S4PM 5.15.0 Design Document

A design document for NASA's open source Simple, Scalable, Script-Based, Science Processor for Measurements (S4PM)

August 2006

Document Version 1.0.0

Stephen W. Berrick, NASA

Table of Contents

TABLE OF CONTENTS 2

1. INTRODUCTION 5

1.1 GOALS OF S4PM 5
1.2 FUTURE DIRECTIONS 6

2. RELATED DOCUMENTATION 7

3. DESIGN AND IMPLEMENTATION PRINCIPLES 8

3.1 SIMPLICITY 8
3.2 THE 80/20 RULE 8
3.3 "USE THE OS, LUKE!" 8
3.4 DESIGN-FOR-TROUBLE 8
3.5 TRANSPARENCY 8
3.6 KEEP THINGS TOGETHER 8

4. S4PM ARCHITECTURE OVERVIEW 9

4.1 OVERALL REQUIREMENTS 9
4.1.1 EVOLUTION AWAY FROM ECS 10
4.2 ARCHITECTURAL DESCRIPTION 11
4.2.1 FLAVORS OF S4PM 11
4.2.2 STANDARD PROCESSING ARCHITECTURE IN ECS 13
4.2.3 ON-DEMAND PROCESSING ARCHITECTURE 16
4.2.4 STANDARD PROCESSING ARCHITECTURE IN S4PA 20

5. S4PM STATIONS 22

5.1 SUBSCRIPTION NOTIFY (SUB_NOTIFY) 24
5.2 SPLIT SERVICES (SPLIT_SERVICES) 26
5.2.1 PROCESSING THE REQUEST_DATA WORK ORDER 26
5.2.2 PROCESSING THE TRACK_REQUEST WORK ORDER 27
5.2.3 PROCESSING THE FIND WORK ORDER 27
5.2.4 PROCESSING THE EXPECT WORK ORDER 28
5.2.5 THE ORDER_FAILURE WORK ORDER 28
5.2.6 THE PSPEC FILE 28
5.2.7 ADDITIONAL FEATURES 29
5.3 POLL PDR (POLL_PDR) 30
5.5 REQUEST DATA (REQUEST_DATA) 32
5.5.1 STANDARD PROCESSING 33
5.5.2 ON-DEMAND PROCESSING 33

5.5.3 DATA MINING PROCESSING 33
5.6 AUTO REQUEST (AUTO_REQUEST) 34
THE OUTPUT WORK ORDER IS EITHER A REQUEST_DATA WORK ORDER GOING TO THE REQUEST DATA STATION (IF IN ECS) OR AN ACQUIRE_DATA WORK ORDER GOING TO THE ACQUIRE DATA STATION. **34**
5.7 POLL DATA (POLL_DATA) 35
5.8 RECEIVE DN (RECEIVE_DN) 37
5.9 REGISTER DATA (REGISTER_DATA) 38
5.10 SELECT DATA (SELECT_DATA) 40
5.10.1 PRODUCTION RULES 41
5.10.2 PROXY DATA TYPES 44
5.11 TRACK DATA (TRACK_DATA) 45
5.12 FIND DATA (FIND_DATA) 47
5.12.1 STANDARD PROCESSING 47
5.12.2 ON-DEMAND PROCESSING 49
5.13 PREPARE RUN (PREPARE_RUN) 50
5.13.1 STANDARD PROCESSING 50
5.13.2 ON-DEMAND PROCESSING 51
5.14 ALLOCATE DISK (ALLOC_DISK) 52
5.14.1 ON-DEMAND STRINGS 53
5.15.1 ON-DEMAND PROCESSING 55
5.16 TRACK REQUESTS (TRACK_REQUESTS) 57
5.17 SWEEP DATA (SWEEP_DATA) 60
5.18 REGISTER LOCAL DATA (REGISTER_LOCAL_DATA) 61
5.19 EXPORT (EXPORT) 62
5.20 INSERT DATAPOOL (INSERT_DATAPOOL) 64
5.21 SHIP DATA (SHIP_DATA) 65
5.22 RECEIVE PAN (RECEIVE_PAN) 66
5.23 REPEAT HOURLY (REPEAT_HOURLY) AND REPEAT DAILY (REPEAT_DAILY) 68

6. S4PM WORK ORDERS 69

6.1 WORK ORDER NAMES 69
6.2 WORK ORDER FORMATS 70
6.2.1 PDR FORMAT 70
6.2.2 PCF FORMAT 81
6.2.3 PATHNAME LIST FORMAT 86
6.2.4 E-MAIL FORMAT 88
6.2.5 ODL FORMAT 90

7. CROSS-STATION DEPENDENCIES 91

7.1 EXPORT AND RECEIVE PAN 91
7.2 ALLOCATE DISK, REQUEST DATA, AND SWEEP DATA 91
7.3 REQUEST DATA AND REGISTER DATA 91

8. USER INTERFACES 93

8.1 S4PM MONITOR **93**
8.2 JOB MONITOR **94**
8.2.1 FAILURE HANDLERS 95
8.2.2 MANUAL OVERRIDES 97
8.3 COMPOSE DATA REQUEST TOOL **97**
8.4 VIEW/DELETE DATA TOOL **98**
8.5 S4PM ADMINISTRATION TOOL **99**
8.6 VIEW DISK ALLOCATE AND USAGE **101**

9. S4PM LOGGING **102**

9.1 PURPOSE **102**
9.2 LOGGING DETAILS **102**
9.2.1 CHAIN LOGS 102
9.2.2 FIND DATA LOG 104
9.2.3 TRACK DATA TRANSACTION LOG 105
9.2.4 CASE-BASED REASONING LOG 105

1. Introduction

This document describes the design characteristics of S4PM, version 5.15.0. For information on installing and configuring S4PM, please read the S4PM 5.15.0 Installation and Configuration Guide. For information on operating S4PM, please read the S4PM 5.15.0 Operations Guide. This document assumes that S4PM has already been properly installed and configured.

