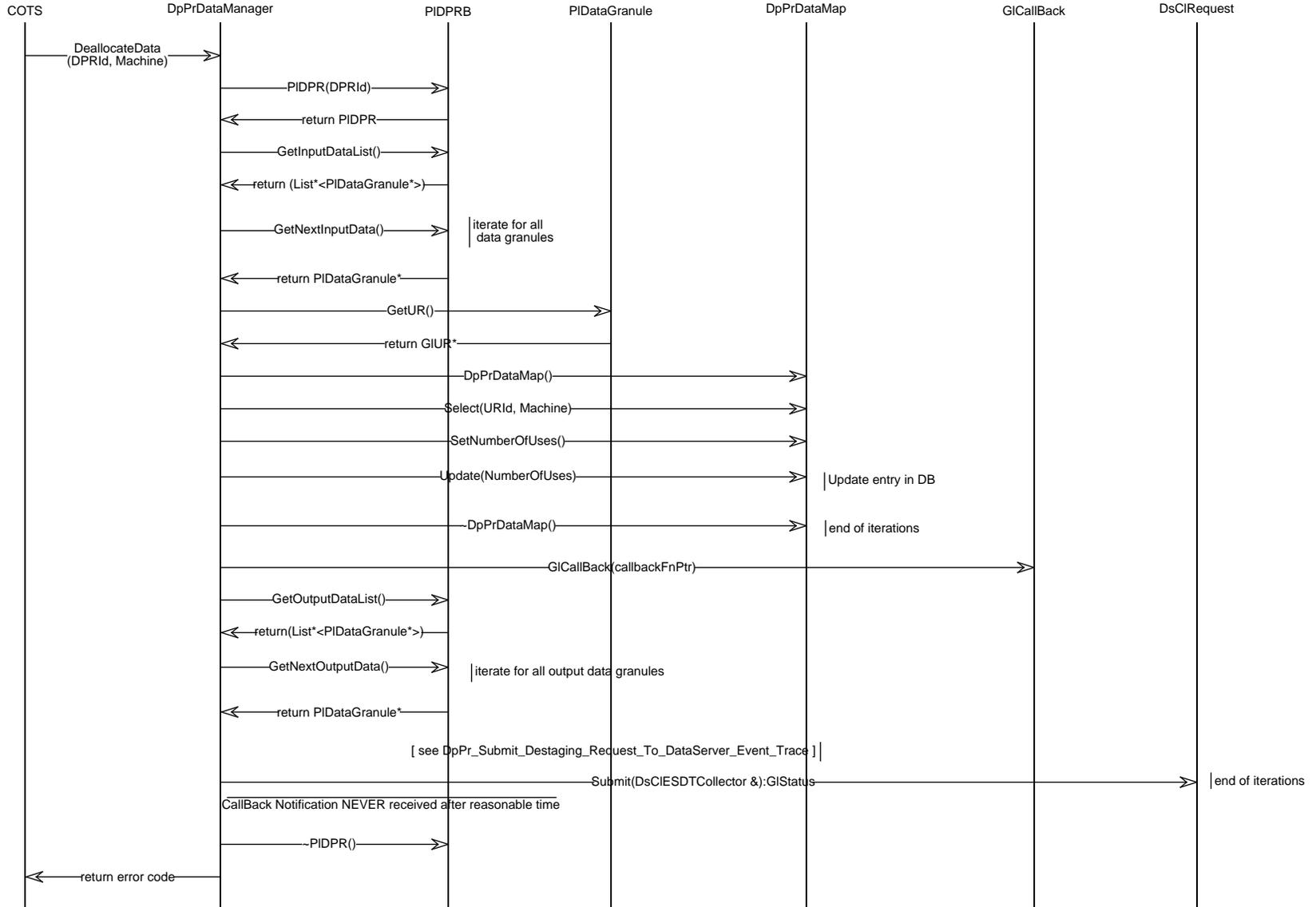


Figure 4.5-18. Failure of Destaging

4.5.3.7 Failure of Data Server Communication

4.5.3.7.1 Abstract

The failure of Data Server Communication during staging or destaging of data is handled similarly to the handling of the failure of staging. All appropriate error recovery actions will be tried including initiating the data staging or destaging again, suspending further processing of the PGE job definition (i.e., Data Processing Request) and awaiting Operations intervention, or possibly terminating further processing of the PGE job definition and submitting a communications fault to MSS.

4.5.3.7.2 Stimulus

The stimulus which triggers a Data Server Communication Failure occurs external to the Processing CSCI. This scenario will address the outcome on Processing when such an event occurs during Process Control File (PCF) destaging.

4.5.3.7.3 Desired Response

A single Process Control File (PCF) is created by the PGE Execution Management services and may undergo minor modifications during the execution of the current PGE. On failing to destage the Process Control File (PCF) due to a failure of the communication channels between Processing and Data Server, the Execution Management services should be able to determine if the problem lies with the local resources. If so, steps should be taken to identify the failed resources in order to prevent similar problems during subsequent processing. However, since the PCF may still remain on the failed resource, if deallocation cannot be achieved after several attempts, the resource is taken off-line to allow for operator intervention. In any event, deallocation of resources should not occur for this process until the PCF is recovered and destaging has completed successfully.

4.5.3.7.4 Participating Classes from the Object Model

- COTS
- DpPrPge
- DpPrExecutionManagement
- DpPrResourceManagement
- PIDataGranule
- PIDPRB
- DsCIESDTReferenceCollector
- DsClRequest
- DsClCommand
- MsMgCallbacks
- MsManager
- MsEvent

4.5.3.7.5 Scenario Description

- a. After the COTS Scheduler has activated the destaging of the Process Control File (PCF) to support post-processing of a PGE, the following activities may occur:
 1. With the data successfully copied to the Data Server, deallocation of Processing resources may proceed.
 2. The Execution Management service issues a request to destage the Process Control File (PCF) to support the SCF analysis of a failed, or suspect processing attempt.
 3. If an event, which is external to the Processing CSCI, causes the communication with the Data Server to become disrupted, the Data Server will not be able to fulfill the destaging request. The end result is that the Execution Management "callback" routine will not be activated by the Data Server and Execution Management, therefore, will not receive a complete notification.
 4. If a problem occurs during the destaging effort, the Execution Management destaging services will time-out and conclude that the destage operation has failed. Resource Management services will then be called upon to perform a check on the local resources for problems which may have caused the destaging failure.
 5. Deallocate the remaining processing resources initially allocated for this run of the PGE, leaving allocated those resources where output data may still reside.
 6. If it is discovered that local resources were the cause of the destaging failure, a resource fault will be logged with MSS, otherwise a communications fault will be logged.
 7. Update the COTS Scheduler interface to indicate a failure during Post-processing.
- b. End of Scenario.

4.5.3.7.6 Event Trace

Figure 4.5-19 shows the failure of data server communication event trace.

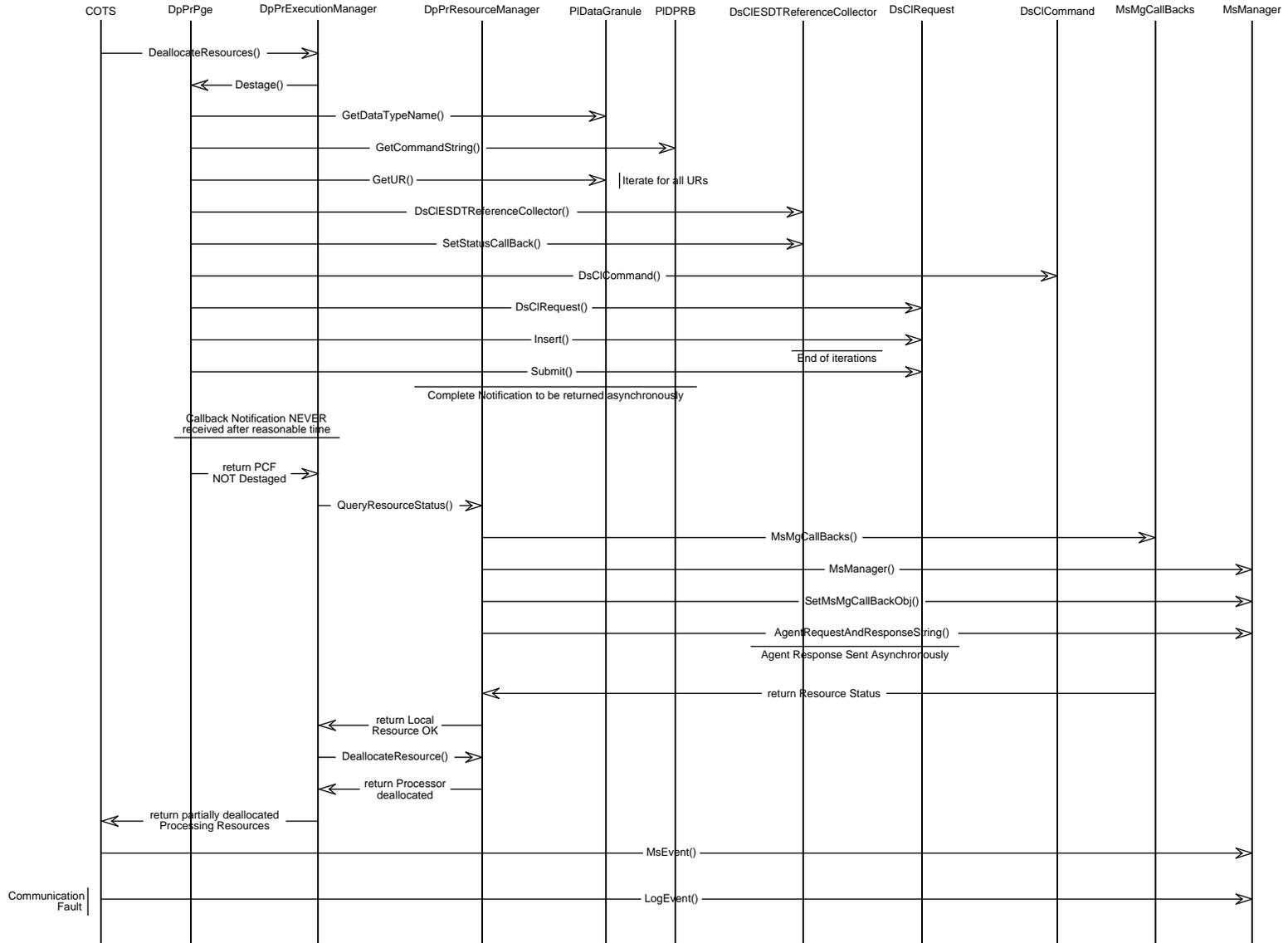


Figure 4.5-19. Failure of Data Server Communication

4.5.3.8 Predictive staging / make data local job scenario

4.5.3.8.1 Abstract

A DPR/PGE may need a huge amount of data as inputs, which may take hours to be staged. To offset the difference between the start time of a PGE and the end time of the PGE input data staging, a predictive staging mechanism is used to stage part or all of the inputs needed by the PGE from a data server to the local data staging disk before the PGE is ready to run.

4.5.3.8.2 Stimulus

Autosys will start the predictive staging job at the start time specified during the creation of the job.

4.5.3.8.3 Desired Response

The predictive staging job will be executed and input data for the PGE which are available from the data server and not available in the local disks will be staged to the local staging disk.

4.5.3.8.4 Participating Classes from the Object Model

- COTS
- DpPrDataManager
- DpPrResourceManager
- PIDPRB
- DpPrDataMap
- DsCIRequest
- DsSubmittedRequest
- PIDataGranule
- PIPerformance

4.5.3.8.5 Scenario Description

When the predictive staging job is executed, the MakeDataLocal operation in DpPrDataManager is called and the following step will happen.

- a. Construct a copy of the PIDPRB object for the actual DPR job in order to get its input data list and data granules from the PDPS database.
- b. Loop through the input data granules list
 - If a granule are available in the data server and not available in the local disk, stage it to the local disk and mark this input data granule in the submitted input data list from the PIDPRB. Update the DpPrDataMap to indicate the local availability of this data granule.
- c. Update the predictiveStagingTime field of the PIPerformance object using the running time used by this predictive staging job.

4.5.3.8.6 Event Trace

Figure 4.5-20 shows the predictive data staging event trace.

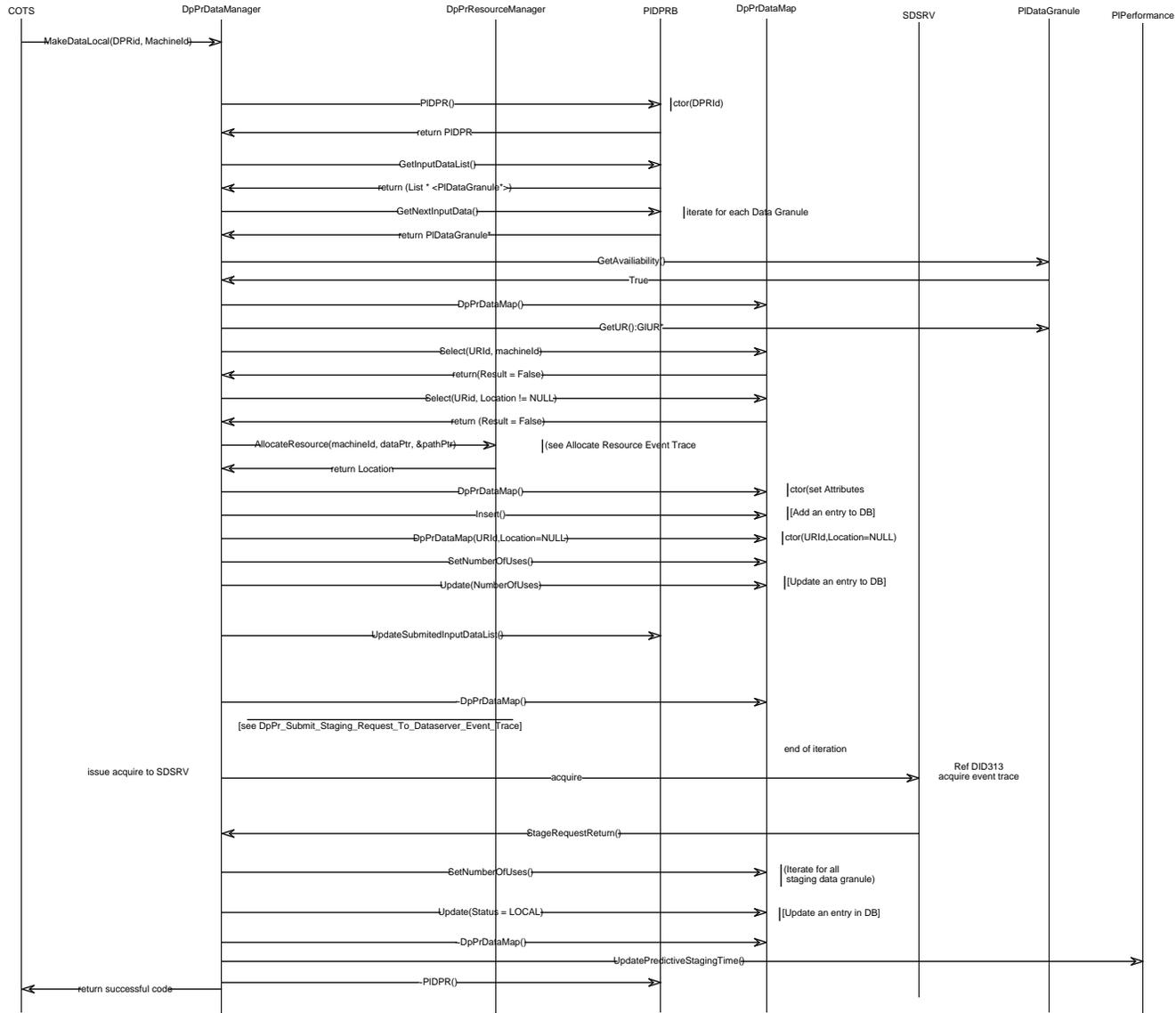


Figure 4.5-20. Predictive Data Staging

4.5.4 Execution Management Scenarios

Execution Management services are used to initiate and monitor the execution of a PGE. All support for PGE execution, i.e., SDP Toolkit interfaces, production history generation, resource monitoring, etc., will be provided by these services. The execution of a PGE is performed on Data Processing subsystem's resources and is initiated by the Processing CSCI.

The execution of a PGE is initiated when the following conditions are met:

- a. Data Processing subsystem resources are available to support the successful execution of the PGE.
- b. The priority and resource information assigned to the Data Processing Request has positioned the Data Processing Request, within the COTS Scheduler, as the next request to execute.

Conceptually, a PGE is defined as any processing job that requires Data Processing subsystem resources. Therefore, a PGE can define different types of processing; science software, quality assurance, or science pre-processing. If there are standard types of processing jobs that are performed periodically, these jobs are planned and submitted to Processing from Planning. Of course, there may be exceptions, such as operations staff intervention in processing of a PGE, where this is not followed. Most PGEs will be defined as science software PGEs.

The execution of a science software PGE will result in the generation of data products. These data products are of two types:

- a. Intermediate—This data is used to support the execution of other PGEs. Even though this data may only be required for a finite period of time, to facilitate the generation of relatively long term products, all such data files will be archived at the Data Server.
- b. Final Data Product—This data is defined as Level 1, Level 2, Level 3, or Level 4 data products. These products can be defined as Intermediate data to support the production of other products. For example, a Level 1 product is used to create a Level 2 product. The Level 2 product is a Final Data Product, but could also be used to create a Level 3 product.

Before initiating the execution of the PGE, the Processing CSCI provides information to the PGE about the location and names of input data and the destination and names of output data. Currently, this information is provided through a data mapping mechanism known as a Process Control File (PCF). This file contains information linking the logical representation of data to its physical counterpart, i.e., logical unit numbers to file paths. This data mapping represents the only interface between the Processing CSCI and the SDP Toolkit API. As such, this interface will be used to provide metadata to the PGE, through the SDP Toolkit, to be associated with the data products.

During execution, the Processing CSCI performs a monitoring role. Periodically, the Processing CSCI collects information on executing PGE(s). This information includes execution errors and warnings generated from the science software, as well as certain runtime performance statistics. Also, the Processing CSCI monitors the resources being used by the PGE. By comparing current resource usage data with the nominal resource usage data prepared by the Algorithm Integration and Test Team, the Processing CSCI can determine if a resource fault exists, or that a PGE fault has occurred. The Processing CSCI may terminate a PGE if such runtime information indicates that the PGE is not performing within the nominal bounds defined at AI&T, or if resource faults are occurring. To support the monitoring of the PGE, the Processing CSCI will use services provided by MSS.

Currently, no unique services have been identified to provide support for the PGEs which are not science software.

Figure 4.5-21 shows a state transition diagram for the execution of a PGE. This diagram should be reviewed while reading the scenarios and event traces.

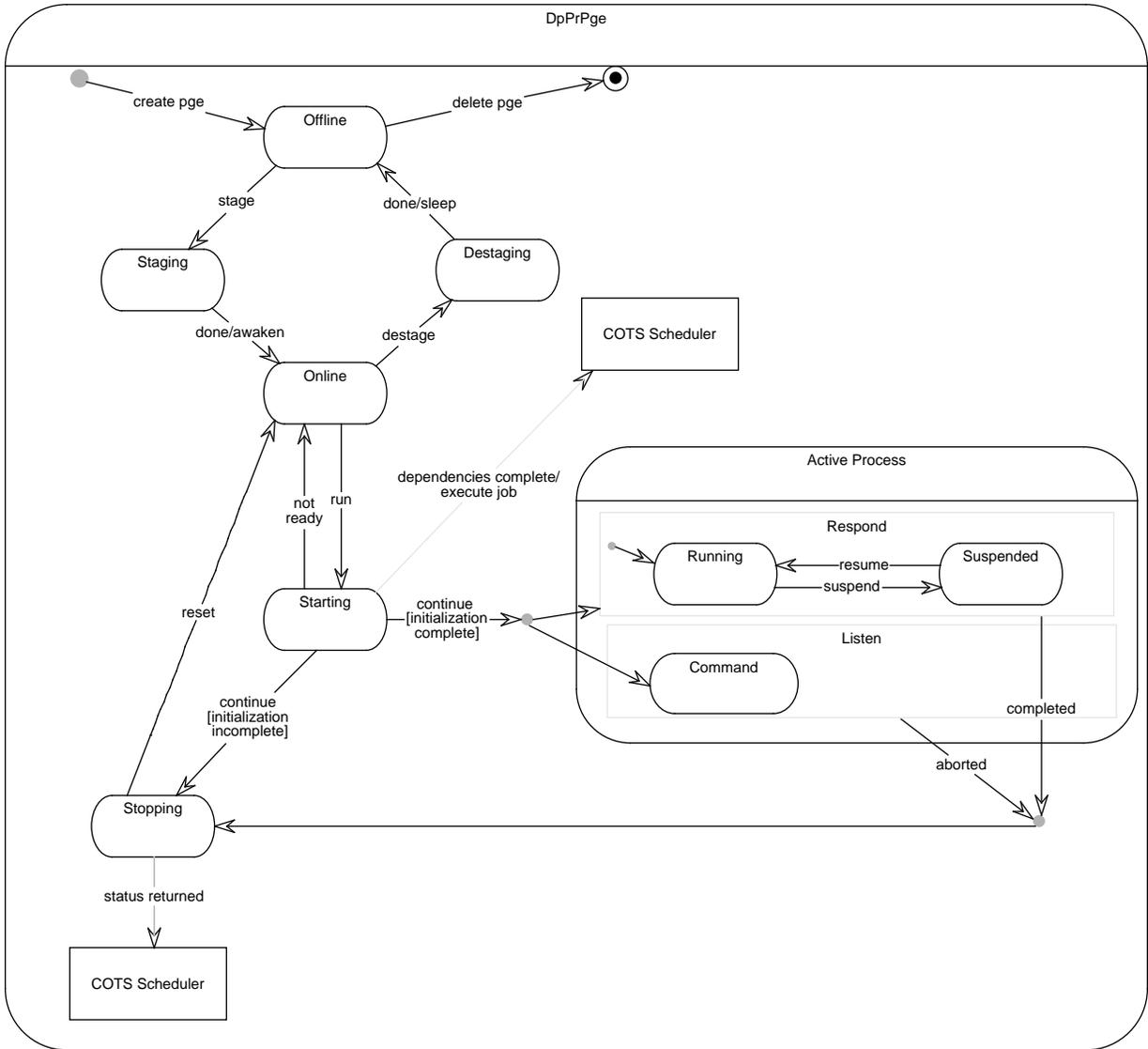


Figure 4.5-21. Execution of PGE State Transition Diagram

The scenarios provided for Execution Management are defined as the following

- a. Initiate Execution—This scenario provides information on the required activities for initiation of the execution of a PGE. See scenario "Execution Management: Initiate PGE Execution" for more information.
- b. Monitor Execution—These scenarios provides information on the required activities to monitor PGE execution. See scenarios "Execution Management: Monitor Resource", "Execution Management: Monitor Performance," and "Execution Management: Monitor Status Return" for more information.
- c. Execution post-processing—This scenario provides information on the required activities for PGE execution post-processing. See scenario "Execution Management: PGE Execution Post-Processing" for more information.
- d. Failure of Execution—This scenario provides information on the required activities following failure of PGE execution. See scenario "Execution Management: Failure of PGE Execution" for more information.
- e. Failure of Data Server Communications—This scenario provides information on the required activities following failure of communications during Data Server contact. See scenario "Execution Management: Failure of Data Server Communications" for more information.
- f. Failure of Processing Resource—This scenario provides information on the required activities following failure of a processing resource. See scenario "Execution Management: Failure of Processing Resource" for more information.