The Simple, Scalable, Script-based Science Processor for Measurements (S4PM) is a NASA developed system for highly automated processing of science data. S4PM is the main processing engine at the Goddard Earth Sciences Data and Information Services Center (GES DISC). In addition to being scalable up to large processing systems such as the GES DISC, it is also scalable down to small, special-purpose processing strings.

S4PM consists of two main parts: the kernel is the Simple, Scalable, Script-based Science Processor (S4P), an engine, toolkit and graphical monitor for automating script-based, data-driven processing. The S4PM system is built on top of S4P and implements a fully functioning processing system that supports a variety of science processing algorithms and scenarios.

S4PM requires Perl (5.6.0 or higher) with the Perl Tk module. If interoperating with ECS, you will also need the DBI and supporting Sybase modules. On Mac OS 10.x, you will need to have the optional Mac X11 and Xcode packages installed. S4PM has been run successfully on Irix, Linux (Red Hat), Solaris, Macintosh OS X, and Microsoft Windows.

S4PM was released to the open source community under the NASA Open Source Agreement in April 2005 with version 5.6.2. The software is available from SourceForge at this URL: http://sourceforge.net/projects/s4pm/.

1.1 Goals of S4PM

The main goal of S4PM is to automate science processing to the extent that a single operator can monitor all of the processing in an "industrial-size" data processing center. A second goal is to be flexible enough to easily add new processing strings or new algorithms to an existing string with a minimum of effort.

High usability is another key goal of S4PM, deriving from the need for more automation at less operational cost. Specific goals are:

- Allow a single operator to manage and monitor hundreds of jobs simultaneously.
- Drill down to troubleshoot a problem in two mouse clicks.
- Set up a new processing string in less than 30 minutes.

1.2 Future Directions

The architecture of S4PM and S4P was specifically designed to be highly modular so that it could evolve quickly and flexibly. It has already evolved from data-driven processing of MODIS instrument data to AIRS processing to on-demand subsetting based on user requests. Version 5.7.0 was the first release incorporating data mining into S4PM, allowing users to upload algorithms via a Web interface for execution at the GES DISC.

For the future, S4PM will evolve to:

- Support an ever-increasing variety of processing algorithms, scenarios and data interfaces.
- Increase the automation of failure monitoring and recovery.
- Reduce the time and expertise needed to setup and adapt S4PM to new processing algorithms.
- Support for a more distributed S4PM.

We hope that some or all of these goals will be reached by collaborating with the open source community.

2. Related Documentation

The S4PM home page is at: http://disc.gsfc.nasa.gov/techlab/s4pm/ where the following documents are available:

- S4PM 5.15.0 Installation and Configuration Guide
- S4PM 5.15.0 Operations Guide
- S4P Programmer's Guide
- S4PM 5.15.0 Release Notes

3. Design and Implementation Principles

3.1 Simplicity

In all code for S4PM, a concerted effort must be made to keep it simple. Non-comment lines of code should be kept as small as possible without becoming cryptic. (Comments, on the other hand, should be used liberally.)

3.2 The 80/20 Rule

There is a rule of thumb that one often achieves 80% of the functionality with the first 20% of the effort. This tells us to move on to the next item when we have achieved that 80% functionality, coming back for more only if time permits.

3.3 "Use the OS, Luke!"

Years and years of development have gone into the operating system, which is optimized for the machine it is on. Thus, if you can use the operating system to achieve something for you (e.g. files, directories, UNIX commands), do not bother to reinvent it. The only exception is that if you know with absolute certainty that (a) the OS solution will not meet performance requirements and (b) your solution will.

3.4 Design-For-Trouble

Things will go wrong in any system. Therefore, the design should have features to enable troubleshooting built in from the start, not simply added on when things don't work. This does not necessarily mean automated failure recovery, simply ensuring that operators can easily and quickly determine what has gone wrong.

3.5 Transparency

Following from the Design-for-Trouble principle, the operation of the system should be as transparent as possible. That is to say, operational staff and sustaining engineers should be able to see easily everything that is going on, the contents of data moving hither and yon, etc. (This is the principle behind the use of work order files, rather than socket messages, which can disappear into the ether.)

3.6 Keep Things Together

Troubleshooting is easier if everything (e.g., input, output, templates, configuration files, logs, etc.) can be found in one place.

4. S4PM Architecture Overview

4.1 Overall Requirements

The original overall requirements for S4PM were to process the AIRS, DPREP, and MODIS Level-0 data into higher level products using algorithms provided by instrument science teams (DPREP, however, was delivered by the Earth Observing System Data and Information System [EOSDIS] Core System [ECS]). Data were to be obtained from the ECS Science Data Server (SDSRV) and inserted back into same (Figure 4-1). The ECS-provided SDSRV command line interface (SCLI; see Section 10.2) was provided by ECS for the data acquisition, but inserts were to be accomplished using the standard interface for Science Investigator-Led Processing Systems (SIPS), that is, Polling with Delivery Record.

The context diagram for this original S4PM concept is shown in Figure 4-1. S4PM would register a subscription in ECS for Level-0 data needed to initiate processing. The rest of the data flow between ECS and S4PM would proceed as follows:

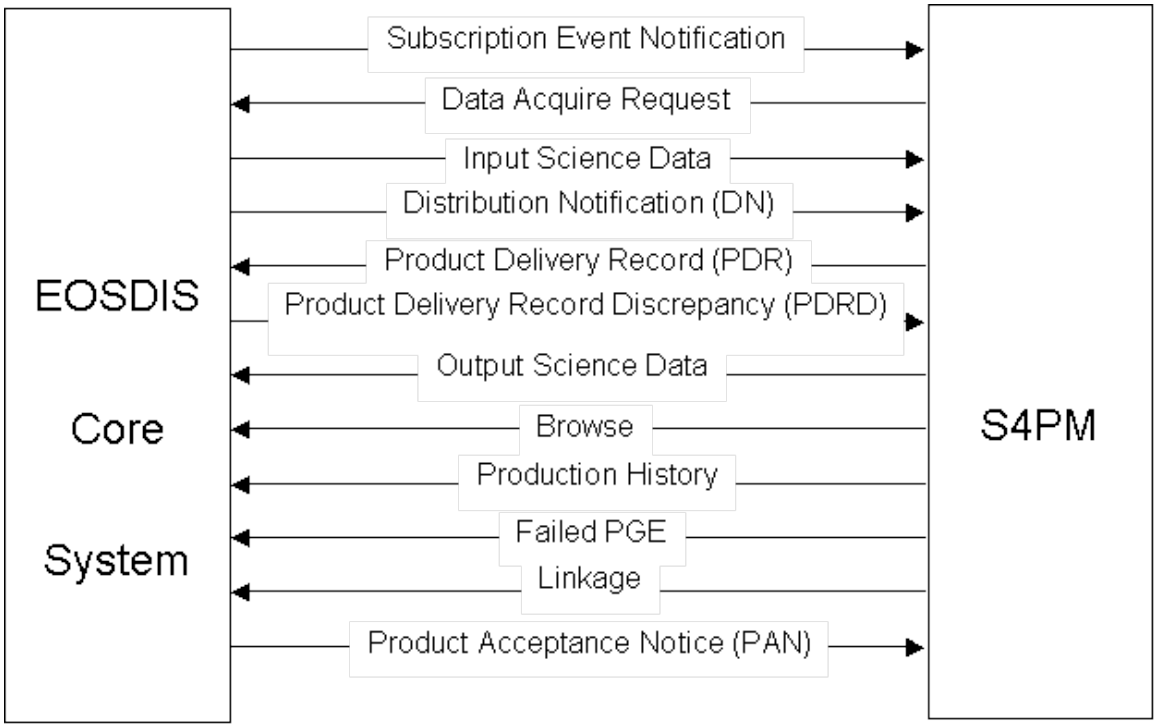


Figure 4-1. Context Diagram for the original S4PM concept.

1. Once Level-0 data became available in the ECS Science Data Server, a subscription notification would be sent to S4PM. This notification would simply alert S4PM that new Level-0 data were available.
2. S4PM would then make a request to acquire those data.
3. ECS would push the Level-0 data to a directory in S4PM.

4. A Distribution Notification (DN) would also be sent by ECS to S4PM immediately after the data were pushed.
5. The receipt of the DN by S4PM would trigger the processing of the Level-0 into higher level science data products. When a product was completed, S4PM would construct a Production Deliver Record (PDR) and place it in a directory where ECS was polling for PDRs.
6. ECS would examine the PDR. If there was a problem in the PDR itself, a Product Delivery Record Discrepancy (PDRD) would be sent back to S4PM. S4PM would then have to repair the offending PDR or the job would fail there.
7. Assuming the PDR passed inspection, the output science data would be pulled into the ECS Science Data Server.
8. Optionally, S4PM would include one or more Browse images associated with the science data in the PDR.
9. Another option would be to include the Production History and have that associated with the data.
10. In the event of an algorithm failure, the debris would be packaged up into a tar file and sent via PDR as a failed algorithm package.
11. A linkage file, which creates the associations between data and Browse or Production History would also be sent if these options were included.
12. Once the ingest of the science data (and Production History or Browse, if included) had been completed successfully, a Product Acceptance Notice would be sent back to S4PM. This would close the loop.

As S4PM became more stable and new requirements arose, S4PM was adapted to fill more processing roles than the original concept held. S4PM was adapted to handle the processing of MODIS direct broadcast data. Modifications were made to enable on-demand processing and the capability of running non-ECS algorithms (that is, algorithms that do not use the ECS Toolkit). S4PM was adapted to work with the ECS Datapool for input data or for inserting output products. The most recent enhancements enable data mining using algorithms uploaded, integrated, and tested via a Web interface. Other additions to S4PM are likely to occur in the future.

4.1.1 Evolution Away From ECS

Because of the evolution of the GES DISC away from ECS, future S4PM versions will reflect this change. As such, there will be no further enhancements to support ECS-specific interfaces, although being an open source project, we welcome contributed code. Rather, enhancements will be in supporting interfaces to the Simple, Scalable, Script-Based, Science Processing Archive (S4PA), another software system based upon the S4P core. For further information about S4PA and its architecture, please see http://disc.gsfc.nasa.gov/techlab/s4pa/index.shtml.

4.2 Architectural Description

S4PM was built upon the core of S4P. For a description of how S4P works, see the S4P Programmer's Guide. The overall set of stations to implement S4PM is shown in Figure 4-2. In reality, several of the stations handle similar operations for multiple algorithms. These may ultimately be broken up into separate stations, or implemented as a single station running different scripts depending on the work order that arrives.