4.5.4.1 Initiate Execution

4.5.4.1.1 Abstract

When it is determined by the COTS Scheduler that the processing of a PGE can proceed, PGE preparation activities can begin. These activities consist of resource allocation and SDP Toolkit initialization.

Currently, the SDP Toolkit requires the creation of a Process Control File (or PCF), which acts as the interface between the science algorithm and the Processing CSCI. The data mappings contained within the PCF contain all of the information needed for the SDP Toolkit to access the data requested by the science algorithm. Since only one Process Control File is associated with a given PGE, it must be created and populated with new mapping information for each unique run of the PGE.

In addition to the input data that is staged for the PGE, SDP Toolkit and algorithm specific Status Message Files (SMFs) must also be staged to provide for the error and status reporting needs of the PGE. However, before any data item can be staged, the necessary disk resources must first be allocated for the current PGE. This occurs through an allocation process which is managed by the Resource Management services.

4.5.4.1.2 Stimulus

The stimulus to initiate PGE execution is received as a request from the COTS scheduler package. The events which occur just prior to the start of PGE execution signal the completion of resource allocation, data staging and SDP Toolkit initialization for this task.

4.5.4.1.3 Desired Response

The execution of a PGE job will be triggered by the COTS scheduling package.

4.5.4.1.4 Participating Classes from the Object Model

- COTS
- DpPrExecutable
- DpPrPcf
- DpPrPge
- DpPrExecutionManager
- DpPrResourceManager
- PIDataGranule
- PIDPRB
- DsCIESDTReferenceCollector
- DsCIRequest
- DsCICommand

4.5.4.1.5 Scenario Description

- a. When the COTS scheduler has requested the initialization of a PGE, the following activities are done:
 1. Perform the resource preparation activities required to support the staging of data and PGE, as well as the processing of the PGE (e.g., allocate disk space, CPU(s)).
 2. If not already resident on the local host, stage the actual PGE executables and associated Status Message Files (SMFs) which were delivered along with the science algorithm.
 3. Construct the Process Control File, for this instance of the PGE, to create the mapping of logical identifiers to physical file and runtime parameter values.
 4. Update the Process Control Table with "live" information to complete the mapping of logical identifiers to physical file and parameter values.
 5. Send Event to the COTS scheduler signifying the completion of the initialization phase.
 6. Log the success or failure to initialize the execution phase of the PGE execution. (i.e., job status is "Success," or "Failure").
 7. With all job dependencies satisfied, the COTS scheduler issues a start processing request to begin the actual PGE job execution.
- b. End of "Execution Management (Initiate Execution)."

4.5.4.1.6 Event Trace

Figure 4.5-22 shows the initiate execution event trace diagram.

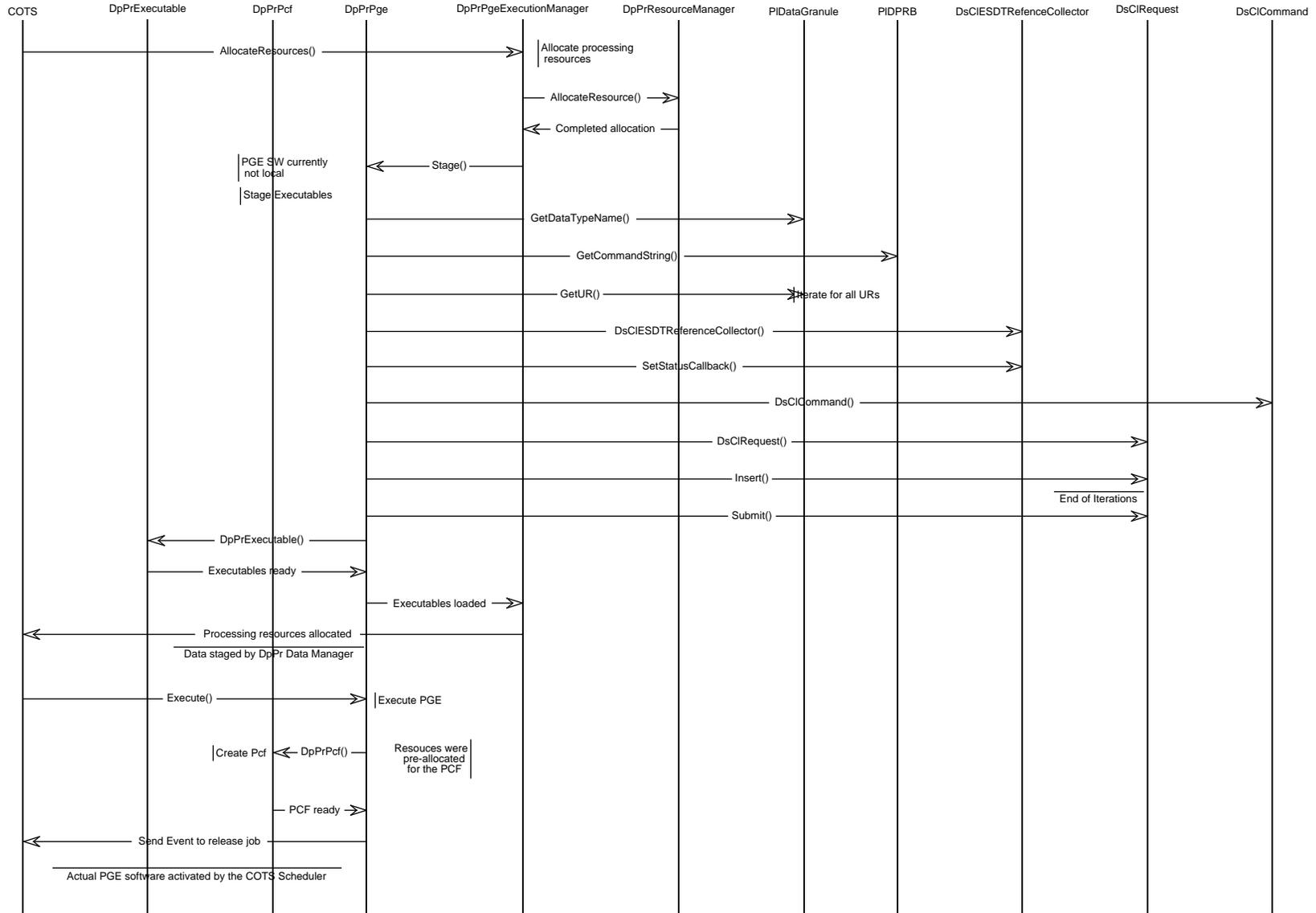


Figure 4.5-22. Initiate Execution

4.5.4.2 Monitor Execution

4.5.4.2.1 Abstract

Execution Management services will monitor the execution of a PGE and the resources that the PGE is using during execution. To monitor the execution of the PGE at runtime, the Execution Management services will obtain performance status information through a MSS defined method. To obtain the termination status of a PGE, the final state of processing will be returned to the COTS Scheduler upon completion of the PGE's execution.

The definition of the status condition codes used by the PGEs will be determined through ECS interaction with Instrument Team algorithm developers. Once a complete set of status condition codes is defined and distributed to Instrument Team sites, all science algorithm developers will be required to return one of these condition codes at the terminus of their processing. This will further require science algorithm developers to handle error conditions gracefully to ensure that this critical section of code always gets invoked.

Resource monitoring will consist of the monitoring of the usage of resources, e.g., CPU, disk space, and memory by the PGE. During AI&T, the PGE resource usage data will be established. These data are likely to include averages and maximums of the CPU, memory, and disk space that a PGE uses during execution. This information will be updated as the PGE is used in the production environment. During execution, resource monitoring will alert operations if the PGE has reached some percentage over the average, or maximum allowed. Resource monitoring also will continuously check on the health of the resources being used. Resources will be monitored jointly by MSS and Processing.

All of the information acquired through monitoring will be captured through the COTS logging mechanisms.

4.5.4.2.2 Stimulus

The Execution Management services will perform these monitoring activities on periodic basis for selected PGE runs.

4.5.4.2.3 Desired Response

The execution of a PGE will be monitored to check for anomalies which might develop with the resources, the performance of the algorithm, or with the algorithm itself. If conditions warrant, the execution of the PGE may be terminated by operations personnel.

4.5.4.2.4 Participating Classes from the Object Model

- COTS
- DpPrPge
- DpPrExecutable
- DpPrExecutionManager
- DpPrResourceManager
- DpPrComputer
- MsMgCallbacks

- MsManager
- MsEvent

4.5.4.2.5 Scenario Description

- a. During PGE execution, Execution Management Services will do the following:

Case 1: Monitor Resource Health during PGE execution

1. Check the health of the resources being used by the PGE. This is an MSS provided service.
2. If the state of the resources being used by the currently executing PGE are failing or have failed, terminate execution of the PGE gracefully.
3. The COTS Scheduler log is updated with the final state of the PGE.
4. An exception record has been submitted to the MSS Event log to indicate that the PGE has been aborted.

Case 2: Monitor performance during PGE execution

1. Check on the performance of the currently executing PGE.
2. If the usage of the resources is not following the established performance characteristics for the PGE, terminate execution of the PGE gracefully.
3. The COTS Scheduler log is updated with the final state of the PGE.
4. An exception record has been submitted to the MSS Event log to indicate that the PGE has been aborted.

Case 3: Monitor status return following PGE execution

1. The executing PGE continually updates the Toolkit Status Message Log with algorithm, or SDP Toolkit specific status information.
2. The PGE, upon detecting a fatal error condition, updates the Toolkit Status Message Log and exits gracefully, returning a predefined condition code to the COTS Scheduler.
3. This information is captured by the COTS Scheduler to determine the reason for the PGE failure.
4. The COTS Scheduler log is updated with the final state of the PGE.
5. An exception record has been submitted to the MSS Event log to indicate that the PGE has been aborted.

- b. End of "Execution Management (Monitor Execution)."

4.5.4.2.6 Event Trace

Figure 4.5-23 shows the Case 1 monitor resource health trace. Figure 4.5-24 shows the Case 2 monitor performance trace. Figure 4.5-25 shows the Case 3 monitor status return trace.

Pge using
bad Resource

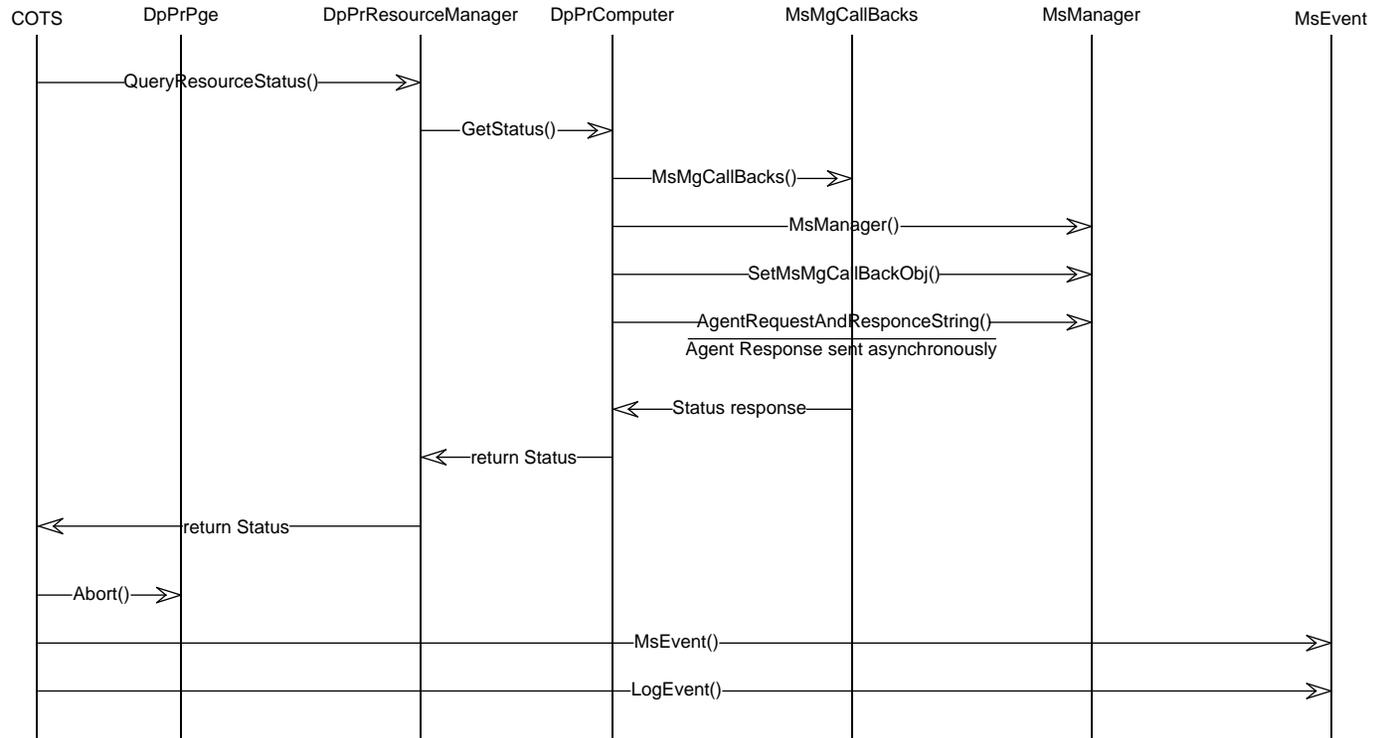


Figure 4.5-23. Monitor Resource Health

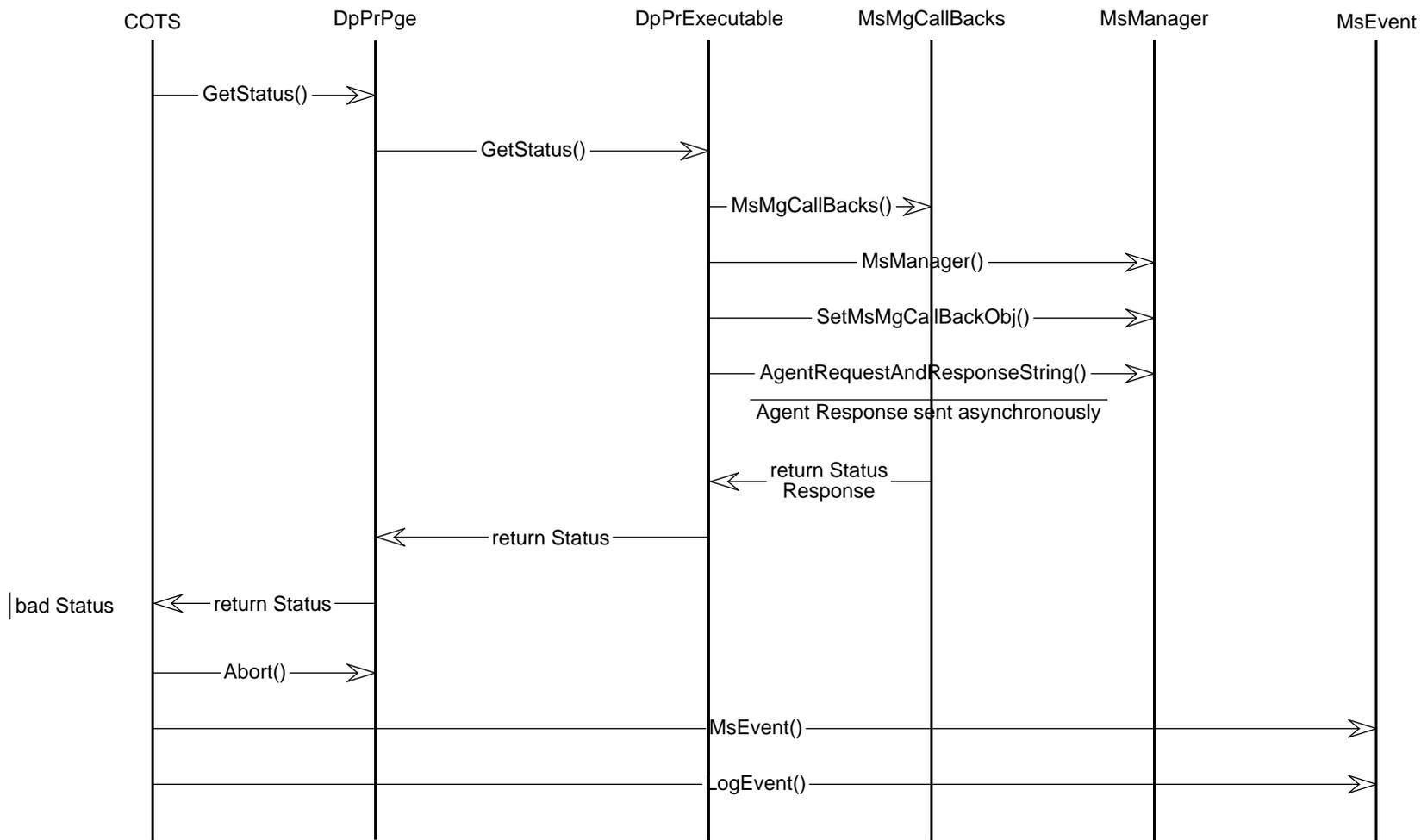


Figure 4.5-24. Monitor Performance

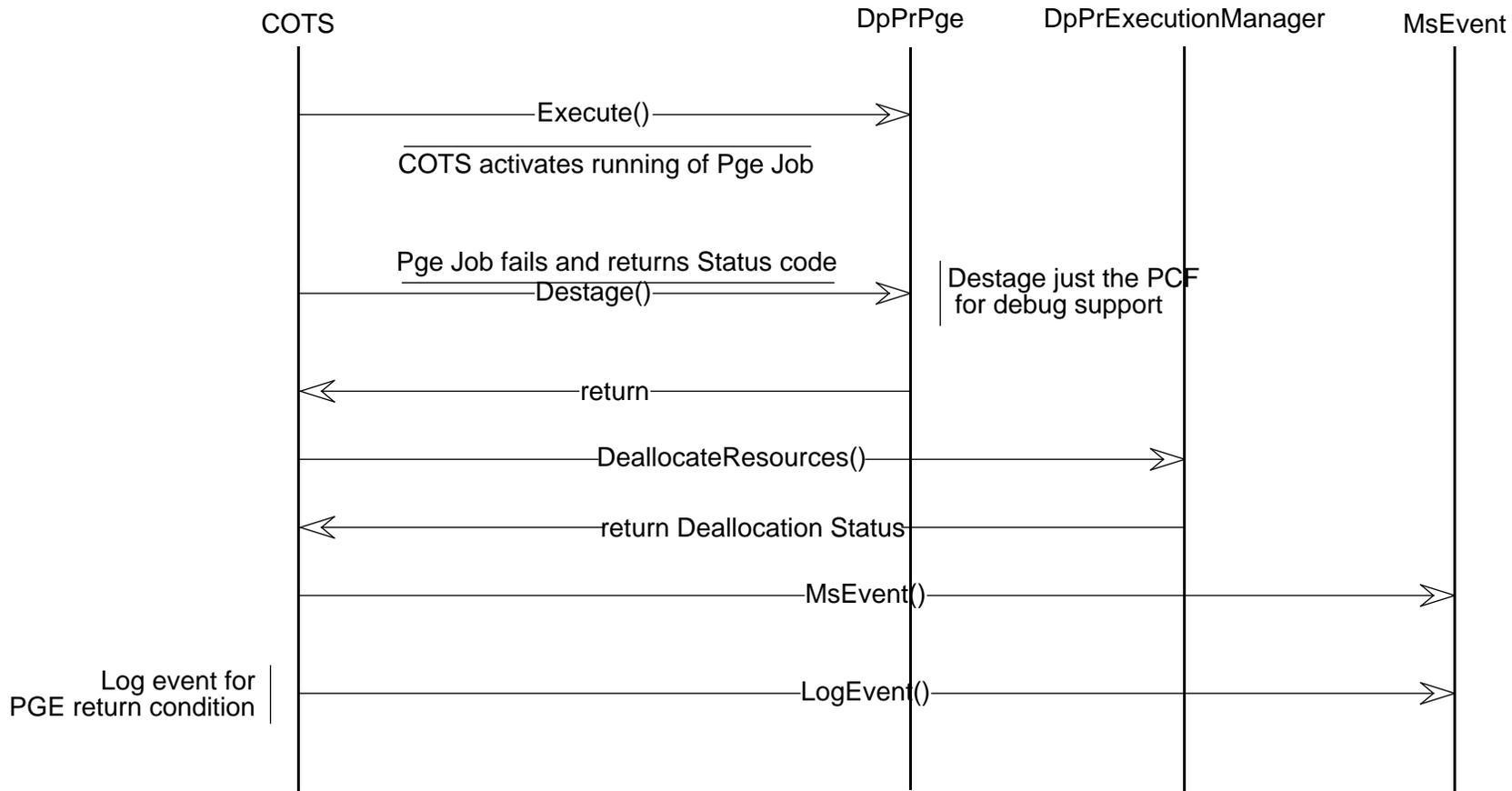


Figure 4.5-25. Monitor Status Return

4.5.4.3 Execution Post-Processing

4.5.4.3.1 Abstract

Execution Management provides services that are needed upon the completion of PGE execution. These services are used to initiate data destaging, creation of product history, deallocation of resources, etc. All generated outputs, Intermediate and Final Data Products, SDP Toolkit Process Control and Status Message Log files, etc., will be destaged to the Data Server. The generated data products may be required as input to another PGE and therefore may not be removed from the local storage. The management of the data products will be necessary to insure that data staging of a data product is not performed excessively. Execution Management services will be required to generate processing-specific metadata. This metadata will be associated to a generated data product during the process of destaging to the Data Server. Any clean-up, such as deletion of input data and output data, may also be performed by the Processing CSCI during this phase.

4.5.4.3.2 Stimulus

The successful or unsuccessful completion of the execution of a PGE. The Execution Management services will perform these activities for each PGE which has completed execution.

4.5.4.3.3 Desired Response

All processing required at the completion of PGE execution will be performed. This includes initiation of data destaging, and other clean-up activities.

4.5.4.3.4 Participating Classes from the Object Model

- COTS
- DpPrPge
- DpPrExecutionManager
- DpPrResourcemanager
- DpPrDataManager

4.5.4.3.5 Scenario Description

- a. Upon completion of PGE execution, the Execution Management Services will do the following:
 1. Check the generated errors codes to determine if the PGE output is useful.
 2. Generate processing-specific metadata for incorporation into a Production History File, to facilitate its association with the output products during the destaging operation. Production History File consists of information used to identify what occurred during execution. This includes actual resource usage, input data references, date and time of execution, and user define parameters. Most of the information located in the Process Control File and the Data Processing Request will be added to the Production History File.

3. If the PGE produced an output data product, i.e., Intermediate Data or Final Data Product, initiate destaging of the data products. In addition, destage other non-product data like Status Message Log files.

NOTE: The following two steps are performed internally by the COTS Scheduler as per the logic imparted during the job definition phase.

4. Determine if the PGE's generated output data is required by another scheduled PGE
 5. Determine if the PGE's input data is required by another scheduled PGE
 6. Destage the PGE's data mapping to preserve any runtime updates regarding Intermediate temporary files created during this run of the PGE, or requests for the transfer of runtime data files (this service, acquired through the SDP Toolkit, provides a mechanism for the preservation of critical data files used during the processing of the PGE).
 7. If the input data is not needed to execute another PGE, deallocate the resources that were used for the input data.
 8. The COTS Scheduler Log is updated to indicate the current status of the PGE. (i.e., Processing Status is "Complete").
 9. Deallocate the processing resources used by the PGE.
- b. End of "Execution Management (Execution Post-Processing).

4.5.4.3.6 Event Trace

Figure 4.5-26 shows the execution post processing event trace.

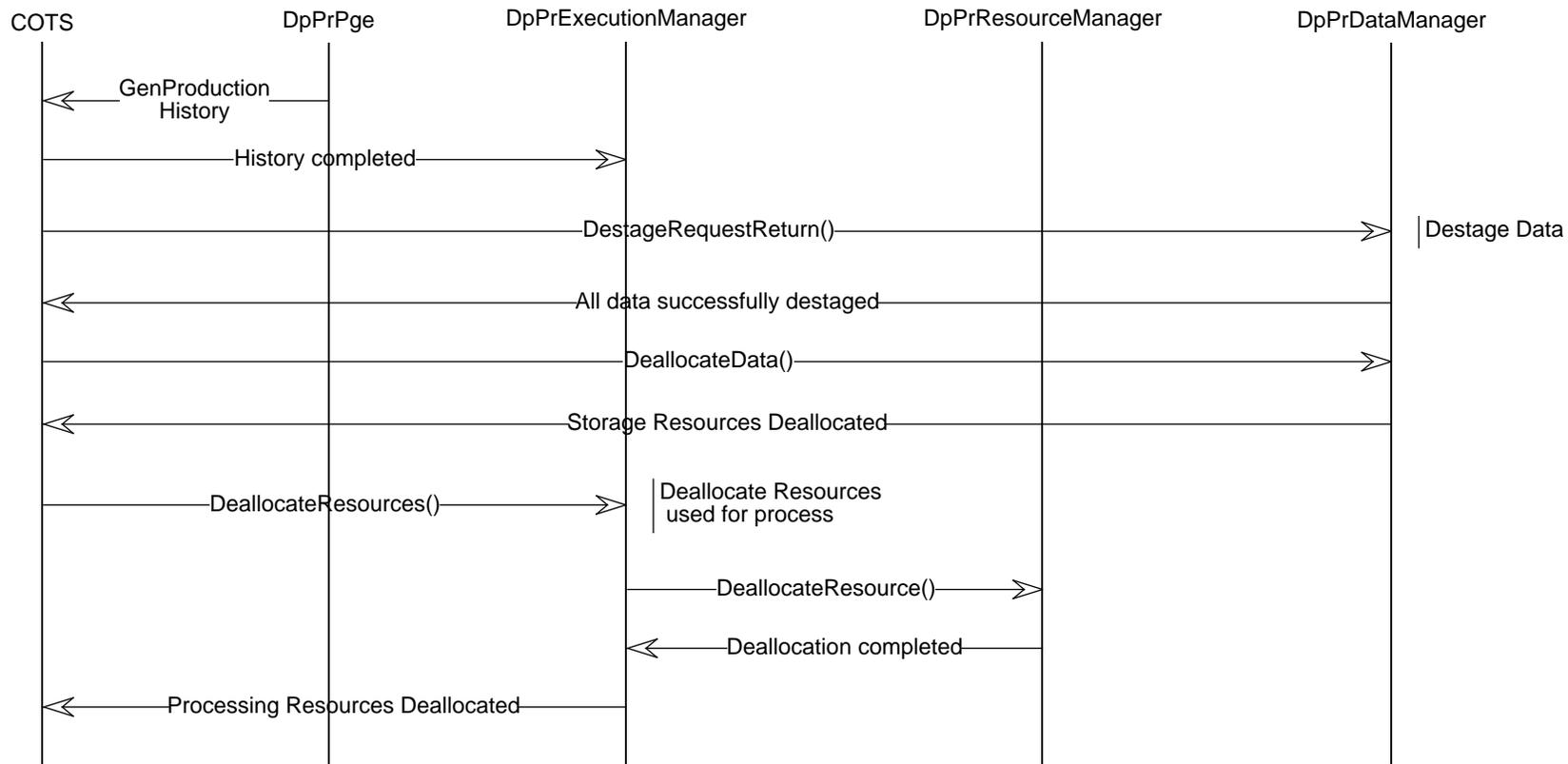


Figure 4.5-26. Execute Post Processing

4.5.4.4 Failure of Execution

4.5.4.4.1 Abstract

The failure of the execution of a PGE may be caused by either an external fault, such as a resource failure, or by an internal fault due to a science algorithm error. In order to achieve the best response to a failed PGE, that PGE must terminate gracefully and return a status condition code which conveys the reason for the failure. The COTS Scheduler will determine, from this code, what further processing may be performed from the job interdependencies which are already defined, or through human interaction with the Alarm Manager utility. If a resource failed, there may be an attempt to move the PGE to similar resources in order to initiate execution of the PGE again. If the PGE failed because of an internal fault, an attempt will be made to determine the cause of this fault through the status condition code. If there is hope that the PGE may execute properly, the COTS Scheduler may initiate execution again.

4.5.4.4.2 Stimulus

The stimulus which causes a PGE to fail during execution comes from a fatal error condition that occurs during processing of the algorithm. Whether internal, or external to the PGE, the fault should be handled properly by the science software.

4.5.4.4.3 Desired Response

Upon detection of the fatal error condition, the error handling portion of the PGE should take-over to redirect processing, or exit gracefully; the latter case is depicted in this scenario.

4.5.4.4.4 Participating Classes from the Object Model

- COTS
- DpPrExecutable
- DpPrPcf
- DpPrPge
- DpPrExecutionManager
- DpPrResourceManager
- DpPrDataManager
- MsEvent
- PIDataGranule
- PIDPRB
- DsCIESDTRreferenceCollector
- DsCIRequest
- DsCICommand

4.5.4.4.5 Scenario Description

- a. When the PGE fails due to a fatal error condition during processing, the following activities are performed:
 1. Retrieve the error condition code from the PGE.
 2. To signal that recovery attempts are underway, update the COTS Scheduler Interface to display "PGE Failed".
 3. Destage the Process Control File for this run to preserve a copy for SCF investigation.
 4. Destage whatever output data was produced. This data will not be archived necessarily, but may be used for investigation into the cause of the malfunction (the actual list of data to preserve is identified at runtime by the science software, through interaction with the SDP Toolkit; if nothing else, the PGE should handle the error to the point of being able to specify the set of data for post-mortem analysis).
 5. At this point, the resources are still allocated and all of the input data is still staged. Just prior to resource deallocation, checks are performed on the health of the allocated resources to determine if the fault was external or internal to the PGE.
 - (a) For the latter case, the COTS Scheduler proceeds with cleanup activities in order to make room for subsequent processing. If the cause was due to resource problems, the PGE may be re-initiated after sufficient resources are re-allocated.
 - (b) If resources cannot be reallocated to re-initiate the PGE, then the job which represents the Data Processing Request for this PGE gets reprioritized within the COTS Scheduler, to be retried at a later time.
 6. Deallocate all of the Resources used for this processing run.
 7. Log a report on the health of currently allocated resources to MSS.
 8. Also, send a message to the COTS Scheduler Log indicating that processing status is "Failure."
- b. End of "Execution Management (Failure of Execution)."

4.5.4.4.6 Event Trace

Figure 4.5-27 shows the failure of execution event trace.

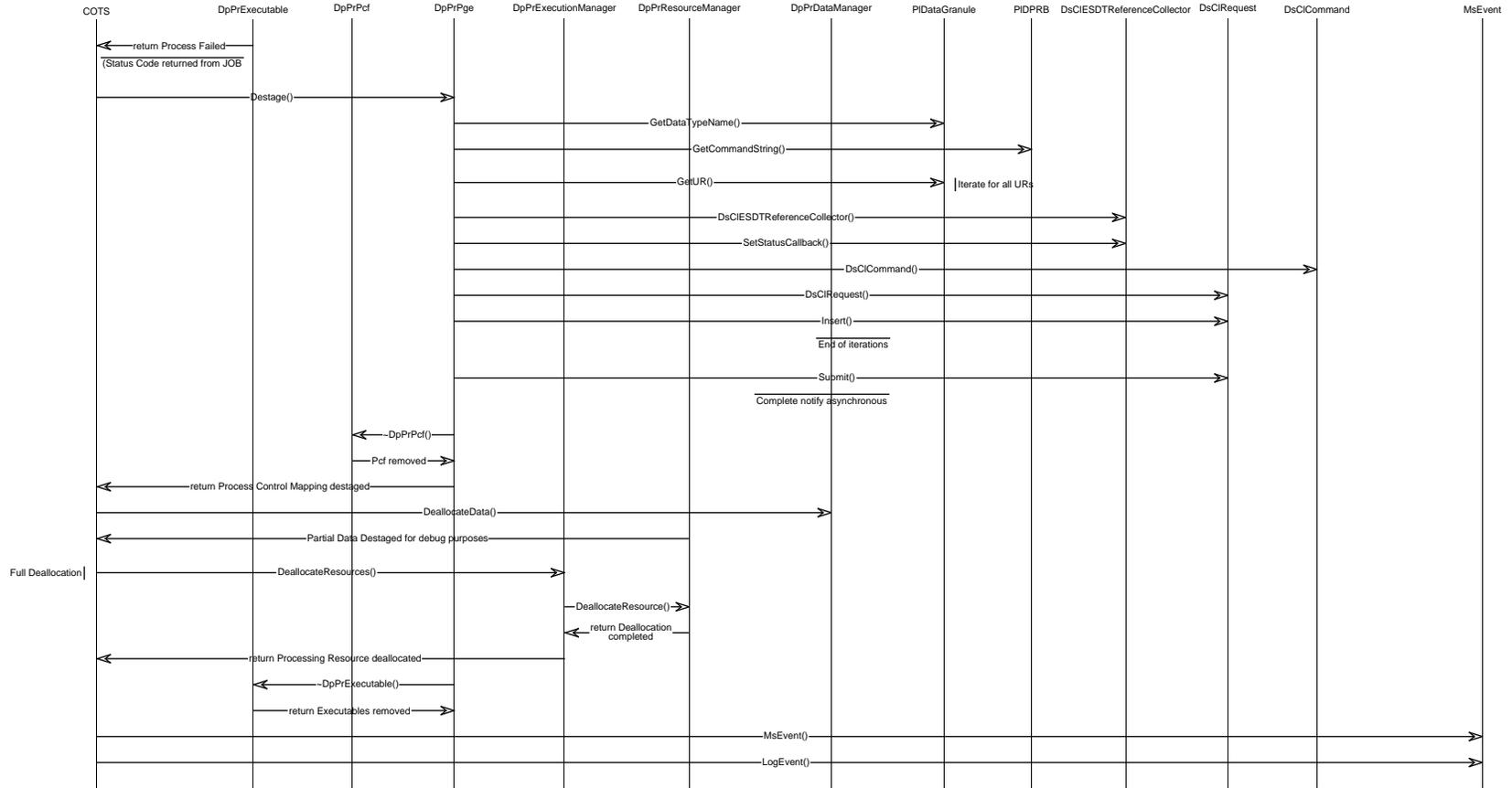


Figure 4.5-27. Failure of Execution

4.5.4.5 Failure of Processing Resource

4.5.4.5.1 Abstract

If a resource fails during execution of a PGE, the COTS Scheduler will attempt to re-initiate the PGE using other suitable resources, if local policies so dictate. This will be done only if the resource information in the Data Processing Request is generic enough to allow this transferal. If the resource information is tied to a particular computer because of the unique characteristics of this computer, other error recovery actions, including Processing Operations staff intervention, may be initiated.

4.5.4.5.2 Stimulus

The stimulus which causes a PGE to fail during execution, due to a problem with a resource, originates from a fatal error condition that is detected during processing of the algorithm. Most likely, these type of external error conditions will not be overcome by the algorithm; the algorithm is still expected to fail in a graceful manner to ensure the return of an appropriate error condition code.

4.5.4.5.3 Desired Response

The execution of a PGE will be terminated by the error handling component of the PGE; all PGEs should contain a termination section built into the highest-level module of the PGE, through which control ultimately passes and a final process status gets defined (e.g., at the end of a PGE shell script). The resource which created the error condition should be replaced so that the PGE can be reactivated.

4.5.4.5.4 Participating Classes from the Object Model

- COTS
- DpPrExecutable
- DpPrPcf
- DpPrPge
- DpPrExecutionManager
- DpPrResourceManager
- DpPrDataManager
- MsMgCallbacks
- MsManager
- MsEvent

4.5.4.5.5 Scenario Description

- a. When the PGE fails due to a resource-induced fatal error condition during processing, the following activities are performed:
 1. Send a PGE condition code "Failed due to resource" to the COTS Scheduler

2. Signal that recovery efforts are under way - Update the COTS interface to display "PGE Failed"
 3. Retrieve the error condition code from the PGE and activate the appropriate recovery job (this is performed automatically through predefined job definitions).
 4. Destage whatever output data was produced. This data will not be archived necessarily, but may be used for investigation into the cause of the malfunction (the actual list of data to preserve is identified at runtime through science software interaction with the SDP Toolkit).
 5. Destage the Process Control File (PCF) for this run to preserve a copy for SCF investigation.
 6. At this point, the resources are still allocated and all of the input data is still staged. The Resource Management service is activated, as part of the data deallocation phase, to perform a check on the health of the allocated resources to determine if the fault was external or internal to the PGE.
 - (a) If the cause was due to resource problems, the PGE may be re-initiated after sufficient resources are re-allocated. For the latter case, the Resource Management service proceeds with cleanup activities in order to make room for subsequent processing.
 - (b) If resources cannot be re-allocated to re-initiate the PGE, the Data Processing Request for this PGE gets reprioritized within the COTS Scheduler to be retried at a later time.
 7. Update the operational mode of the wayward resource to "OFFLINE."
 8. Log a report on the health of currently allocated resources to MSS.
 9. Deallocate the Processing resources. All CPU resources will be reclaimed but depending on the nature of the resource failure, only part of the PGE storage resources may be deallocated.
 - (a) If the cause of the resource failure was due to one particular disk device (i.e., file system) on the local host then simply deallocate that resource if it was allocated for the failed PGE. Only those PGE constituents which were resident on that resource will need to be restaged
 - (b) If however the entire host contributed to the resource problems, then all resources for that host will have to be deallocated. This means that all executables and Status Message Files for the PGE will have to be restaged to a different host for a subsequent run.
 10. Send a message to the COTS Scheduler Log indicating that processing status is "Failure."
- b. End of Scenario.

4.5.4.5.6 Event Trace

Figure 4.5-28 shows the failure of processing resource event trace.

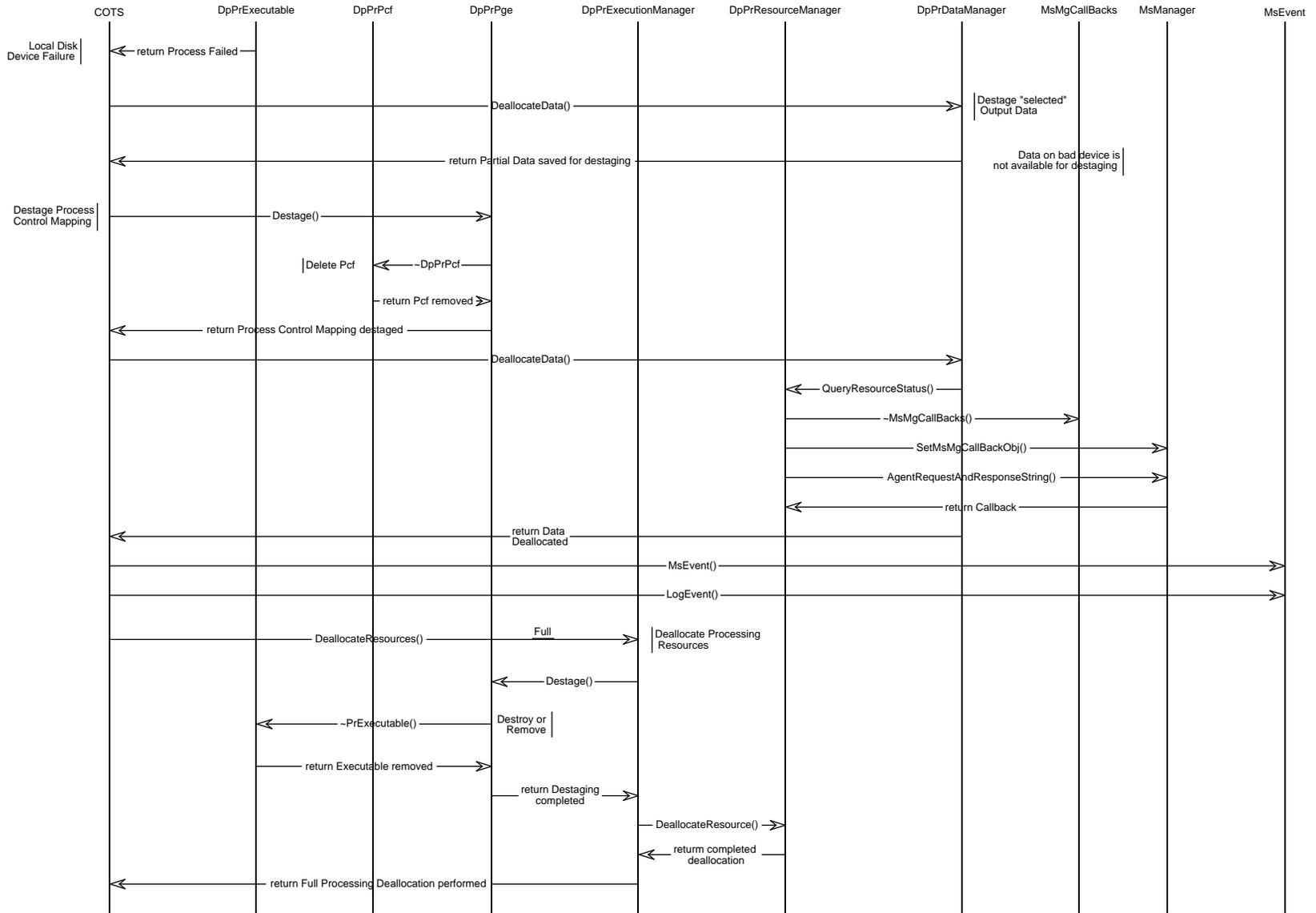


Figure 4.5-28. Resource Failure

4.5.4.6 Expand DPR job

4.5.4.6.1 Abstract

This scenario describes how the jobs in a job box are expanded after the resource allocation is completed successfully.

4.5.4.6.2 Stimulus

When the job dependencies are met for a resource allocation job, Autosys will activate the resource allocation job. If the resource allocation job is completed successfully, it will trigger the creation of other jobs in the job box.

4.5.4.6.3 Desired Response

The jobs in the job box are created and get inserted in to the Autosys database

4.5.4.6.4 Participating Classes from the Object Model

- PIDPRB
- COTS
- PIPGE
- DpPrExecutionManager
- DpPrCotsManager

4.5.4.6.5 Description

When the job dependencies are met for the resource allocation job of a DPR, Autosys will kill it off. If the resource allocation is successful, this job will get the DPR/PGE related parameters from PIDPRB and PIPGE and then call the operation AddJob in DpPrCotsManager to add the data staging job, PGE prep job, PGE job, destaging job and the resource deallocation job to the Autosys database.