4.2.1 Flavors of S4PM

S4PM supports several built-in architecture configurations. In addition to these built-in configurations, other ad-hoc configurations are possible. The configurations currently available in S4PM are summarized in Table 4-1 below:

S4PM Flavor	Description
Real-Time Standard Processing	The default configuration for S4PM. It is largely data driven. Data arrival triggers a subscription notification, which in turn triggers the automatic ordering of the data. As a result, data are essentially processed as they arrive. Although called “real-time”, it isn’t really quite real time processing. This is the default configuration.
Retrospective Standard Processing	In this configuration, data over a particular time range need to be explicitly requested. The stations are the same as in real-time standard processing except that there is no Subscription Notify station.
On-Demand Processing	On-demand processing is meant for processing service requests (typically for subsetting) from a client or other user interface (*e.g.* EDG). The new stations in on-demand processing are Split Services, Track Requests, and Ship Data. A Standard processing stations not used in on-demand processing is Select Data.
Data Mining	Algorithms are uploaded, integrated, tested, and promoted into an operational string via a Web interface. Products generated are staged to a FTP pull area on disk. Users are notified periodically via e-mail. Each user gets their own S4PM string.
Polling From Datapool	In this configuration, data are polled from the ECS Datapool rather than being retrieved from the ECS archive. In principle, any disk with a directory structure similar to that of the ECS Datapool can be used with this configuration. This configuration can be done with real-time or retrospective processing.
Insert To Datapool	In this configuration, output products are inserted directly into the ECS Datapool, bypassing the ECS archive. Currently, a S4PM string must be configured for all output going to Datapool or all going to the archive. A future release will allow this to be selected by data type. This configuration can be done with real-time or retrospective processing.
ECS Data Archive	In this configuration, the external archive system is assumed to be ECS. This configuration can be combined with other configurations. Not all combinations, however, have been tested.
S4PA Data Archive	In this configuration, the external archive system is assumed to be S4PA. Again, this can be combined with other configurations although not all have been tested.

Table 4-1. Built-in S4PM architecture configurations currently available in S4PM.

4.2.2 Standard Processing Architecture in ECS

In standard processing when the archive is ECS, the arrival of data triggers processing and all data are processed in the order of arrival. In real-time standard processing, the arrival of new data triggers the processing of those data. In retrospective standard processing, data are ordered retrospectively and there is, therefore, controls over which data are processed.

S4PM, like S4P, is essentially a straightforward implementation of the data flow diagram representing the processing. In Figure 4-2, processes are implemented as stations (bubbles) and data flows are implemented as work orders (boxes). All processing is essentially asynchronous. That is, there is no real-time communication between stations (hence no double-headed arrows).

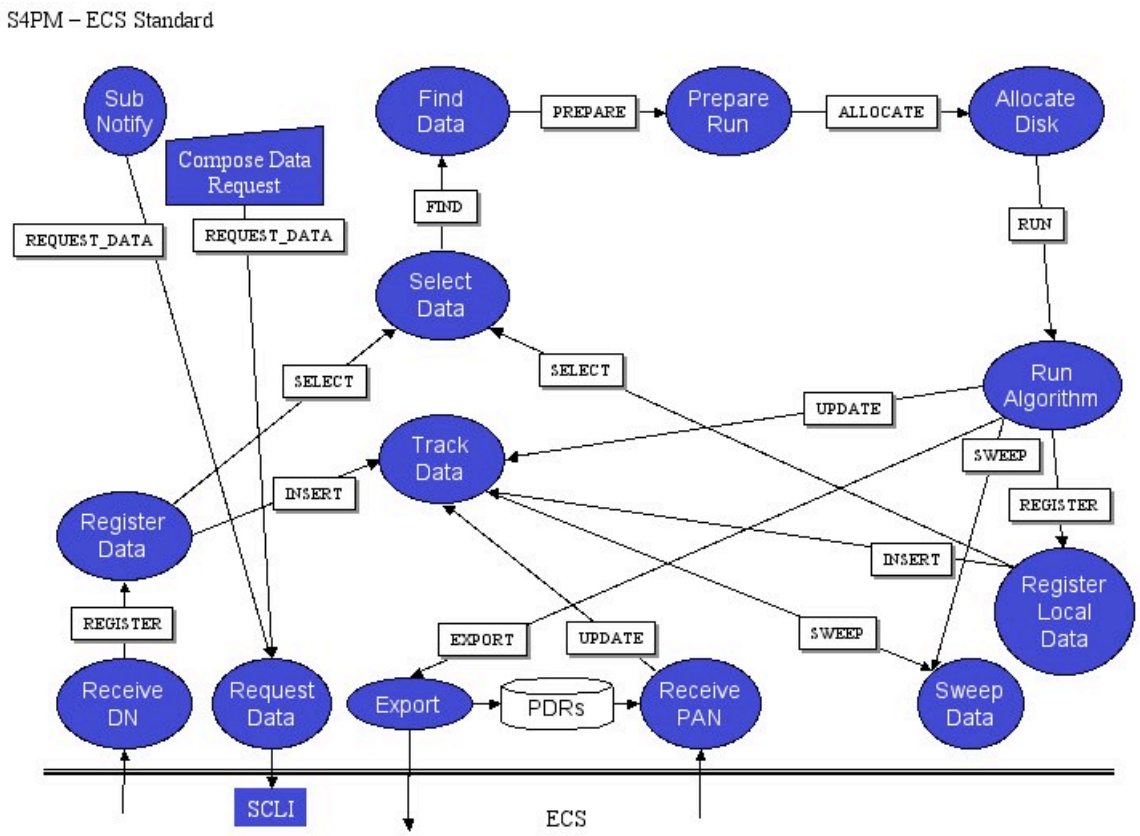


Figure 4-2. S4PM Architecture for Standard Processing interoperating with ECS. Stations are shown as bubbles, with all work order data flows shown as solid arrows. In a few cases, stations share data stores, either databases or simple files, in which case the flows are showed as dashed arrows. In retrospective processing, there is no Subscription Notify station as shown here.