4.5.4.6.6 Event Trace

Figure 4.5-29 shows the expand jobs event trace.

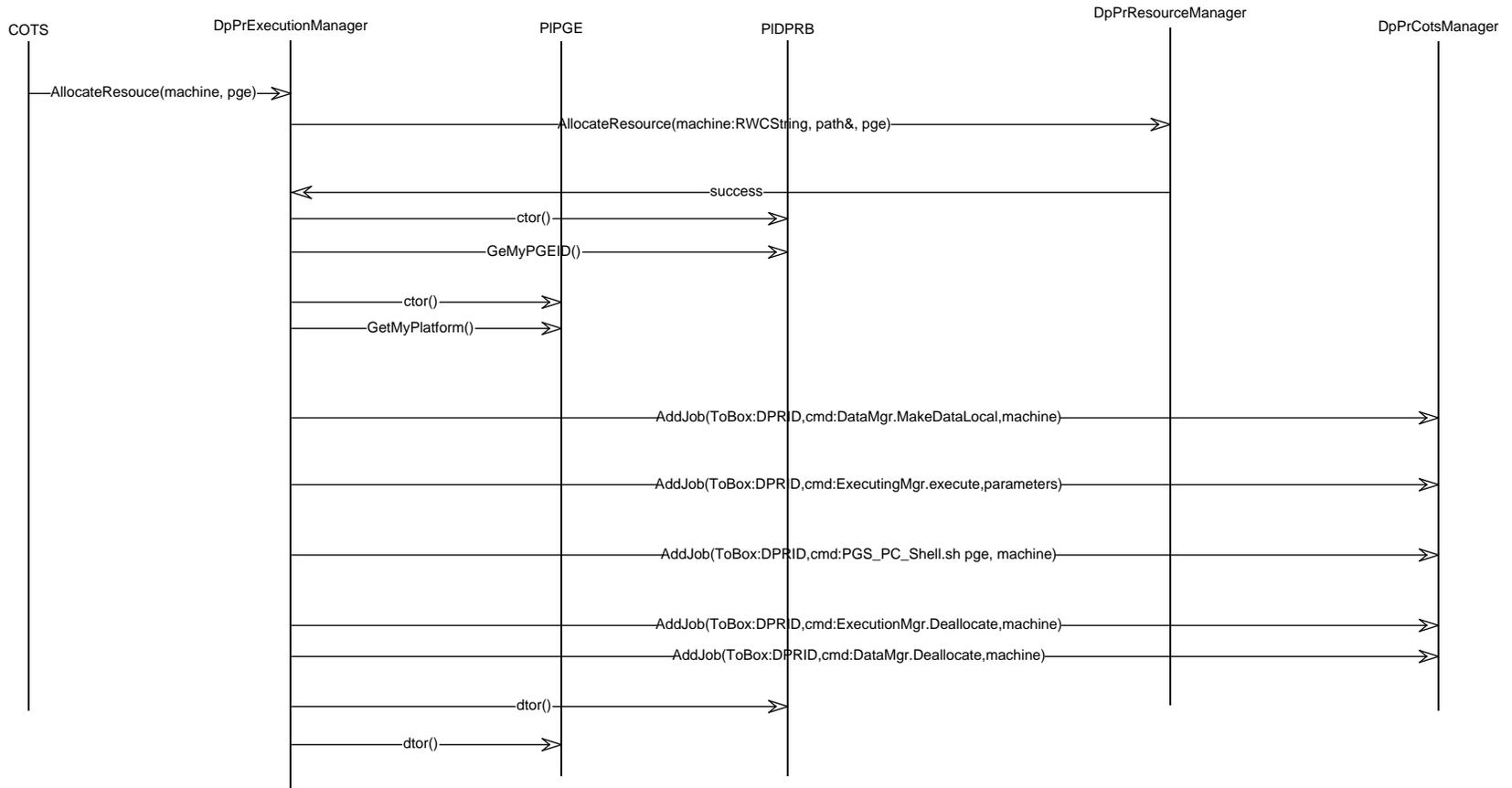


Figure 4.5-29. Expand Data Processing Request Jobs

4.5.5 Resource Management Scenarios

Resource Management services are provided to manage and monitor the Data Processing subsystem resources which are used exclusively in the production environment. These services will be used to maintain the information needed to track the health and availability of these resources and coordinate their allocation and deallocation within the Data Processing subsystem.

Resource Management services include the following activities:

- a. Initialization of Resource Management Information—This activity occurs at system initialization and on the update of resource availability information as it is received from MSS. This is the interface which will be used, in conjunction with MSS, to provide the initial resource configuration and updated resource information. See scenario "Resource Management: Initialization of Resource Management Information."
- b. Modification of Resource Management Information—This activity occurs when resources are allocated or deallocated to support data staging/destaging and PGE execution. See scenario "Resource Management: Modification of Resource Management Information."
- c. Query Resource Management Information—This activity occurs whenever Data Management is checking for available resources, or when processing resources are needed to stage a PGE and its components to a local resource. See scenario "Resource Management: Query Resource Management Information" for more details.

4.5.5.1 Resource Management Configuration Initialization

4.5.5.1.1 Abstract

The Planning and Data Processing resource characteristics are collected, maintained and used to determine whether or not sufficient resources are available to support the Data Processing subsystem production, such as the execution of PGEs and the staging of data for Data Processing Requests. The initialization of these resources is accomplished through interaction with MSS to create a working model of only those resources which may be used by Planning and Processing.

4.5.5.1.2 Stimulus

The stimulus for the initialization of the Resource Management Configuration is as follows:

The initialization of the Resource Management information is performed at Planning subsystem start-up through manual interaction with the Planning workbench and at production plan activation time.

4.5.5.1.3 Desired Response

A list of resources will be created and the data necessary to manage the resources, such as current state (OFFLINE, ONLINE, etc.) and characteristics (e.g., amount of usable disk space) will be recorded.

4.5.5.1.4 Participating Classes From the Object Model

- ResourceManager
- PIResourceUI

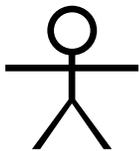
- DpPrResourceConfiguration
- MsDAAC
- DpPrString
- DpPrComputer
- DpPrDiskPartition
- DpPrDiskAllocation

4.5.5.1.5 Scenario Description

- a. Through manual interaction with a Planning provided User Interface (UI), the Resource Manager activates the method to build the resource configuration.
- b. Through the iterative application of a filtering method on MSS services, current DAAC resource information which applies to Planning and Processing can be retrieved.
 1. Create the appropriate resource objects using the configuration information obtained from MSS. Initialize data for each resource in the Data Processing subsystem.
 - (a) Initialize Computer resource information—Retrieve machine type, operating system, number of CPUs, total RAM and current state.
 - (b) Initialize Disk Device resource information—Retrieve file system base paths, block size, total partition size and current state.
 - (c) Initialize Disk Allocation information—Retrieve all of the system uses for this Disk Device.
 2. Create a String object and associate it with this Computer resource, or add the resource to the String if it's already defined.
- c. End of Resource Management (Initialization of Resource Configuration Information).

4.5.5.1.6 Event Trace

Figure 4.5-30 shows the initialization of resource configuration information event trace.



Resource Manager
Resource Manager

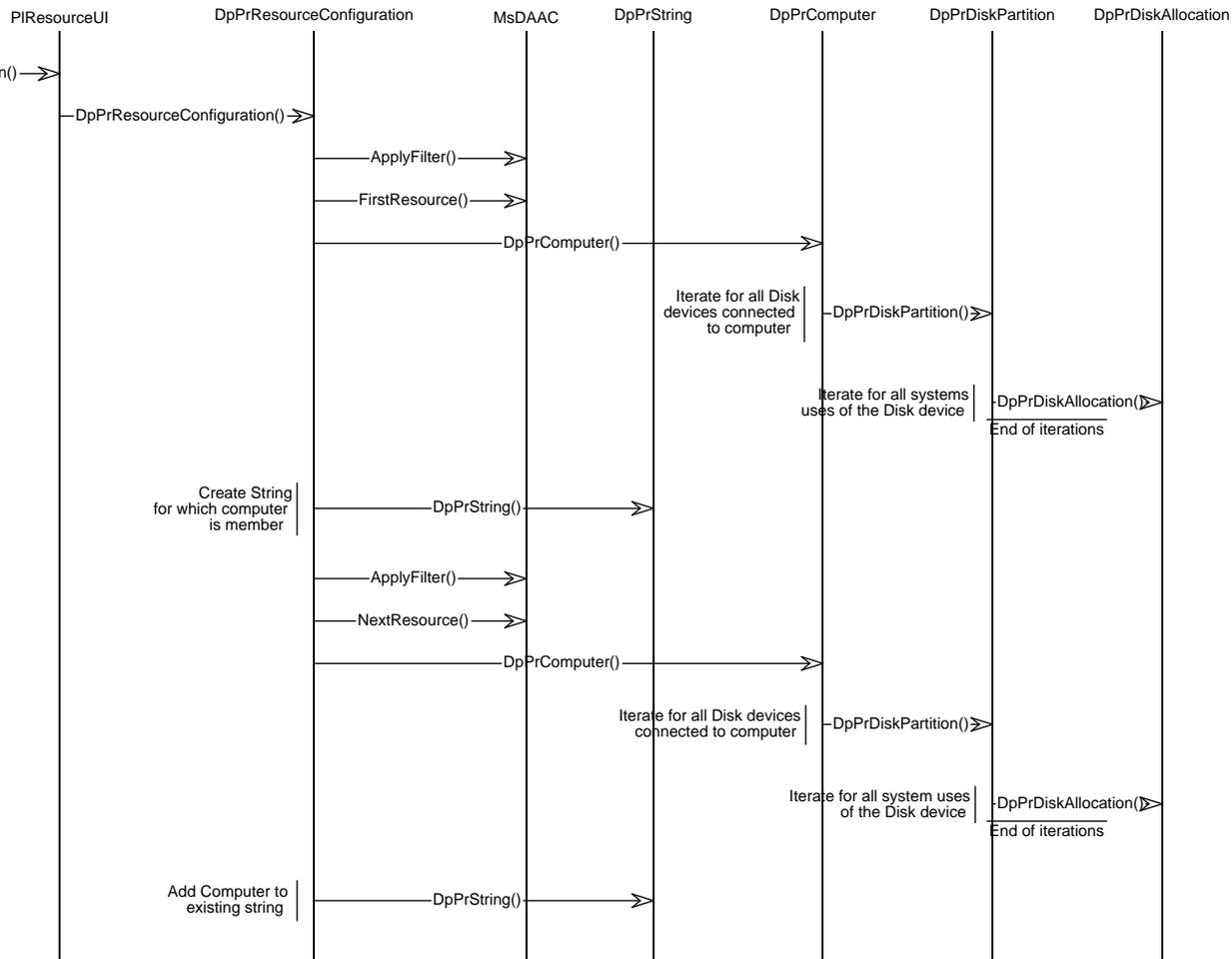


Figure 4.5-30. Initialization of Resource Configuration

4.5.5.2 Modify the Resource Management Information

4.5.5.2.1 Abstract

The modification of the Resource Management information is performed whenever resources are used to support the Processing CSCI, such as at the beginning, or end of the execution of a COTS Scheduler defined job (i.e., Data Processing Request), or the beginning or end of the staging or destaging of data. The state of a resource may also change as a result of system error conditions, or as a resource is scheduled to be out of service.

4.5.5.2.2 Stimulus

The stimuli for the modification of the Resource Management List is the following:

- a. Data Staging
- b. Data Destaging
- c. PGE Execution
- d. Resource Update Information obtained through MSS contact.

4.5.5.2.3 Desired Response

The entry of the resource to be modified will be updated to indicate current state of allocation, and/or changed resource characteristics.

4.5.5.2.4 Participating Classes From the Object Model

- COTS
- DpPrPge
- DpPrExecutionManager
- DpPrResourceManager
- DpPrDataManager
- MsMgCallbacks
- MsManager

4.5.5.2.5 Scenario Description

- a. Whenever the COTS Scheduler executes a job to allocate or deallocate resources, or if an update to the status of resources has been acquired through MSS, do the following:
 1. If a Resource status update has been obtained from MSS then
 - (a) Update the State of the Resource.
Resource State = { ONLINE, OFFLINE, etc. }
 2. If Allocation of resources required for data staging and PGE Execution then
 - (a) Update allocation information for the resources involved in the current processing.
 - (1) Allocate disk space to support Data staging and Output Generation.

- (2) Allocate disk space required to support the hosting of the PGE's executables and runtime Status Message Files (SMFs). This step will only be necessary if the runtime files were not already resident due to first time run of the PGE, or prior resource problems which necessitated the removal of the files following a prior run.
- (3) Allocate the disk space required to support the Process Control File (PCF) for this run of the PGE.
- (4) Allocate the CPU(s) required to support PGE Execution.
- (b) If Allocation failed due to resource problems then
 - (1) Obtain a resource status update from the MSS agent.
 - (2) Update the local resource status information and return an error condition to the COTS Scheduler.
 - (3) Depending on the error condition, re-allocation of resources may be performed to support the current processing (see 2a).
3. Following data destaging, if deallocation of resources required then
 - (a) Update information for the resources involved in the processing that just completed. If the same PGE will be executing again on the same host, disk space for the PGE's constituents (e.g., executable files) may not be deallocated.
 - (1) Deallocate disk space to support Data staging and Output Generation.
 - (2) Deallocate disk space required to support the Process Control File (PCF).
 - (3) Deallocate the CPU(s) required to support PGE Execution.
 - b. End of Resource Management: Modify Resource Management Information.

4.5.5.2.6 Event Trace

Figure 4.5-31 shows the modify resource information event trace.

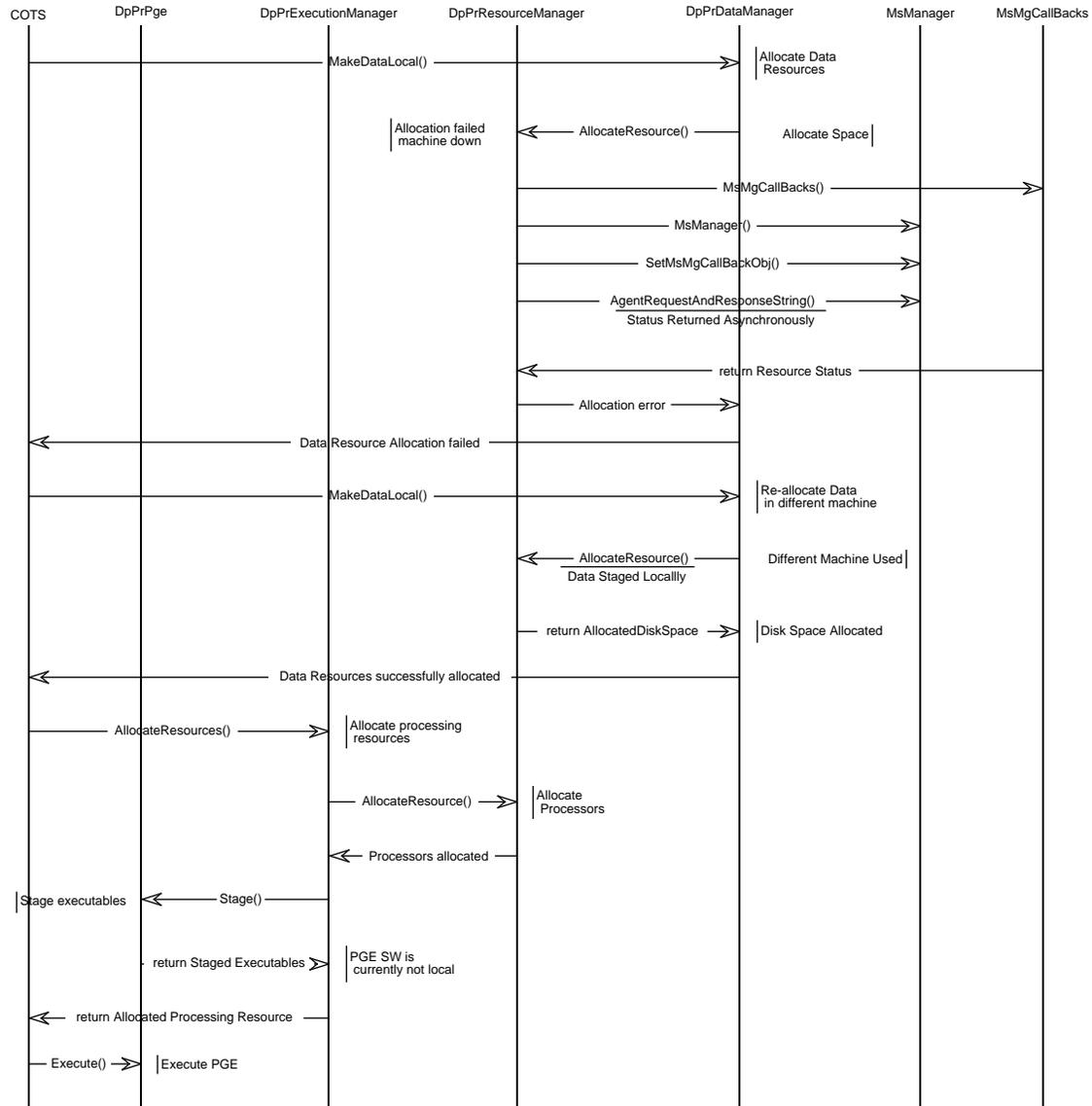


Figure 4.5-31. Modify Resource

4.5.5.3 Query the Resource Management List

4.5.5.3.1 Abstract

A query of the Resource Management information is performed to determine if resources are available to support the execution of a PGE. This scenario may also occur when checking to see if resource faults have occurred during PGE execution.

4.5.5.3.2 Stimulus

The stimuli for the query of the Resource Management information is the following:

This operation is performed whenever an allocation of processing resources is requested as a prelude to the execution of a PGE.

4.5.5.3.3 Desired Response

The information being requested for a set of resources will be produced.

4.5.5.3.4 Participating Classes From the Object Model

- COTS
- DpPrExecutionManager
- DpPrResourceManager
- DpPrComputer
- DpPrDiskPartition

4.5.5.3.5 Scenario Description

- a. When the Execution Management services are performing the initiation of staging and execution of a PGE, determine whether the appropriate resources will be available to support PGE execution.
 1. Query for information on the set of resources required by the PGE.
 - (a) Resource State {ONLINE, OFFLINE, etc.}
 - (b) Number of available processors
 - (c) Amount of available disk space
 - (d) Memory configuration for the machine
 2. End of Resource Management (Query of Resource Management Information).

4.5.5.3.6 Event Trace

Figure 4.5-32 shows the query of resource management information event trace.

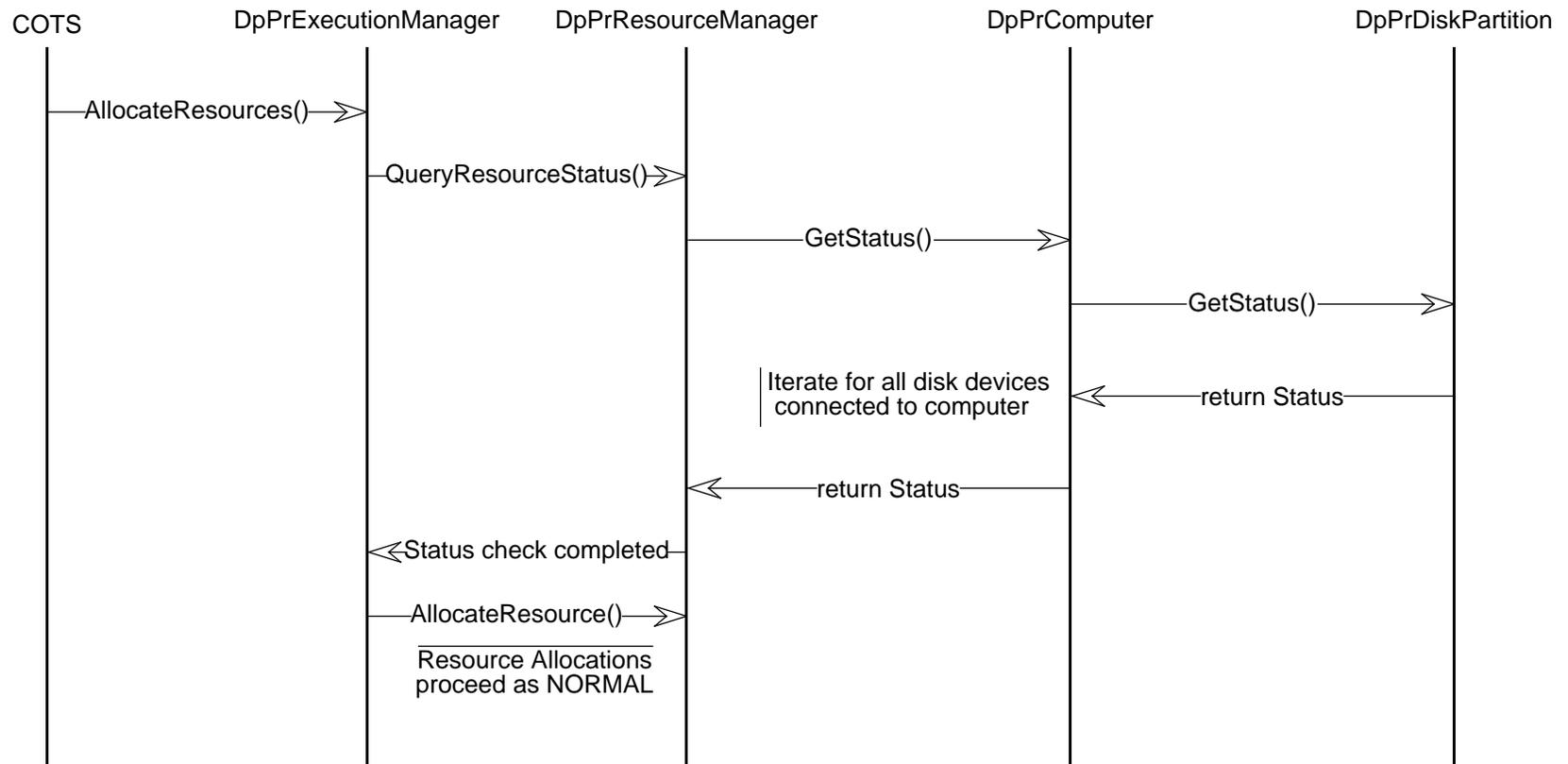


Figure 4.5-32. Query Resource Management Information

4.5.5.4 Report Resource Usage

4.5.5.4.1 Abstract

This scenario describes how the data processing subsystem collects resource usage information and send it to MSS.