4.2.1.1 Data Flow Initiation

4.2.1.1.1 Standard Real-Time Processing

Subscription Notify → Request Data → │ECS│→ Receive DN → Register Data

In standard near real-time processing, an Insert Notification is sent by e-mail to the S4PM user (the user under whose account the S4PM string is running). A procmail filter is used to send that notification to the Subscription Notify station. This station is common to all S4PM strings.

The Subscription Notify station processes the notification as a NOTIFICATION work order. Based upon the data type listed in the contents, Subscription Notify routes the resulting REQUEST_DATA work order to the Request Data station of the S4PM string needing those data.

The Request Data station first allocates the enough disk space to hold the data and then processes the REQUEST_DATA work order into one or more requests that are sent to ECS via the SCLI. As each request is made, the station keeps track of them by generating a request data stub file. If another request for the same data is received, the Request Data station will quietly ignore it.

Once data requested by the Request Data station are pushed to S4PM, ECS sends a Distribution Notification (DN) via e-mail to the S4PM user. Procmail filtering is used once again to route the DN to the S4PM string that requested the data in the Receive DN station.

The Receive DN station quickly validates the DN and passes a work order onto Register Data. The Register Data station then registers the data within S4PM. At this point, S4PM begins management of the data.

4.2.2.1.2 Standard Retrospective Processing

Compose Data Request Tool → Request Data → │ECS│→ Receive DN → Register Data

In the retrospective case, the data flow is initiated by the operator's selection of input data to be processed. The operator uses the Compose Data Request tool to select a time range and the data types desired. The interface then generates a number of REQUEST_DATA work orders that are sent to the Request Data station.

From this point on, the flow is as shown and described above in Section 4.2.2.1.1.

4.2.2.2 Preparing To Run the Algorithm

Register Data → Select Data → Find Data → Prepare Run → Allocate Disk

The Register Data station registers the new data via an INSERT work order sent to the Track Data station. The Track Data station, in turn, will add the new data files to its databases that maintain directory locations and number of uses outstanding. The number of uses represents the number of times that a data file will be used (read by an algorithm or exported) before that data can be safely deleted. Data are first assigned a maximum use number.

Register Data also sends a SELECT_*algorithm* work order on to Select Data, which specifies other input data needed for processing a given algorithm. That specification is sent as a FIND_*algorithm* work order to the Find Data station to be filled in with the actual locations of the needed files.

Finally, Register Data writes the UR or Local Granule ID (LGID) to a file with a .ur file name extension; this serves as a signal file to the Find Data station that the data are ready for use in downstream processing.

The Find Data station uses the S4PM file name (which itself is configurable) to "predict" the file name (or rather the file name pattern) and attempts to locate in the file system both the file, the metadata file, and the UR file. If found, Find Data fills in the FIND_*algorithm* work order with the appropriate directory and file name information.

Find Data can account for required versus optional data, with timers if necessary. Manual override routines are provided through the Job Monitor to enable the release of jobs with optional data on timers.

Find Data passes on the filled out FIND_*algorithm* to the Prepare Run station as a PREPARE_*algorithm* work order.

The Prepare Run station creates a Process Control File (PCF), using an algorithm-specific PCF template file. The directories for the output data in the PCF are left with placeholders. The Prepare Run station then sends the PCF as a ALLOCATE_*algorithm* work order to the Allocate Disk station, which allocates the amount of disk necessary and replaces the placeholders with actual directories. This fully qualified PCF is then sent to the Run Algorithm station which finally runs the algorithm.

4.2.1.3 Running the Algorithm

Run Algorithm → Register Local Data → {Select Data, Track Data}

The Run Algorithm station executes the algorithm and then sends an REGISTER work order for new data created by the algorithm to the Register Local Data station.

The Register Local Data station is actually a second copy of Register Data with some different configuration parameters. The Register Local Data station sends an INSERT work order recording the output data to the Track Data station, which updates its database accordingly. The Run Algorithm station also sends an UPDATE work order to the Track Data station to let it know that the uses tracked on the input data can be decremented.

Track Data → Sweep Data

When the uses for a data file is reduced to zero, Track Data sends a SWEEP work order to the Sweep Data station which deletes the files and frees the disk allocated by the Allocate Disk station. Finally, some data files are allocated but not actually created (for example, an algorithm that only produces an output file for daytime data). The Run Algorithm station sends a SWEEP work order directly to the Sweep Data station in these cases.

4.2.2.4 Archiving the Data

Run Algorithm → {Export, Insert Datapool} → | ECS |

The Run Algorithm also sends an EXPORT work order to have the output data inserted into ECS or the Datapool (depending upon how the string was configured). For data going to the ECS archive, the Export station renames the Product Delivery Record and places it in a directory for the ECS to pick up. The ECS delivers a Product Delivery Record Discrepancy (PDRD) if the PDR is faulty. Otherwise it attempts to insert the data into the archive generating a Product Acceptance Notification (PAN) detailing which data were successful or unsuccessful. If all were successful, only a short PAN is delivered. Otherwise, a long PAN is delivered. For data going to the Datapool, the process is much the same. A short PAN is delivered if successful, a long PAN if not.

| ECS | → Receive PAN → {Track Data, Export}

The Receive PAN station matches information contained in the PAN with information in the PDR to determine which data can be checked off and sends an UPDATE work order to the Track Data station. For PDRs corresponding to data, the Receive PAN station checks a "PDR Limbo" directory for Browse and Production History PDRs. These PDRs must not be released to ECS before the corresponding data have been successfully ingested. If the PAN has been successful, Receive PAN moves these PDRs from the limbo directory to the Export station from where they will get polled by ECS.

4.2.3 On-Demand Processing Architecture

The base S4PM standard processing model was designed to deal with 24x7 data-driven science processing. An enhancement to this is on-demand or request-driven processing in which services such as subsetting, subsampling, and reprojection can be performed. On-demand assumes that requests come via the V0 Data Gateway although there are hooks

for receiving requests through other means (*e.g.* the Web Hierarchical Ordering Mechanism (WHOM) at the GES DAAC).

The trigger for on-demand processing is an ODL work order resulting from a subset request going to the V0 Data Gateway. This ODL file is sent to the Split Services station of S4PM as a SERVICE work order.

The S4PM architecture for on-demand processing is shown below. Stations shown in red are unique for on-demand processing; they don't show up in standard processing strings. Stations in purple have had significant changes in them in order to support both standard and on-demand processing; and stations in blue are essentially unchanged. Finally, some stations are not used at all in on-demand processing. These are: Subscription Notify and Select Data. In previous releases of S4PM, the Export, and Receive PAN stations were also not used in on-demand processing; they have, however, been brought back in 5.8.1 to support the External Product Dispatcher mechanism.

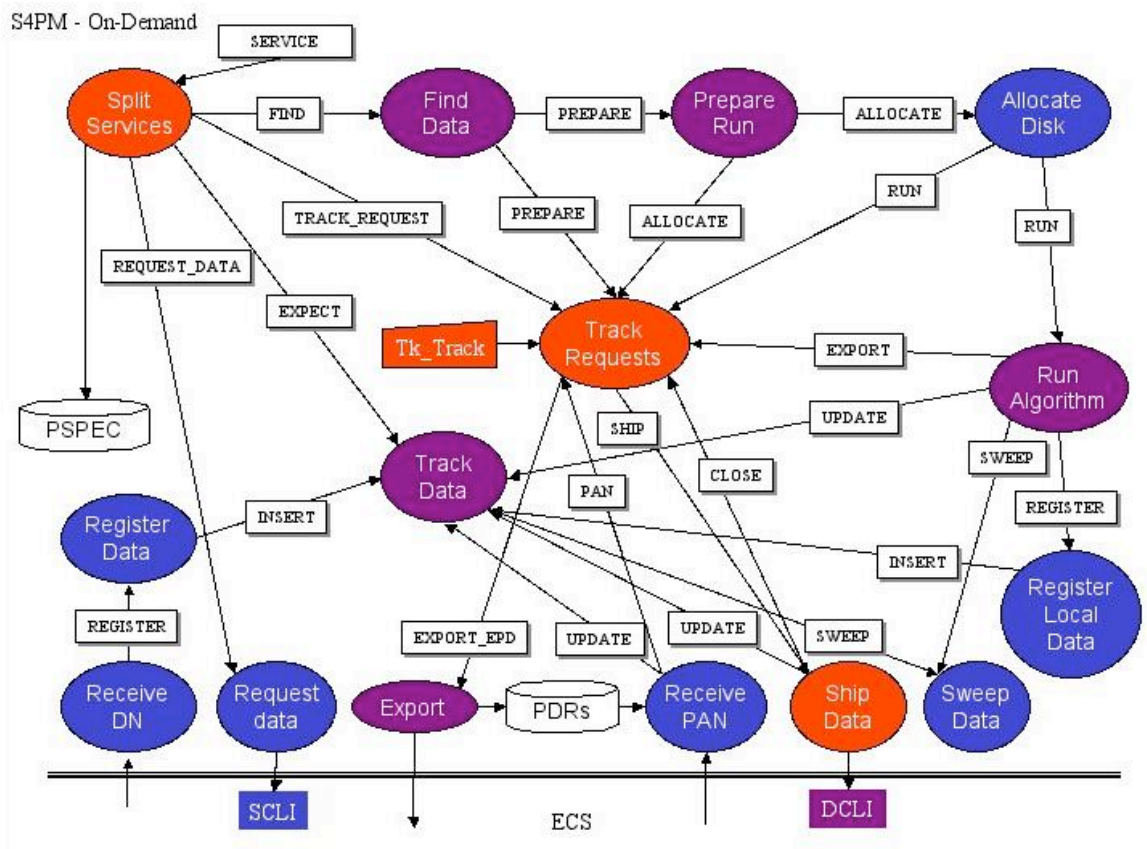


Figure 4-3. S4PM Architecture for On-Demand Processing. In comparison to standard processing, stations in red are new; stations in purple behave differently, and stations in blue are essentially the same as for standard processing.

4.2.3.1 Data Flow Initiation

Split Services → {Request Data, Track Data, Track Requests, Find Data}

The Split Services station is responsible for receiving the initial ODL request from the V0 Data Gateway (or via other means) which describes what services are to be performed on what data. The names of the services requested are included in the ODL along with all data files needed as input to complete those services. The Split Services will break up the initial request according to media type and number of data files.

The request for input needed is sent to the Request Data station via a REQUEST_DATA work order. Unlike in standard processing, the requested data are ordered by default via FTP pull (this is a change instituted in 5.8.1 to improve performance). With FTP pull, data are staged to the ECS Datapool and a notification is sent to the user, in this case, S4PM. Rather than moving the data over from the Datapool to the S4PM input disk pool, symbolic links are instead created to the data on Datapool.