4.5.5.4.2 Stimulus

When Autosys starts the execution of a PGE job through the execution manager, the execution manager will trigger the resource collection process.

4.5.5.4.3 Desired Reposes

The resource usage information including disk usage, CPU usage etc is sent to MSS for accounting purposes.

4.5.5.4.4 Participating Classes from the Object Model

- DpPrExecutionManager
- DpPrResourceUsageNB
- DpPrPcf
- MsBaCotsIFB

4.5.5.4.5 Description

When Autosys starts a PGE job through execution manager, the execution manager will call SetTimeAndIOUsage operation of DpPrResourceUsageNB to collect CPU usage and IO usage related to this PGE. After the PGE is completed, the execution manager will call SetDiskUsage operation of DpPrResourceUsageNB to collect disk usage. Finally the resouce usage information is sent to MSS through MsBaCotsIFB class.

4.5.5.4.6 Event Trace

Figure 4.5-33 shows the report resource usage event trace.

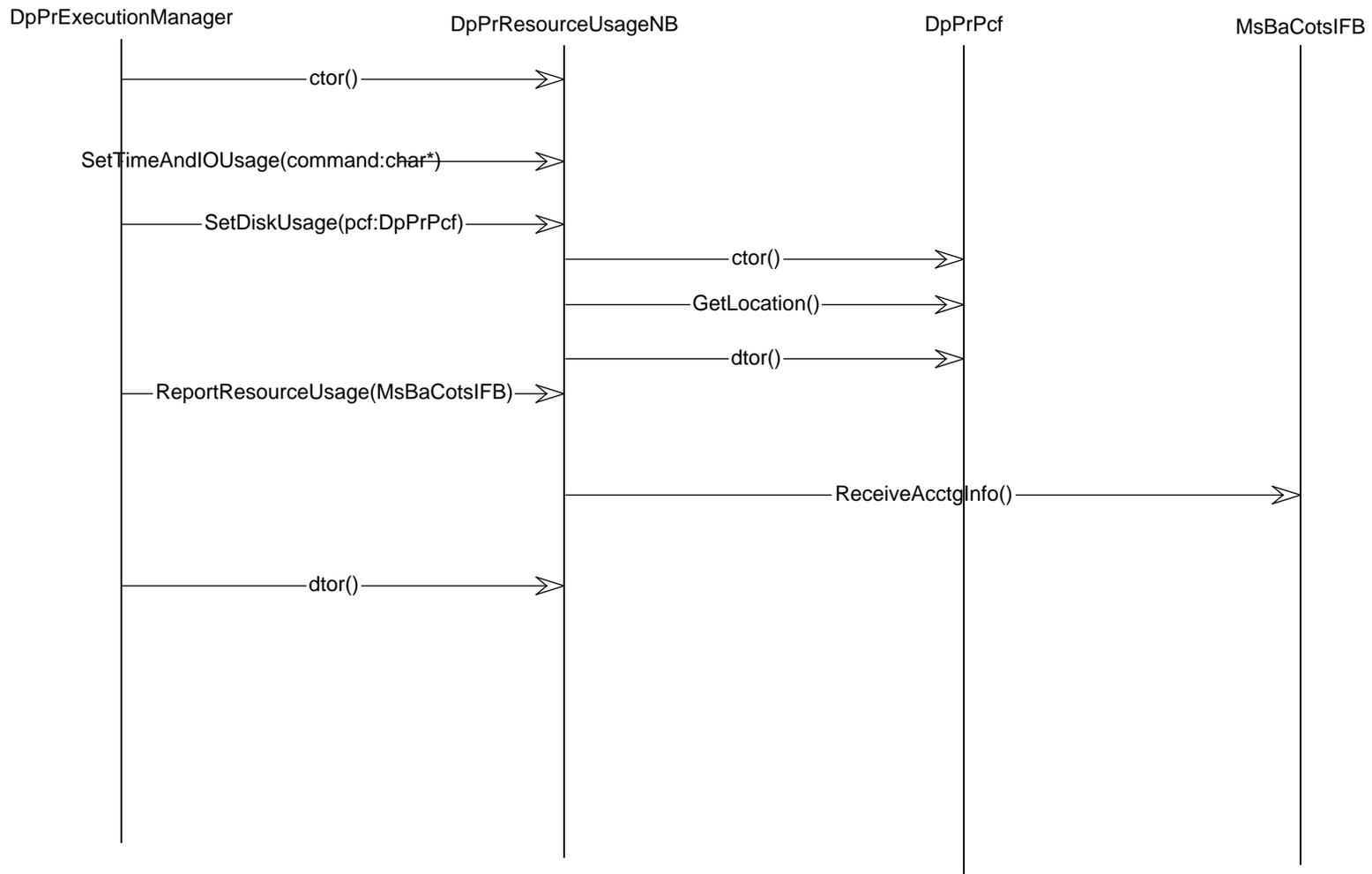


Figure 4.5-33. Report Resource Usage

4.5.6 Quality Assurance Scenarios

The Quality Assurance Scenarios represent DAAC manual QA and DAAC non-science QA activities that are functions of the Processing CSCI. DAAC manual QA activities require use of data visualization tools for viewing products, subscription submittal and withdrawal services for requesting and retrieving products from the data server, and editing tools for updating product metadata. These activities are supported by Client provided functions. Additional unique functions only required for DAAC manual quality will be provided by the Processing CSCI. DAAC non-science QA activities represent processing that are intrinsic to the production process particularly post-production processing.

4.5.6.1 Q/A Subscription Submittal

4.5.6.1.1 Abstract

A Q/A position can subscribe to data products generated by the Processing CSCI's execution of a PGE. The Data Server then informs Q/A when the product has been generated and is available for review. This allows Q/A to be decoupled from the generation and initial storage of the data product. It also means that Q/A activities do not have to occur as soon as the product is generated.

4.5.6.1.2 Stimulus

Starting of the Q/A Monitor Command GUI with the intent of submitting a subscription will initiate the processing in this scenario.

4.5.6.1.3 Desired Response

The following actions will occur on the subscription request input:

- a. The Data Server will be notified that Q/A wishes to subscribe to a data product.
- b. The Data Server enters the subscription request.
- c. The fact that the product is subscribed to is recorded in the PDPS database.

4.5.6.1.4 Participating Classes From the Object Model

- DpPrQaMonitor
- PIDataTypes
- PIDataType
- AdCollection
- Advertisement
- DsClSubscription

4.5.6.1.5 Scenario Description

- a. The Q/A position brings up the Monitor Command Window and selects *Display Data Types for Subscription*.
- b. The Data Type Selection Window comes up, displaying the data types which are not currently subscribed to.

- c. The Q/A position selects the data type for subscription; the Data Type Selection Window goes away.
- d. The Q/A position selects to *Submit the Subscription* from the Monitor Command Window.
- e. A subscription (Data Server Client public class) is created with a command type of Submit and submitted to the Data Server.
- f. The myQaSubscription flag in the Data Type information recorded in the PDPS database is set to *TRUE*.

4.5.6.1.6 Event Trace

Figure 4.5-34 shows the Q/A subscription submittal event trace.

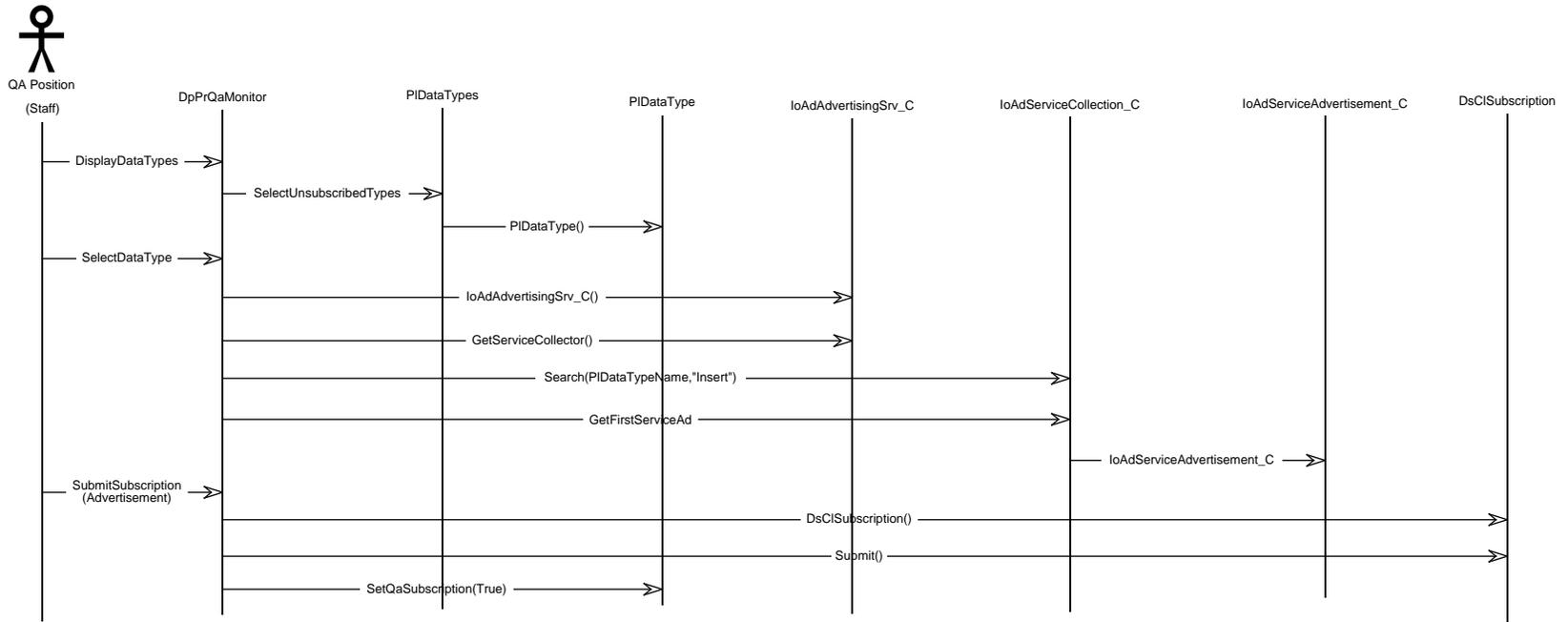


Figure 4.5-34. Q/A Subscription Submittal

4.5.6.2 Q/A Subscription Withdrawal

4.5.6.2.1 Abstract

A Q/A position can withdraw subscriptions to data products generated by the Processing CSCI's execution of a PGE. The Data Server no longer informs Q/A when products of that data type have been generated.

4.5.6.2.2 Stimulus

Starting of the Q/A Monitor Command GUI with the intent of withdrawing a subscription will initiate the processing in this scenario.

4.5.6.2.3 Desired Response

The following actions will occur on the subscription withdrawal input:

- a. The Data Server will be notified that Q/A wishes to withdraw its subscription to a data product.
- b. The Data Server enters the subscription withdrawal request.
- c. The fact that the product is no longer subscribed to is recorded in the PDPS database.

4.5.6.2.4 Participating Classes From the Object Model

- DpPrQaMonitor
- PIDataTypes
- PIDataType
- AdCollection
- Advertisement
- DsClSubscription

4.5.6.2.5 Scenario Description

- a. The Q/A position brings up the Monitor Command Window and selects *Display Data Types for Subscription Withdrawal*.
- b. The Data Type Selection Window comes up, displaying the currently subscribed-to data types.
- c. The Q/A position selects the data type for withdrawal; the Data Type Selection Window goes away.
- d. The Q/A position selects to *Withdraw the Subscription* from the Monitor Command Window.
- e. A subscription (Data Server Client public class) is created with a command type of withdraw and submitted to the Data Server.
- f. The myQaSubscription flag in the Data Type information recorded in the PDPS database is set to *FALSE*.

4.5.6.2.6 Event Trace

Figure 4.5-35 shows the Q/A subscription withdrawal event trace.

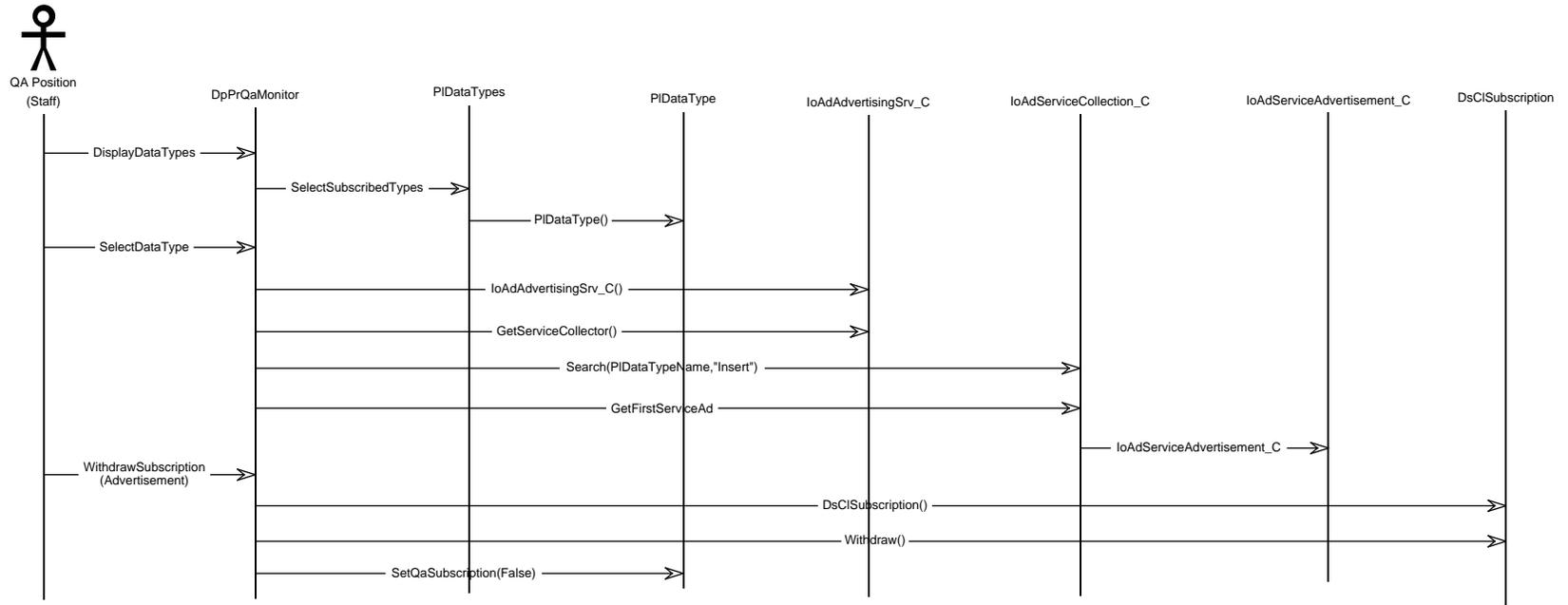


Figure 4.5-35. Q/A Subscription Withdraw

4.5.6.3 Q/A Get Data

4.5.6.3.1 Abstract

When a product has been generated by a PGE and is available for review, the Data Server checks to see if there are any Q/A subscriptions to that data. If there are, the Data Server sends e-mail to the Q/A position stating that the product has been generated and is available for review.

Then, when the Q/A position is ready to review the product, it starts up the Q/A Monitor Command GUI and can request the data from the Data Server.

4.5.6.3.2 Stimulus

Entering the GetData command in the Q/A Monitor Command GUI will initiate the processing in this scenario.

4.5.6.3.3 Desired Response

The following actions will occur upon the stimulus of this scenario:

- a. Q/A enters a request for subscribed-to data.
- b. The data is retrieved from the Data Server and returned to the Q/A position.

4.5.6.3.4 Participating Classes From the Object Model

- DpPrQaMonitor
- DsCIESDTReference
- PIDataGranules
- PIDataGranule

4.5.6.3.5 Scenario Description

- a. The Q/A position selects to *Retrieve Data* from the Monitor Command Window, using the UR obtained in the e-mail message.
- b. The correct instance of the data type is retrieved from the Data Server.

4.5.6.3.6 Event Trace

Figure 4.5-36 shows the Q/A get data event trace.



Qa Position
(Staff)

DpPrQaMonitor

DsCIESDTReference

PIDataGranules

PIDataGranule

GetData(UR)

ctor(UR)

Inspect(esdtParmList)

MatchInstances(StartTime, StopTime)

ctor(Instance)

Return Data

4-309

305-CD-027-002

Figure 4.5-36. Q/A Get Data

4.5.6.4 Q/A Visualize Data

4.5.6.4.1 Abstract

When a product has been generated by a PGE and is available for review, the Data Server checks to see if there are any Q/A subscriptions to that data. If there are, the Data Server sends e-mail to the Q/A position stating that the product has been generated and is available for review.

Then, when the Q/A position is ready to review the product, it starts up the Q/A Monitor Command GUI and can choose to visualize the data. This will retrieve the data from the Data Server and invoke EOSVIEW to display a visual interpretation of the product.

4.5.6.4.2 Stimulus

Entering the Visualize Data command in the Q/A Monitor Command GUI will initiate the processing in this scenario.

4.5.6.4.4 Desired Response

The following actions will occur upon the stimulus of this scenario:

- a. Q/A enters a request to visualize subscribed-to data.
- b. The data is retrieved from the Data Server and returned to the Q/A position.
- c. EOSVIEW is invoked to display a visual rendition of the product.

4.5.6.4.4 Participating Classes From the Object Model

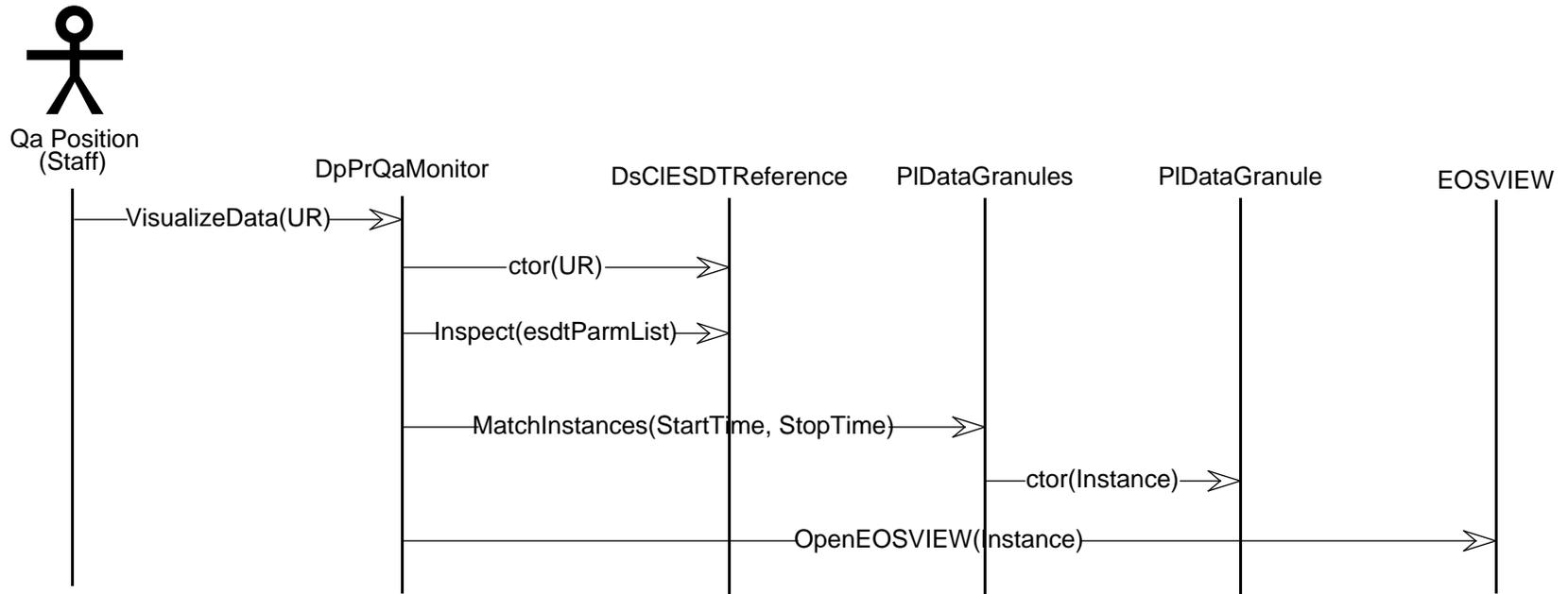
- DpPrQaMonitor
- DsCIESDTRreference
- PIDataGranules
- PIDataGranule
- EOSVIEW

4.5.6.4.5 Scenario Description

- a. The Q/A position selects to *Visualize Data* from the Monitor Command Window, using the UR obtained in the e-mail message.
- b. The correct instance of the data type is retrieved from the Data Server.
- c. EOSVIEW is invoked with the data retrieved from the Data Server.

4.5.6.4.6 Event Trace

Figure 4.5-37 shows the visualize science data event trace.



4-311

Figure 4.5-37. Q/A Visualize Data

305-CD-027-002

4.5.6.5 Q/A Metadata Update

4.5.6.5.1 Abstract

Q/A specifies metadata associated with data products to which it subscribes. This metadata describes format, amount, sampling information, etc. Q/A has the option of changing or updating the Q/A metadata associated with any data product.

4.5.6.5.2 Stimulus

Entering the Update Metadata command in the Q/A Monitor Command GUI will initiate the processing in this scenario.

4.5.6.5.3 Desired Response

The following actions will occur upon the stimulus of this scenario:

- The quality assurance metadata is updated, the metadata update will be stored with the product at the appropriate Data Server.

4.5.6.5.4 Participating Classes From the Object Model

- DpPrQaMonitor
- PIDataTypes
- PIDataType
- AdCollection
- Advertisement
- DsCIESDTReferenceCollector
- GIParameter
- GIParameterList
- DsCICommand
- DsCIRequest

4.5.6.5.5 Scenario Description

- a. The Q/A position brings up the Monitor Command Window and selects *Display Data Types for MetaData Update*.
- b. The Data Type Selection Window comes up, displaying the currently subscribed-to data types.
- c. The Q/A position selects the data type for which the Metadata is to be updated; the Data Type Selection Window goes away.
- d. The Q/A position selects to *Update Metadata* from the Monitor Command Window.
- e. The Metadata Editor Window comes up, showing the current values for the fields which are valid for the user to change. The user can update any or all these fields, followed by selecting the *Update* function.
- f. The Metadata Editor Window goes away.
- g. The Q/A Monitor constructs and submits a request to the Data Server to update the existing Metadata for the product with the new Metadata.

4.5.6.5.6 Event Trace

Figure 4.5-38 shows the update Q/A metadata event trace.

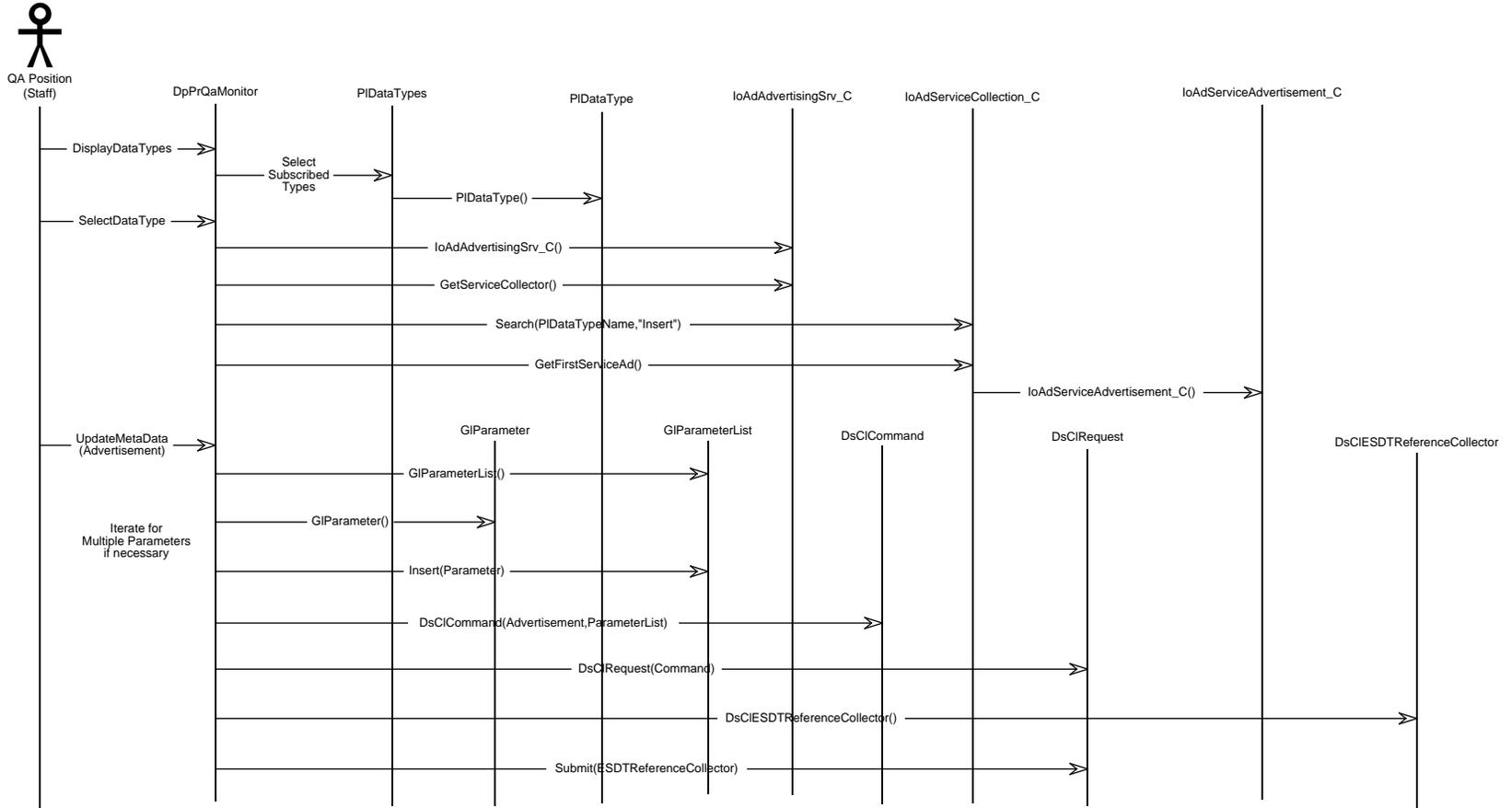


Figure 4.5-38. Q/A Update MetaData

4.5.6.6 DAAC Non-Science QA Checking

4.5.6.6.1 Abstract

DAAC non-science QA activities occur during post-production processing. After PGE execution the number of outputs generated, input and output granule sizes and output granule metadata values are evaluated. Data generated during the process provide a measure of the integrity level employed when generating products. This product information is saved. Data is destaged and resources are deallocated after completion of DAAC non-science QA activities.

4.5.6.6.2 Stimulus

DAAC non-science QA events presented in this scenario occur when Planning releases a data processing request to Processing.

4.5.6.6.3 Desired Response

The following actions will occur upon the stimulus of this scenario:

- PGE execution
- Product generation
- Request for deallocation of resources and destaging of data preceded by DAAC non-science QA activities

4.5.6.6.4 Participating Classes From the Object Model

- DpPrExecutionManager
- PIDataGranule
- DpPrDatabaseValNB
- DpPrDataManager
- PIDPRB
- DpPrPge
- DpPrPcf
- DpPrMetadataNB
- DpPrNonScienceQANB

4.5.6.6.5 Scenario Description

- a. Planning releases a DPR to Processing which triggers PGE execution.
- b. The PGE generates output granules.
- c. A request for data to be destaged is received by the Data Manager. It begins the processing the list of PGE inputs to deallocate resources and the list of PGE outputs to archive to the data server for the DPR that Processing handled. DAAC non-science QA is invoked for each input and output.

- d. For each input and output during DAAC non-science QA processing, checking is performed to verify that the size of the granule is within the pre-defined size class. Results of the size checks are generated and save as metadata.
- e. Each output undergoes size checking and is tallied to verify that the PGE produced the correct number of outputs. Then output granule metadata is evaluated. Metadata parameter checking compares PGE generated parameter values with PDPS database values defined and entered during SSIT. The results of all of these checks are saved as metadata.
- f. Upon completion of DAAC non-science QA outputs are destaged and pertinent resources are deallocated.

4.5.6.6.6 Event Trace

Figure 4.5-39 shows the DAAC non-science QA event trace.

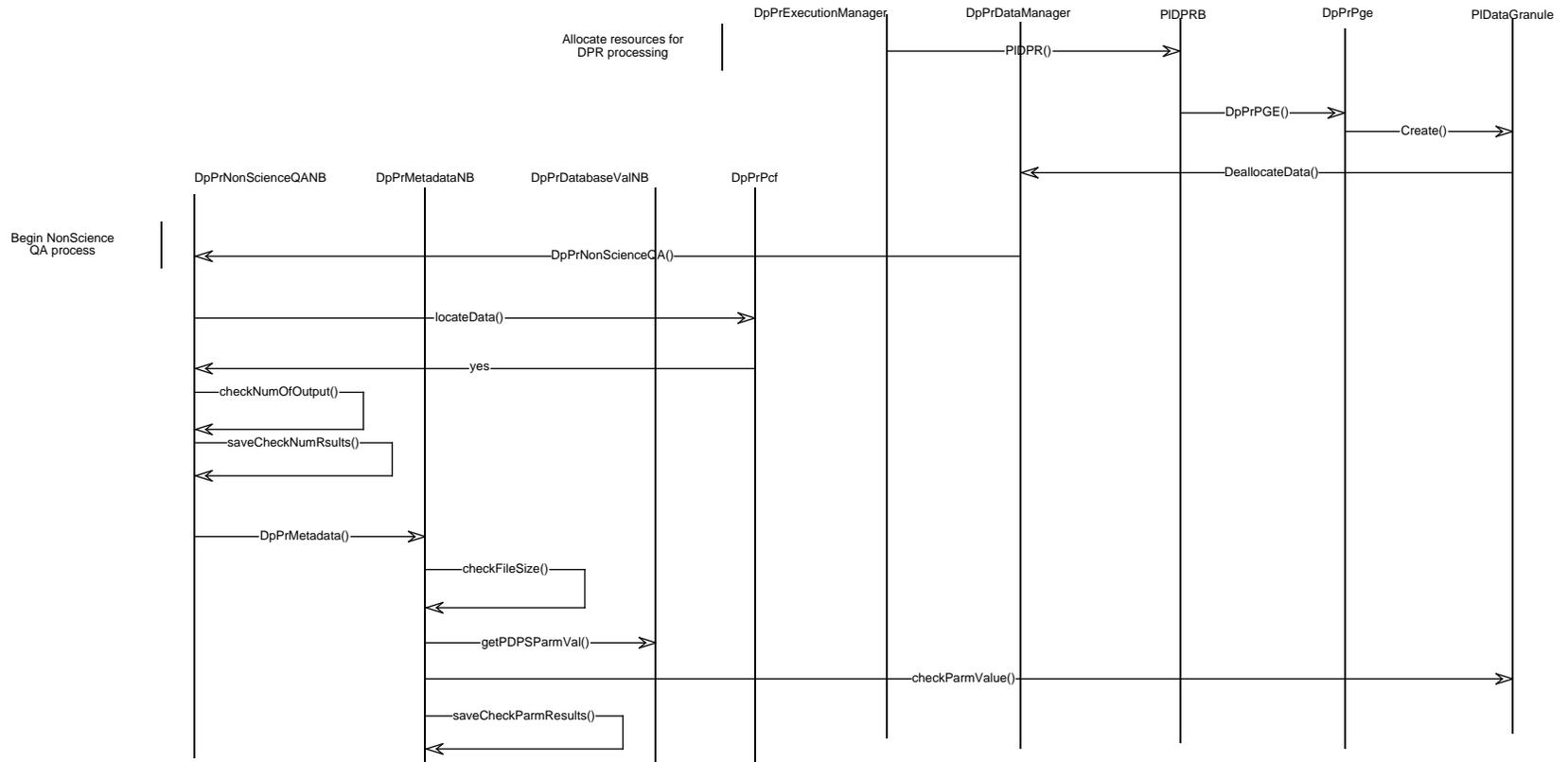


Figure 4.5-39. DAAC non-science QA

4.5.7 Data Pre-Processing Scenarios

The software used to create pre-processed data to be used by a PGE should be thought of as a PGE. This PGE will be input into the Processing CSCI through the normal mechanisms provided to the Planning CSCI. The Data Preprocessing CSC preprocesses Level 0 data sets to allow PGEs to access the data via the SDP Toolkit.

4.5.7.1 Scenario for Producing TRMM O/A Data Set

4.5.7.1.1 Abstract

Consider an example of CERES data preprocessing using SDPF-generated L0 data, and FDF-generated definitive orbit data. Level 0 CERES data received from SDPF will be received by the Ingest CSCI at the Goddard (GSFC) DAAC. The FDF-generated ephemeris data could arrive later than L0 data. Metadata are extracted for both ephemeris and L0 data and archived for later use. The attitude data along with position data are used to earth-locate each CERES footprint and calculate viewing geometry.

The ephemeris data from FDF will be in binary format as described in the FDF Generic Data Products Format ICD, 533-FDD-91/028, June 1991. There are two ways to handle the preprocessing of FDF-generated ephemeris file. Either it could be reformatted to HDF-EOS, appropriate metadata created and archived in the Data Server. If conversion to HDF is likely to degrade performance of the CERES algorithm, then it is stored in the Data Server in the original format. When a CERES Product Generation Executive (PGE) requests this ephemeris data, with information from the Planning subsystem, the data are retrieved from the Data Server, reformatted to the format of the hardware where the PGE will be executed. Appropriate metadata are prepared for the SDP Toolkit, and then staged for processing. This staged data is a new preprocessed object called "O/A Data Set".

When a CERES PGE requests L0 data, with information from the Planning subsystem, the Level 0 data and metadata are retrieved from the Data Server, and then staged for processing.

4.5.7.1.2 Stimulus

Initiation of TRMM Pre-Processing Job by AutoSys.

4.5.7.1.3 Desired Response

Generation of pre-processed TRMM ephemeris and attitude data sets.

4.5.7.1.4 Participating Classes From the Object Model

- DpPpAttitudePacket
- DpPpAttitudePackets
- DpPpAttitudeProcessingSet
- DpPpQacList
- DpPpQaParameters
- DpPrEphemerisMetadata
- DpPrEphemerisRecord

- DpPrEphemRecord
- DpPrFdfProcessingSet

4.5.7.1.5 Scenario Description

Scenario for Creating an Attitude File

- a. Identify L0 Housekeeping data sets to process
- b. Establish L0 Housekeeping data set processing order
- c. Find a boxcar averaging window number of attitude telemetry packets
- d. Check the QAC list to determine quality flagging of telemetry packets
- e. Initialize boxcar averaging of roll, pitch and yaw angles and their rates
 1. Check boxcar average to look for outliers in the angles and rates
 2. Write an attitude archive record
- f. Find the next attitude telemetry packet
- g. Check the QAC list to determine quality flagging of telemetry packets
- h. Advance boxcar
 1. Check boxcar average to look for outliers in the angles and rates
 2. Write an attitude archive record
- i. Exhaust attitude telemetry packets
- j. Check the QAC list to determine quality flagging of telemetry packets
- k. Terminate boxcar
 1. Check boxcar average to look for outliers in the angles and rates
 2. Write an attitude archive record

Scenario for Creating an Ephemeris File

- a. Identify FDF data sets to process
- b. Establish FDF data set processing order
- c. Read an FDF EPHEM format record
- d. Unpack the FDF EPHEM format record orbital position and velocity vectors
- e. Initialize boxcar averaging of the position and velocity vectors
 1. Check boxcar average to look for outliers in position and velocity vector
 2. Write ephemeris archive record
- f. Advance boxcar
 1. Check boxcar average to look for outliers in position and velocity vector
 2. Write ephemeris archive record
- g. Exhaust unpacked data points
- h. Read next FDF EPHEM record

- i. Unpack the FDF EPHEM format record orbital position and velocity vectors
- j. Check for gaps in the position and velocity vector timelines
- k. Advance boxcar
 - 1. Check boxcar average to look for outliers in position and velocity vector
 - 2. Write ephemeris archive record
- l. Exhaust FDF EPHEM format records
- m. Terminate boxcar
 - 1. Check boxcar average to look for outliers in position and velocity vector
 - 2. Write ephemeris archive record

4.5.7.1.6 Event Traces

Figure 4.5-40 shows the attitude file creation event trace. Figure 4.5-41 shows the ephemeris file creation event trace.

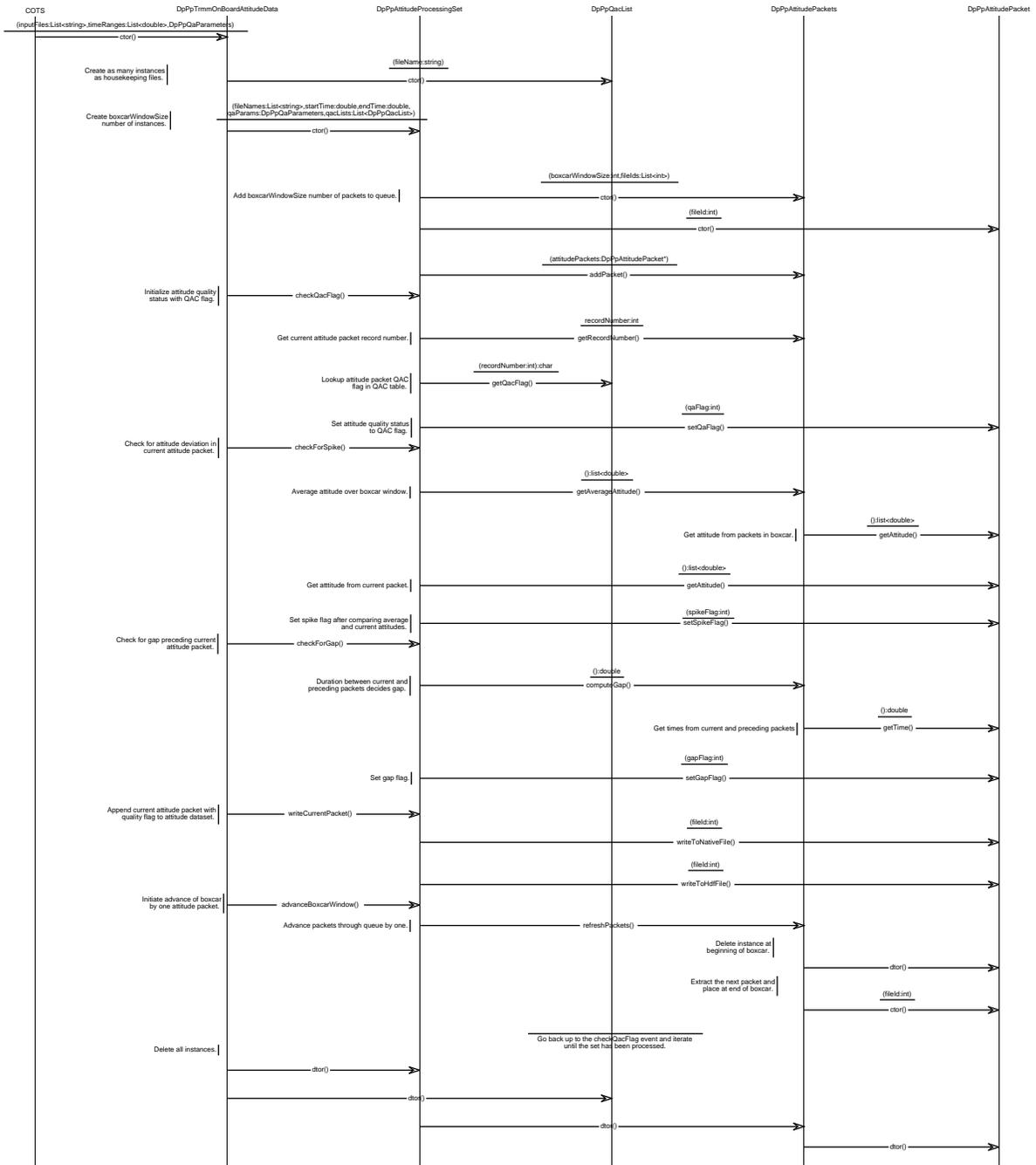
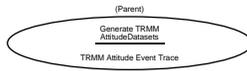


Figure 4.5-40. Creating a TRMM Attitude Data Set

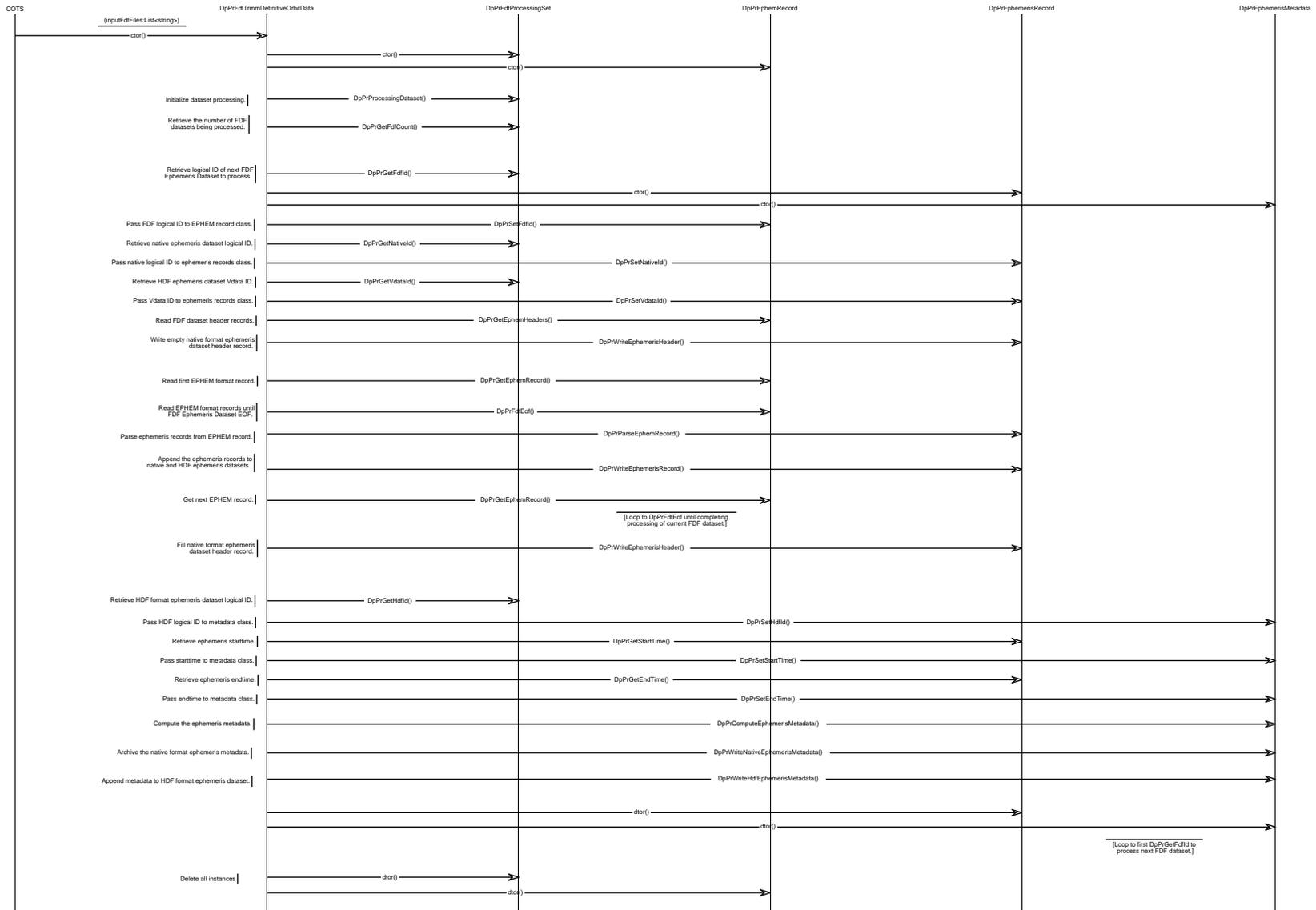


Figure 4.5-41. Creating a TRMM Ephemeric Data Set

4.5.7.2 Scenario for Producing EOS AM-1 O/A Data Set

4.5.7.2.1 Abstract

Preprocessing L0 data of AM-1 is slightly different from the preprocessing of TRMM data. Ephemeris and attitude data for AM-1 instruments are both contained in the spacecraft ancillary packets. EDOS will supply the level zero Production Data Set (PDS) containing the ancillary packets to Ingest at the GSFC DAAC. Ingest then supplies the PDS to the Planning and Data Processing subsystem where it is preprocessed by the DPREP CSC. The ephemeris and attitude data are extracted from the ancillary packets and reformatted to HDF-EOS and to a native data set for access via the SDP Toolkit. The packets are quality checked for missing data and data spikes. Metadata containing packet quality and orbit information are also created for the data set. PGEs can then access the ephemeris and attitude data through the SDP Toolkit.

4.5.7.2.2 Stimulus

Initiation of AM-1 Pre-Processing Job by AutoSys.

4.5.7.2.3 Desired Response

Generation of preprocessed AM-1 ephemeris and attitude data sets.

4.5.7.2.4 Participating Classes From the Object Model

- DpPpAm1AncillaryPacketNB
- DpPpAm1AncPacketProcessorNB
- DpPpAm1ScOaDataNB
- DpPpAttitudeDataSetNB
- DpPpCcscsPacketNB
- DpPpEdosConstructionNB
- DpPpEdosLevelZeroPDSNB
- DpPpEphemerisDataSetNB
- DpPpPacketVectorNB

4.5.7.2.5 Scenario Description

Scenario for Creating an AM-1 Attitude Data Set and an AM-1 Ephemeris Data Set

- a. Read n number of packets.
- b. Check the quality of the current packet.
 1. Check the ephemeris data for spikes by checking against n-1 of the surrounding packet's ephemeris data.
 - a. Check the magnitude of the position vector.
 - b. Check the magnitude of the velocity vector.
 2. Check the attitude data for spikes by checking against n-1 of the surrounding packet's data.

- a. Check the roll angle.
- b. Check the pitch angle.
- c. Check the yaw angle.
- 3. Check for a gap between the current packet and the next packet.
- c. Add an ephemeris record to the HDF-EOS and native data set.
- d. Add an attitude record to the HDF-EOS and native data set.
- e. If necessary, write metadata for the current orbit.
- f. Get the next packet.
- g. Exhaust all of the packets in the PDS.
- h. Report on the quality of the ephemeris data.

4.5.7.2.6 Event Traces

Figure 4.5-42 shows the ephemeris and attitude file creation event trace.

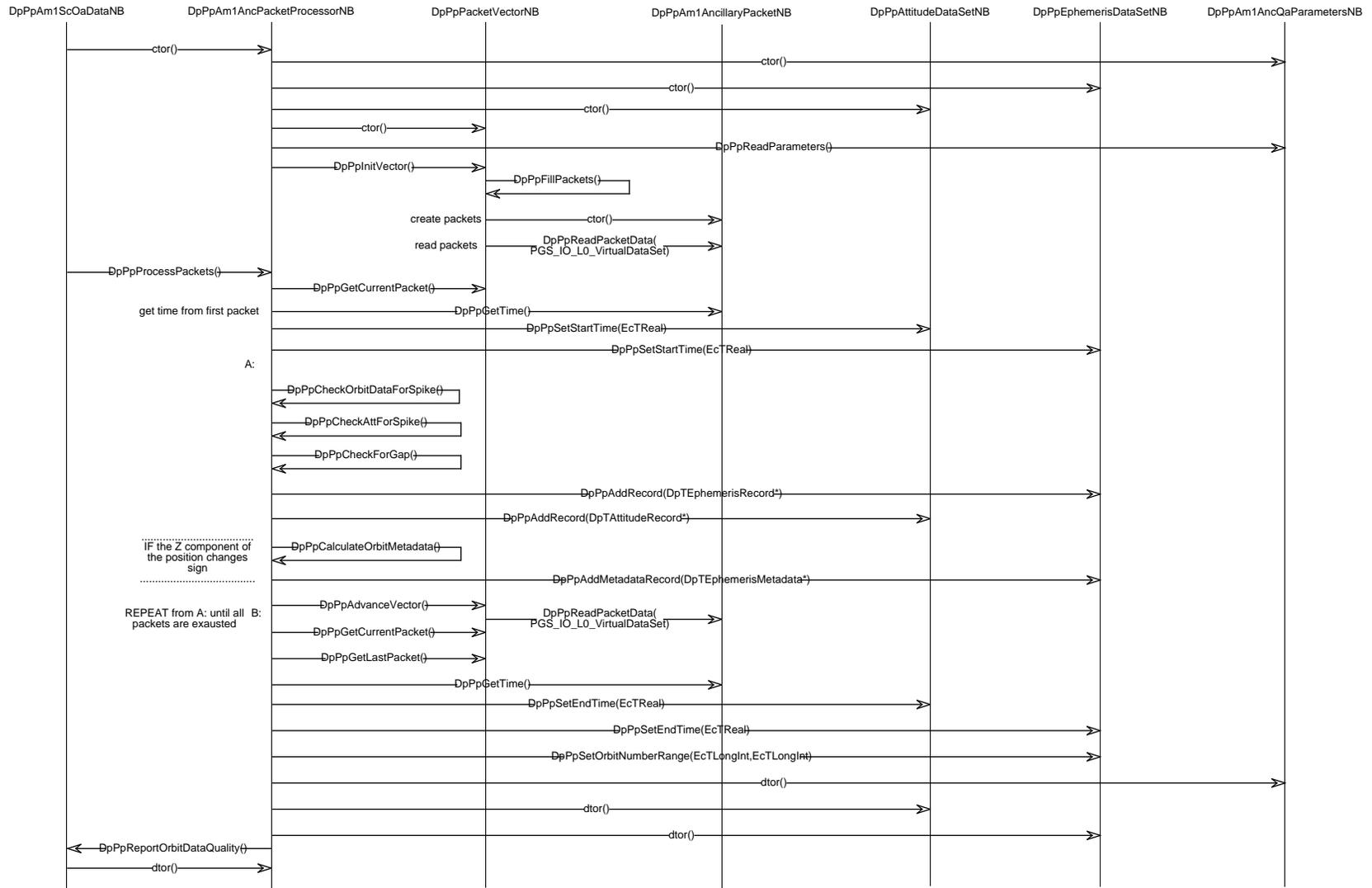


Figure 4.5-42. Creating AM-1 Ephemeris and Attitude Data Sets

4.5.8 Database Interface Scenarios

4.5.8.1 Scenario for Updating a Database Table

4.5.8.1.1 Abstract

This scenario describes the use of the database interface classes to update a table in the PDPS database.

4.5.8.1.2 Stimulus

A persistent object has been updated and needs to be stored in the PDPS database.

4.5.8.1.3 Desired Response

The PDPS database is updated to reflect the changes made to the persistent object.

4.5.8.1.4 Participating Classes From the Object Model

- DpPrDbColVal
- DpPrDbColValList
- DpPrDbConnectRecord
- DpPrDbMaster
- DpPrDbIF
- DpPrDbInterface

4.5.8.1.5 Scenario Description

Scenario for Updating a Table in the PDPS database.

- a. Update object myObject, which is an instance of persistent class myClass
- b. Create object dbIFObject by instantiating the template class DpPrDbInterface for myClass
 1. This causes a database connection to be established via DpPrDbMaster.GetConnection
 - a. the user name and password defaults to that of the caller unless overridden
 - b. a check if made to see if a connection already exists with this user id and password
 - c. if connection already exists, use it and increment use count in connection record
 - d. if not, create a connection record, which is an instance of class DpPrConnectRecord
 - e. establish the database connection
 - c. Identify the name of the database table identified with myClass
 - d. Identify the criterion limiting the rows to be updated
 1. Express the criterion as an instance of class DpPrColValList
 2. Each list member is an instance of class DpPrColVal, describing a column-value pair
 3. Only rows in which every specified column matches its specified value will be updated
 - e. Call dbIFObject.UpdateObject() for table name, myObject, and criterion.
1. myObject is interpreted as a list of column/value pairs to be updated in the table

2. Call UpdateColumns(), inherited from the parent class DpPrDbIF, passing the list in e1)
3. UpdateColumns uses DBtools to send an update command to the database
4. The number of rows updated is returned

4.5.8.1.6 Event Trace

Figure 4.5-43 shows the event trace for updating a table in the PDPS database.

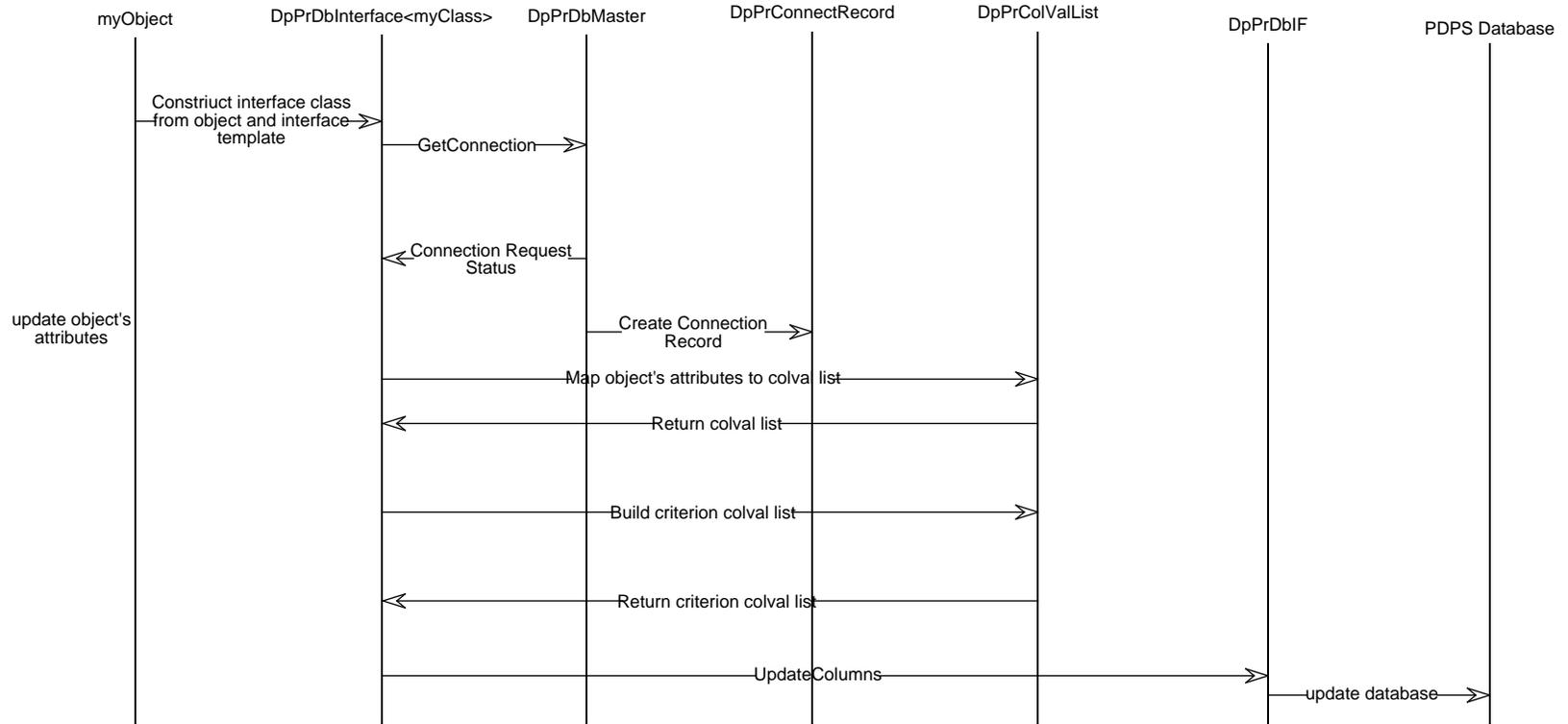


Figure 4.5-43. Update Database Table

4.6 CSCI Structure

This section provides details on the underlying software structure of the Processing CSCI. The different components of the CSCI are defined, and a brief summary of the software architecture of the Planning and Data Processing Subsystems is provided.

The software architecture for the Planning and Data Processing Subsystems can be divided into three major layers of software:

- a. PDPS User Interface Layer—contains the GUI applications used by the Operations staff to initiate Planning, Processing, and Algorithm Integration & Test activities.
- b. PDPS Application Layer—contains the functional components for Planning, Processing, and Algorithm Integration & Test. These components work in collaboration with the GUI applications to perform Operations staff activities.
- c. PDPS Persistent Data Layer—contains the persistent data structures required by Planning, Processing, and the Algorithm Integration & Test CSCIs. This data is retained within an SYBASE RDBMS. These data structures include the AutoSys Database schemas as well as the schemas required to support the Planning, Processing and Algorithm Integration & Test CSCI applications.

Figure 4.6-1 provides a diagram of the PDPS software architecture.

The Processing 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 CSCs defined for the Processing CSCI. Table 4.6-1 provides a brief description for each Processing CSC as well as a mapping to Custom or OTS software.

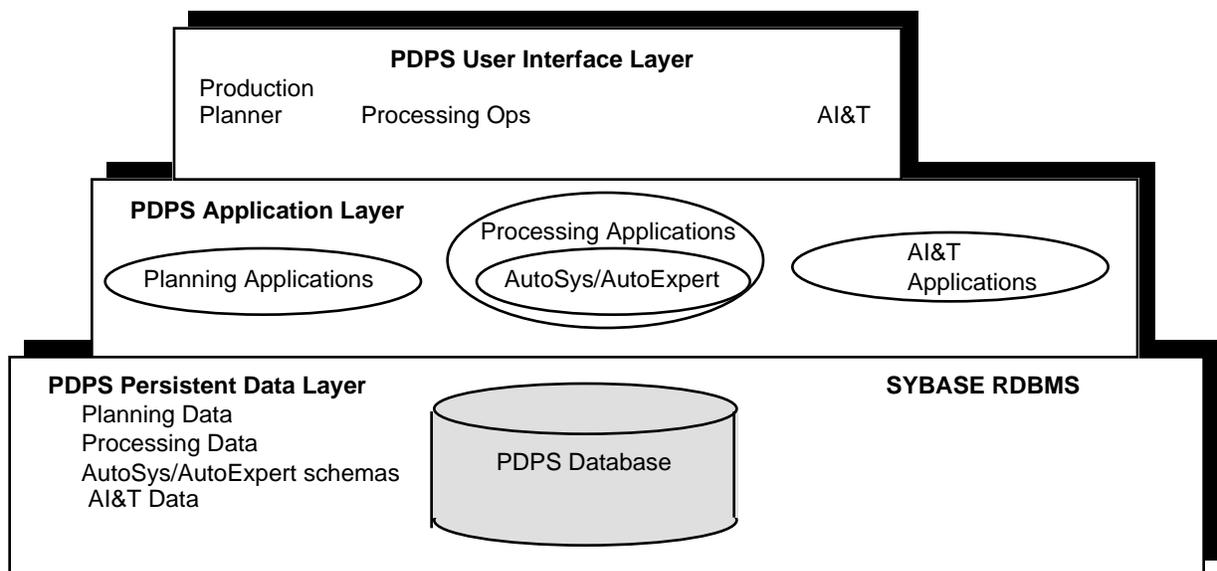


Figure 4.6-1. PDPS Software Archetecture

4.6.1 COTS CSC

4.6.1.1 Purpose and Description

The COTS CSC represents the COTS products, AutoSys and AutoXpert. A description of the components of AutoSys and the capabilities provided by AutoSys and AutoXpert are summarized in the following section.

(AUTOSYS Users Manual)

There are three primary components of AutoSys:

- a. AutoSys Database
- b. Event Processor
- c. Remote Agent

The AutoSys Database is the data repository for all system events as well as all job, monitor, and report definitions. This database is an RDBMS. For ECS, the RDBMS is SYBASE.

The Event Processor is the heart of AutoSys; it interprets and processes all the events it reads from the AutoSys Database. Sometimes called the event-daemon, the Event Processor is the program, running as a UNIX process, which actually runs AutoSys—it schedules and starts jobs. When started, the Event Processor continually scans the database for events to be processed. When it finds one, it checks whether the event satisfies the starting conditions for any job in the database. Based on this information, the Event Processor first determines what actions are to be taken, then instructs the appropriate process (or Remote Agent) to perform the actions. These actions might be the starting or stopping of jobs, checking for resources, monitoring existing jobs, or initiating corrective procedures.

The Remote Agent is a temporary process started by the Event Processor in order to perform a specific task on a remote machine. It starts the UNIX command specified for a given job, sends information about the task as event to the database, then exits. If the Remote Agent is unable to transfer the information, it waits and then tries again.

To support fault tolerance, AutoSys provides a high availability option which consists of a shadow database server plus a shadow event processor. In the event of a failure of the primary AutoSys Database or Event Processor, the shadow AutoSys Database or Event processor will take over its assigned role.

AutoSys is completely event-driven; i.e., to become activated by the Event Processor, an event must occur for a job. For example, the starting data and time have arrived, a prerequisite job has been completed, or a prerequisite file has been received. These events may come from a number of sources, such as:

- a. Jobs changing states, such as starting, finishing successfully, finishing unsuccessfully, etc.
- b. Data and Time.
- c. Internal AutoSys verification agents, such as detected errors.
- d. Events sent with the sendevent command (AutoSys API) - either from the command line, or from user applications.

Table 4.6-1. Processing CSCI Components

CSC	Description	Type (Custom=DEV; off-the-shelf=OTS)
COTS	This CSC is used to represent the COTS products, AutoSys and AutoXpert. AutoSys has three major components; AutoSys Database, Event Processor, and the AutoSys Remote Agent. AutoSys is responsible for providing job management functions for the Processing CSCI. Also, AutoSys provides two GUIs; the Operator Console and AutoXpert. See section 4.1.3 for more information on AutoSys and AutoXpert capabilities.	OTS
COTS Management	Provides services required to interface with the COTS product, AutoSys. All unique AutoSys command-line interfaces and APIs are encapsulated into this collections of classes. Provides the ability to create data staging jobs predictively based on when the PGE requiring the data will be executed.	DEV
Resource Management	Provides services for the management of Science Processing Hardware resources. A low-level component used by the PGE Execution Management and Data Management CSCs to allocate, deallocate, and monitor the using of Science Processing Hardware resources.	DEV
Data Management	Provides services to manage the data required to support the execution of a PGE. Initiates non-science QA processing which utilizes PGE inputs and outputs prior to data destaging and resource deallocation.	DEV
PGE Execution Management	Provides services required to prepare a PGE for execution and perform post-processing activities.	DEV
Data Pre-Processing	Provides data pre-processing services to prepare ephemeris, O/A, and other ancillary data to be used by the PGE.	DEV
Quality Assurance Monitor Interface	The HMI used for performing DAAC manual quality assurance activities.	DEV

The following diagram, pictured in Figure 4-6.2 provides a sample scenario which describes the interactions between the three primary components of AutoSys; the AutoSys Database, the Event Processor, and the Remote Agent. This explains the steps taken to initiate a job which is defined in AutoSys. The scenario steps are the following:

- a. From the RDBMS, the Event Processor reads a new event - a "start job" whose start time has arrived. It reads the appropriate job definition from the database, and based on that definition, determines what action to take (e.g. run the associated command line on Science Processor).
- b. The Event Processor communicates with the Remote Agent on the Science Processor. As the Remote Agent receives the instructions from the Event Processor, the connection between the two processes is dropped. Therefore, even if the Event Processor disappears the remote machine is unaffected.
- c. The Remote Agent performs such resource checks, such as ensuring that the minimum specified number of processes are available, then "forks" a child process, which will actually run the specified command.
- d. The UNIX command completes and exits. The Remote Agent captures this exit code.
- e. The Remote Agent communicates the event (exit code, status, etc.) directly to the RDBMS. If the RDBMS is unavailable for any reason, the Remote Agent will go into a wait/resend cycle until the message is delivered.

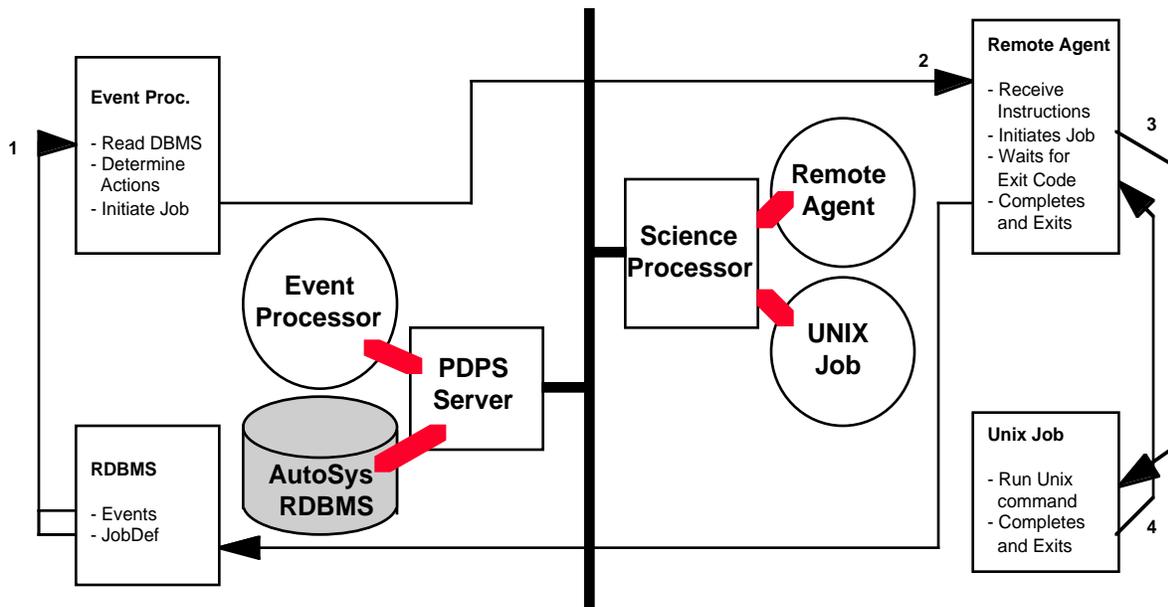


Figure 4.6-2. AutoSys Sample Scenario

Only two AutoSys processes need to be running; i.e., the Event Processor and the RDBMS. When these two components are running, AutoSys is fully operational. The Remote Agent is started on a remote machine on an "as needed" basis. As soon as it completes its assigned task, it exits. Please note that the Remote Agent gets started on the remote machine by the Event Processor talking to the internet daemon (INETD) on the remote machine.

For details on the capabilities on AutoXpert, please see Section 4.1.4.3, Platinum Technology's AutoXpert. In terms of the PDPS Preliminary Design Specification, AutoXpert provides functions which were previously mapped to the Production Management CSC in the Planning CSCI. Since these functions are being provided by the AutoSys and AutoXpert products, it seemed desirable to map the COTS product to one CSCI. This decision eliminated the need to represent interfaces between AutoSys and AutoXpert across CSCI and Subsystem boundaries.

4.6.2 COTS Management CSC

4.6.2.1 Purpose and Description

The COTS Manager is a collection of classes used to interface with AutoSys. This is a part of the Production Planning Workbench application. This collection of classes encapsulates the unique AutoSys command-line interfaces and APIs and provides operations required by Planning to create, modify, cancel, release, suspend, resume, and request status of jobs which exist in the AutoSys Database. Operations are also provided to notify AutoSys of an external events such as resource fault information.

4.6.2.2 Object Model Mapping

Please refer to Table 4.6-2.

4.6.2.3 Candidate Products

There are no candidate products for this CSC. This is a custom component required to interface with ECS specific services.

4.6.3 Resource Management CSC

4.6.3.1 Purpose and Description

The Resource Management CSC provides the services required for managing the Science Processing Hardware resources used for the generation of data products. This CSC manages the storage devices and computers which comprise the Science Processing Hardware resources. These Services support the allocation and deallocation of resources required for the execution of a PGE. Also provided are operations to initialize a logical mapping of the Science Processing Hardware resources. Currently, AutoSys provides some resource management capabilities. These services will augment the capabilities of AutoSys.

4.6.3.2 Object Model Mapping

Please refer to Table 4.6-2.

4.6.3.3 Candidate Products

As stated before, AutoSys provides some limited resource management capabilities. ECS is committed to working with the vendor to influence the future direction of their product to meet the needs of the ECS science processing environment management.

4.6.4 Data Management CSC

4.6.4.1 Purpose and Description

This application manages the staging, destaging, and retention of data on Science Processing Hardware resources. The interface to the Science Data Server CSCI to stage (acquire) and destage (insert) data is provided by this application. This is a distinct application which is initiated as precursor job for a PGE. This job would be initiated by AutoSys when the dependencies defined for this job have been met. This application interfaces with the PDPS Database to retain persistent data on what data currently resides on Science Processing Hardware Resources.

4.6.4.2 Object Model Mapping

Please refer to Table 4.6-2.

4.6.4.3 Candidate Products

There are no candidate products for this CSC. This is a custom component required to interface with ECS specific services.

4.6.5 PGE Execution Management CSC

4.6.5.1 Purpose and Description

This application performs preparation and post-processing activities to support the execution of the PGE. These activities include the following:

- a. Staging the executables and binaries which define the PGE, if required.
- b. Creating the Process Control File (PCF) used to support the PGE. PCF contains input and output data information.
- c. Preparing the Production History File to be destaged (inserted) into the Science Data Service CSCI with the science data products. The Production History File will contain information used as inputs to the PGE (the Data Processing Request information) as well as resource utilization information.
- d. Deallocation of resources at the completion of data destaging.

4.6.5.2 Object Mapping

Please refer to Table 4.6-2.

4.6.5.3 Candidate Products

There are no candidate products for this CSC. This is a custom component required to interface with ECS specific services.

4.6.6 Mapping of objects to CSCs

Table 4.6-2 - Mapping of objects to CSCs

Table 4.6-2. Mapping of objects to CSCs (1 of 3)

Object	CSC
DpPpAm1AncPacketProcessorNB	Data PreProcessing
DpPpAm1AncQaParametersNB	Data PreProcessing
DpPpAm1AncillaryPacketNB	Data PreProcessing
DpPpAm1ScOaDataNB	Data PreProcessing
DpPpAttitudeDataSetNB	Data PreProcessing
DpPpCcsdsPacketNB	Data PreProcessing
DpPpEphemerisDataSetNB	Data PreProcessing
DpPpPacketVectorNB	Data PreProcessing
DpPpAm1AncPacketProcessorNB	Data PreProcessing
DpPpAm1ScOaDataNB	Data PreProcessing
DpPpEdosLevelZeroPDSNB	Data PreProcessing
DpPpEdosPDSConstructionRecordNB	Data PreProcessing
DpPrEphemRecord	Data PreProcessing
DpPrEphemerisMetadata	Data PreProcessing
DpPrEphemerisRecord	Data PreProcessing
DpPrFdfProcessingSet	Data PreProcessing
DpPrFdfTrmmDefinitiveOrbitData	Data PreProcessing
DpPpEphemerisData	Data PreProcessing
DpPpFdfData	Data PreProcessing
DpPpFdfTrmmDefinitiveOrbitData	Data PreProcessing
DpPpLevelZeroData	Data PreProcessing
DpPpPreprocessingData	Data PreProcessing
DpPpSdpfLevelZeroDatasetFile	Data PreProcessing
DpPpSdpfLevelZeroProductionData	Data PreProcessing
DpPpSdpfLevelZeroSfduFile	Data PreProcessing
DpPpTrmmOnBoardAttitudeData	Data PreProcessing
DpPpTrmmScAncillaryData	Data PreProcessing
DpPpTrmmScOaData	Data PreProcessing
DpPpAttitudePacket	Data PreProcessing
DpPpAttitudePackets	Data PreProcessing
DpPpAttitudeProcessingSet	Data PreProcessing
DpPpQaParameters	Data PreProcessing
DpPpQacList	Data PreProcessing
DpPpTrmmOnBoardAttitudeData	Data PreProcessing
DpPrDataManager	Data Management
DpPrDataMap	Data Management
DpPrDprStatusNB	Data Management
DpPrJobManagement	Data Management
DpPrResourceManagement	Data Management
DpPrUnusedData	Data Management
DsCICommand	Data Management
DsCIESDTReferenceCollector	Data Management
DsCIRequest	Data Management
GICallBack	Data Management
PIDPRB	Data Management
PIDataGranule	Data Management
PIPerformance	Data Management

Table 4.6-2. Mapping of objects to CSCs (2 of 3)

Object	CSC
COTS	COTS Manager
DpPrCotsManager	COTS Manager
DpPrDataManagement	COTS Manager
DpPrDprStatusNB	COTS Manager
DpPrJIL	COTS Manager
DpPrPgeExecutionManagement	COTS Manager
DpPrScheduler	COTS Manager
PIDPRB	COTS Manager
PIGroundEvent	COTS Manager
PIPerformance	COTS Manager
PIPge	COTS Manager
DpPrDataManager	QA Monitor
DpPrDatabaseValNB	QA Monitor
DpPrExecutionManager	QA Monitor
DpPrMetadataB	QA Monitor
DpPrNonScienceQANB	QA Monitor
DpPrPCF	QA Monitor
DpPrPGE	QA Monitor
PIDPRB	QA Monitor
PIDataGranule	QA Monitor
PIDataTypeB	QA Monitor
PIPGE	QA Monitor
DpPrQaMonitor	QA Monitor
DsCICommand	QA Monitor
DsCIESDTReference	QA Monitor
DsCIESDTReferenceCollector	QA Monitor
DsCIRequest	QA Monitor
DsCISubscription	QA Monitor
EOSVIEW	QA Monitor
GIPparameter	QA Monitor
GIPparameterList	QA Monitor
GIUR	QA Monitor
IoAdAdvertisingSrv_C	QA Monitor
IoAdServiceAdvertisement	QA Monitor
IoAdServiceCollection_C	QA Monitor
PIDataGranule	QA Monitor
PIDataType	QA Monitor
PIDataTypes	QA Monitor
DpPrComputer	Resource Management
DpPrDiskAllocation	Resource Management
DpPrDiskPartition	Resource Management
DpPrResource	Resource Management
DpPrResourceConfiguration	Resource Management
DpPrResourceManager	Resource Management
DpPrString	Resource Management
MsDAAC	Resource Management
MsManager	Resource Management
MsMgCallbacks	Resource Management
PIResourceUI	Resource Management
DpPrDprStatusNB	Execution Management

Table 4.6-2. Mapping of objects to CSCs (3 of 3)

Object	CSC
DpPrExecutable	Execution Management
DpPrExecutionManager	Execution Management
DpPrPcf	Execution Management
DpPrPge	Execution Management
DpPrResourceManagement	Execution Management
DpPrResourceUsageNB	Execution Management
DsCICommand	Execution Management
DsCIESDTRerenceCollector	Execution Management
DsCIRequest	Execution Management
MsBaCotsIFB	Execution Management
MsManager	Execution Management
MsMgCallBacks	Execution Management
PIDPRB	Execution Management
PIDataGranule	Execution Management

4.7 Processing CI Management and Operations

The following sections discuss the management and operation of the Processing CSCI, addressing how the CSCI is managed at the local and system levels. The Processing CSCI in relation to the system management strategy is discussed in Section 4.7.1. The approach to the development of operator interfaces for the Processing CSCI is then described in Section 4.7.2. Finally, the approach to reporting for the Processing CSCI is described in Section 4.7.3. This section addresses the operations and management activities as they relate to the Processing CSCI of the Processing Subsystem. The other major software component of the Subsystem, the AITTL CSCI, is discussed in Section 7.6.

The role of Production Management is discussed by providing information on the following topics:

- Operations activity level
- Interaction with Planning CSCI
- Use of AutoSys and AutoXpert for queue management and monitoring
- Resource Management
- QA Monitoring

Use of AutoSys and AutoXpert

A significant management concept for the Processing CSCI is the use of the AutoSys and AutoXpert job scheduling and monitoring software packages as a central component of the subsystem. These tools provide significant capabilities for the scheduling, execution, and monitoring of the production processing workload. The tools provide a reliable, off-the-shelf mechanism to define the processing tasks in an extremely flexible fashion, meeting all of the needs of the science software for controlling complex processing flows from one executable to another within a PGE. Capabilities such as AutoSys' 'Job Boxes' provide the needed mechanisms for job dependency definition. This capability to address complex job dependencies insures that DAAC management has the ability, now and for future instruments, to support the data processing schemes of science software developers. Additionally, these tools provide for job error handling,

logging of job status and some resource management capabilities, which are extended as required by ECS custom software.

The operator interface provided by AutoXpert provides three views of the processing activities: the Timeline view depicting the development of processing over time, the Job Network view depicting the relationships between jobs and job boxes, and the Resource view which depicts the allocation of processing activities to processors within the system. Note that the AutoXpert provides the capability to perform limited 'what-if' capabilities to assess the impact of delayed processing on downstream activities. In addition, within AutoSys, the operators are able to modify job characteristics of individual jobs, such as the modification of priorities associated with a job. These tools provide significant capabilities for operations to manage and control processing activities within the ECS.

Operations Activity Level

A key design consideration for the Processing CSCI is that the operations personnel will interact with individual science data processing jobs on an exception basis only. No interactions are required of the operations personnel for normal operations, except for those required by the science software itself. No operations actions are required to start or stop a job, or to stage or destage data files. Operations personnel are provided with job monitoring tools through the AutoXpert system allowing them to observe the progress of the job through the system. Operations personnel are expected to monitor processing for anomalies that may require their attention. The AutoXpert tool provides alerts to operations for predefined conditions, such as processing jobs that have run beyond the expected end time.

Interaction with the Planning CSCI

The Processing CSCI is matched with the Planning CSCI to manage and control the significant science data processing activities allocated to the DAACs. The preparation or definition of processing activities to be accomplished is performed by the Planning CSCI. When the planning for the processing tasks has been completed and the necessary data has arrived to initiate the processing, the Planning CSCI releases the processing task to the Processing CSCI for subsequent management through the processing activities. This cooperative management of the planning and execution activities is a key feature of the management of science data processing.

Resource Management

The Processing CSCI provides resource management capabilities to control the allocation of processing tasks to processors. This capability allocates and deallocates processing resources as required for PGE Execution Management and Data Management CSCs. This capability maintains information on the processors allocated to the Processing CSCI for use in the management of science data processing. As tasks terminate and processors become available for other activities, the resource management function keeps track of the state of the processors. The Processing CSCI monitors the use of resources by the PGEs during their execution. This information is used to project the performance of the PGEs for the purpose of planning future processing.

QA Monitoring

The Processing CSCI includes tools to support quality assurance monitoring of the products generated during science processing. These capabilities include subscription services to access the data files that are to be quality checked, data visualization tools, and interfaces to update metadata

for the products with QA information. This collection of utilities are a part of the essential items needed to manage the data quality checking operations that are an aspect of the end-to-end science data processing activity.

4.7.1 Processing CSCI and the System Management Strategy

The system management strategy as supported by the Processing CSCI is discussed in the following paragraphs.

4.7.1.1 System Management and Operations Philosophy

A primary design consideration for the Processing CSCI from the point of view of system management and operations is that operator interactions with Processing be simplified and automated as much as possible. In addition, the Processing CSCI provides operations with the needed flexibility to respond to unexpected tasks as they arise. The objectives of simplification and flexibility are attained, in large part, through the use of the AutoSys/AutoXpert scheduling tool.

4.7.1.2 Processing CSCI and the System

The following paragraphs discuss key features of the Processing CSCI in relation to system management.

Resource Management and MSS

The Processing CSCI includes capabilities to control the processing resources available to it for science processing. Configuration information received from MSS is used to define the resources available. The current schedule of ground events (e.g., hardware maintenance activities) is also used as a part of the job scheduling process by reserving particular processors for ground events. The Processing CSCI then participates with the MSS in the management of the system processing resources to provide significant flexibility for resource management.

Data Staging and Destaging

The Processing CSCI, in conjunction with the Planning CSCI, provides for management of science data files between related processing jobs. In the general case, a science processing PGE requires several data files, which must be staged from the Data Server/Ingest prior to job execution. However, most processing jobs require data files that are the product of another PGE. For example, a Level 1a processing job might produce a Level 1a data file that is used as input to a Level 2 processing job. The CSCI attempts to manage processing jobs so that it does not need to stage input files from the Data Server if the files have recently been produced and reside on processor local disks. If a job does need input files to be staged from the data server to local disks, the predictive staging scheme is employed to start the file staging at an appropriate time. This eliminates needless delays for data file staging. By providing this broader view of data file management, the Processing CSCI contributes to improved systems management efficiency.

Enterprise Management and Processing

As with the other components of the ECS, the Processing CSCI contributes to the enterprise management approach to system management. The Processing CSCI interacts with the MSS for startup and shutdown procedures. The Processing CSCI uses MSS provided services for the detection and management of faults. It also cooperates with MSS for other enterprise capabilities including logging of events and providing system management information, such as accounting

information. The Processing CSCI also receives information from the MSS for resource management, as discussed earlier. In this way the Processing CSCI participates with the other ECS CSCIs to provide a systematic approach to system management.

PGE Interactions

To include the PGEs into the system management approach, the Processing CSCI provides access to the PGE for management of the PGE execution. The PGE Process Control File that is used by the PGE to receive control information is constructed by the Processing CSCI based upon directions received from the science software developers at the AI&T event. The Processing CSCI also provides an interface to the Data Server staging the PGE if necessary. This scheme incorporates the PGE into the system management scheme.

4.7.2 Operator Interfaces

This subsection describes the operator user interfaces provided by the Processing CSCI to operations personnel. A general description of the framework and methodology employed for the development of these interfaces can be found in Section 4.7 of the Detailed Design Overview (305-CD-020-002). This subsection augments that information with additional design information which is specific to the Processing CSCI.

The operator user interfaces for the science data production environment are COTS provided interfaces. In addition, the operator user interface for the DAAC Quality Assurance operations position will be a ECS custom application. This custom graphical interface will be created with the aid of the 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 development 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 Processing CSCI specific code.

4.7.2.1 Off-The-Shelf Interfaces

The Job Scheduling COTS products, AutoSys and AutoXpert, provide GUIs to interface with their applications. The AutoSys GUI, known as the Operator Console, provides capabilities to manage and monitor a schedule of jobs. The GUI provides visibility to the job stream as well as supporting alarm and monitoring mechanisms. Also provided is a set of job interfaces which allow the operator to interact with a job. These interfaces support job creation, modification, cancellation, suspension, and modification. More information on the AutoSys product can be found in Section 4.1.4 and the underlying subsections.

AutoXpert is another GUI which used in conjunction with AutoSys can provide three different abstract views of the current production schedule. These three graphical views are called TimeScape, JobScape, and HostScape. All three views offer intuitive GUI controls and full-color to illustrate job status and activity in real-time/projection and simulation modes

TimeScape graphically displays the timing and duration of currently running jobs. Jobs are displayed in a Gantt style ribbon graph that shows both starting and ending times. This type of representation gives you a comprehensive view of how job dependencies affect job duration and how modifications to the current stream of events will affect overall job execution. TimeScape provides real-time monitoring, projection, and simulation. In real-time/projection mode,

TimeScape shows real-time execution alongside projections based on past runs. Projections compare the current run to previous runs and provide a prediction of how current events will affect the future. In addition, the duration of a job can be modified and the downstream impact to the job schedule can be determined. Also, the completion status of a job can be changed to determine the impact on dependent jobs. In simulation mode, an entire schedule of jobs can be sped up to determine how the schedule will be followed.

JobScape gives a detailed representation of job flow from a logical, or dependency, point of view. It depicts jobs in a flow diagram, showing the starting conditions for a each job. The flow of the job stream can be followed visually, step by step. Simulations in JobScape also provide a valuable means for testing jobs before they are activated. By running in simulation mode, the job definition can be checked for errors.

HostScape provides a view of jobs that are currently executing and the resources being used to execute the job. HostScape continually verifies whether AutoSys can start jobs on each machine and notifies the console if a machine is unavailable.

Each of these views can assist the operations staff in providing production management oversight. The capabilities of viewing the job stream abstractly were originally mapped to the Production Management CSC of the Planning CSCI. Because of the capabilities of AutoXpert to provide this capability, these capabilities are now mapped to the Processing CSCI. Additional capabilities to perform what-ifs on the daily job schedule are also provided. These capabilities will be helpful in determining the downstream affects that production anomalies, i.e., a job running longer than predicted or a job failure, would have on the job schedule.

More information on the AutoSys and AutoXpert products can be found in Section 4.1.4 and the underlying subsections.

4.7.2.2 Processing CSCI User Interfaces

This section is intended to describe the data that may be displayed for the operations of the Processing CSCI's applications. The exact definition of the GUI will be decided by requesting user suggestions and through demonstrating prototypes.

Quality Assurance Monitor

The Quality Assurance Monitor is a utility which provides the operations staff with the capability to perform quality assurance activities which are defined as data visualization, updating quality assurance metadata, and subscribing to data. The GUI for the Quality Assurance Monitor will be constructed with custom code which interfaces to the PDPS Database for storage of outstanding quality assurance subscriptions. The subscription editor would contain a single view which would allow an operator to enter product subscriptions to the Science Data Server. Two lists could be provided: one list containing unsubscribed products and the second list containing subscribed products. Functions would be provided to assist the operator in adding or deleting subscriptions to these products.

As part of the GUI, utilities would be provided to initiate data visualization tools, such as EOSVIEW, which will be used to visualize a science data product and to update quality assurance metadata for a data product. The update of the quality assurance metadata would occur through the use of public classes provided by the Data Server.

4.7.3 Reports

A variety of ad-hoc and canned reports will be available to the DAAC operations staff to assist in the monitoring of the activities associated with the Processing CSCI. These reports are readily accessible given that the Processing 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:

- a. 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, etc.
- b. Processing Management Reports—These reports will provide the operations staff information on Processing application software events which have occurred. This information will be available from the MSS database.
- c. Job Status/Event Reports—These reports will provide information on the history of the AutoSys job schedule. All status and event changes for a job will be logged in the AutoSys Database. Any information associated with a job can be reported on.
- d. Resource Allocation Report—These reports will track the ability of the Processing CSCI Resource Management CSC to effectively manage the Science Processing Hardware resources.
- e. Quality Assurance Report—These reports will be used to track what products have been Q/A'ed vs. the products awaiting Q/A.
- f. DPR Job Status Report—These reports will capture the current state of all the jobs which have been created to support the processing of one PGE.
- g. PGE Resource Profile Report—These reports will capture information on the actual usage of resources which have been used by a PGE. This information will be fed back into the PDPS Database to update the current PGE Resource Profile.

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 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.

This page intentionally left blank.