An EXPECT work order is sent to the Track Data station to alert it of the data being ordered. The Track Requests station is sent a TRACK_REQUEST work order that initiates the tracking of this order to completion. Finally, the input ODL file is saved as a PSPEC (Processing Specification) file and saved into the INPUT disk pool as if it were a true input file. Find Data station is notified via a FIND work order to begin polling for the arrival of the ordered data. The trigger data type listed in this FIND work order is the PSPEC file.

│ECS│→ Receive DN → Register Data → Track Data

Once the data ordered by Request Data has arrived, the associated Distribution Notification (DN) is processed by the Receive DN station. A REGISTER work order is then sent to Register Data to receive that data and notify the Track Data station that the data expected has now arrived.

4.2.3.2 Preparing To Run the Service

Find Data → Prepare Run → Allocate Disk

The Find Data station begins polling for the input data once it receives a FIND work order from the Split Services station. Unlike in standard processing, the exact file names of the input data are known and fully specified in the FIND work order. In addition, the trigger data type is always the PSPEC file containing the original ODL file processed by Split Services.

Once the data arrive, a PREPARE_service work order is sent to the Prepare Run station. This station will build a runtime PCF that contains entries for the specialized criteria contained in the referenced PSPEC file along with an entry pointing to that PSPEC file. The PCF will still be missing the directories for the output data to be produced. This is then sent to the Allocate Disk station as a ALLOCATE work order to have space allocated for the output products. The output of Allocate Disk is a RUN_service work order, now a fully qualified PCF, which is sent to the Run Algorithm station.

4.2.3.3 Running the Service

Run Algorithm → {Track Data , Track Requests, Sweep Data, Register Local Data}

Track Data → Sweep Data

The Run Algorithm station runs the service using the PCF generated by Allocate Disk. Upon successful completion, the Track Data station receives updates to the outstanding uses on the input files used. Data whose outstanding uses have fallen to zero are deleted when Track Data sends a SWEEP work order to Sweep Data. For optional output not produced, Run Algorithm sends a work order directly to Sweep Data so that space allocated can be returned to the pool. Finally, a REGISTER work order is sent to Register Local Data to alert it of the newly generated products.

4.2.3.4 Distributing the Data

Run Algorithm → Track Requests → Export → │ECS│ → Receive PAN → Track Data → Sweep Data

Run Algorithm → Track Requests → Ship Data → │ECS│ → Track Data → Sweep Data

Run Algorithm also sends an EXPORT work order to Track Requests. Once Track Requests has received all the EXPORT work orders to complete a user request, it prepares to distribute the data to the user.

Release 5.8.1 supports the External Product Dispatcher (EPD) mechanism for distributing data to users in addition to the old Distribution Command Line Interface (DCLI) method. In EPD, the ECS polls the PDR directory for PDRs listing the data to be distributed as well as information about the user and distribution method. In 5.8.1, the EPD method is used for all orders coming into S4PM via the V0 Data Gateway. The old DCLI method is used for all other orders (the GES DISC supports a client whose subsetting requests do not go through the V0 Data Gateway currently).

For V0 Data Gateway orders, the Track Requests station send an EXPORT_EPD work order to the Export station which examines the work order and places the PDR in the PDR directory. Once ECS has picked up the data, it sends back a PAN which is directed to the Receive PAN station.

For non V0 Data Gateway orders, the Track Requests station instead sends a SHIP work order to the Ship Data station. The Ship Data station, in turn, uses the DCLI to place the data into ECS distribution.

Whether data are shipped via EPD or DCLI, the Track Data receives an UPDATE work order upon successful distribution and issues a SWEEP work order to the Sweep Data station to remove the files and reclaim allocated disk space.

4.2.4 Standard Processing Architecture in S4PA

When the archive is S4PA instead of ECS, the processing flow is largely unchanged. There are, however, a few differences.

First, data are brought into S4PM via the Acquire Data station instead of the Request Data station. The Acquire Data station receives data notifications in PDR format from S4PA for data types in which S4PM has a subscription. The notifications get to the Acquire Data station via a direct FTP push from S4PA into the Acquire Data station directory. In principle, they could also come in via an email message (also supported by S4PA) and directed to Acquire Data via a procmail filter.

The Acquire Data station is responsible for bringing in the data. If the S4PA is local (or at least its directories are visible locally), the station will create symbolic links to the data in the INPUT disk pool. If the S4PA is remote, the data will be transferred to the INPUT disk pool after space for them has been allocated.

Figure 4-4 shows the processing flow for S4PM interoperating with S4PA:

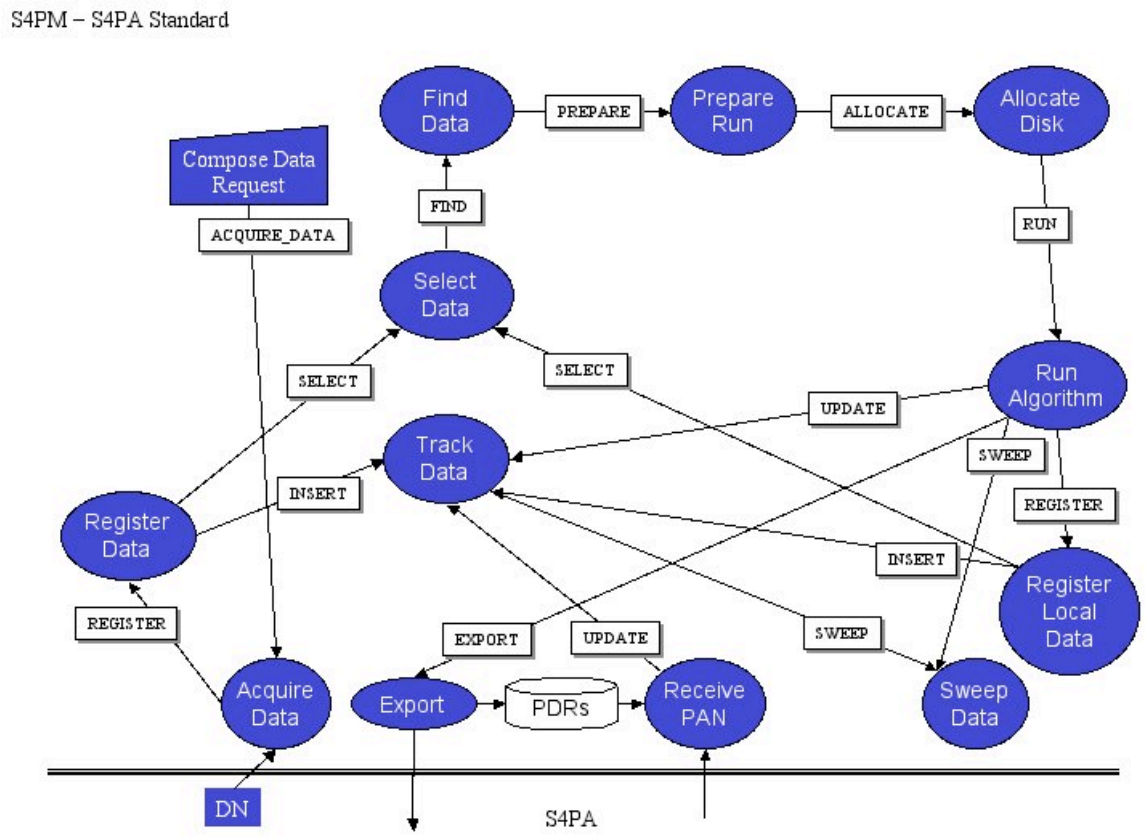


Figure 4-4. S4PM Architecture for Standard Processing interoperating with S4PA. Stations are shown as bubbles, with all work order data flows shown as solid arrows. In a few cases, stations share data stores, either databases or simple files, in which case the flows are showed as dashed arrows.

5. S4PM Stations

This section describes what each station does. Table 5-1 below lists all current S4PM stations.

Station Name	Directory Name	S4PM Flavor	Section
Subscription Notify	sub_notify	Standard (Real-Time with ECS only)	5.1
Split Services	split_services	On-Demand	5.2
Poll PDR	poll_pdr	All (with external data)	5.3
Acquire Data	acquire_data	All (S4PA only)	5.4
Request Data	request_data	All (ECS only)	5.5
Auto Request	auto_request	All	5.6
Poll Data	poll_data	Standard	5.7
Receive DN	receive_dn	All (ECS only)	5.8
Register Data	register_data	All	5.9
Select Data	select_data	Standard	5.10
Track Data	track_data	All	5.11
Find Data	find_data	All	5.12
Prepare Run	prepare_run	All	5.13
Allocate Disk	allocate_disk	All	5.14
Run Algorithm	run_algorithm	All	5.15
Track Requests	track_requests	On-Demand	5.16
Sweep Data	sweep_data	All	5.17
Register Local Data	register_local_data	All	5.18
Export	export	All	5.19
Insert Datapool	insert_datapool	Standard	5.20
Ship Data	ship_data	On-Demand	5.21
Receive PAN	receive_pan	All	5.22
Repeat Hourly/ Repeat Daily	repeat_hourly/repeat_daily	All	5.23

Table 5-1. List of current stations in S4PM.

Each station description in the sections below is summarized in a table preceding that section. These tables include:

1. **Processing Domains:**
   Lists in what processing domain that station is used. Current processing domains are standard processing and on-demand processing.
2. **Scripts:**
   List of the Perl scripts comprising the station
3. **Configuration Files:**
   List of the configuration files used in this station. There is always at least station.cfg.
4. **Logs:**
   List of the log files written to by the station. There is always at least station.log.
5. **Databases:**

List of databases used by the station

6. **Failure Handlers:**
   List of failure handlers available in this station (in at least some incarnations).
7. **Manual Overrides:**
   List of manual overrides available in this station (in at least some incarnations).
8. **Interfaces:**
   List of interfaces (typically, GUIs) available in this station.

Following, the input and output work orders used, their formats, and their source or destination stations are listed. Work order formats are discussed more fully in Section 6.

Many components are common among all or most stations. All stations have their own station.cfg file which is used by stationmaster in monitoring the station. All stations write output to a station.log file as well. The Restart and Remove Job failure handlers are common among most stations (though not all). Note that there is not a one-to-one mapping between stations and the scripts comprising those stations. Some stations are handled via multiple scripts. In other cases, a single script handles multiple stations.

5.1 Subscription Notify (sub_notify)

Processing Domains:	Standard (Real-Time ECS only)
Scripts:	s4pm_sub_notify.pl, send_downstream.pl
Configuration Files:	station.cfg, s4pm_sub_notify.cfg
Log Files:	station.log
Databases:	None
Failure Handlers:	Restart Remove Job Force Request
Manual Overrides:	None
Interfaces:	None

Table 5-2. Subscription Notify Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	NOTIFICATION	E-Mail	ECS
Output:	REQUEST_DATA_*datatype*	PDR	Request Data

Table 5-3. Subscription Notify Station Work Orders

The Subscription Notify is only used in the real-time streams of standard processing.

The Subscription Notify station is responsible for converting Insert Notifications into ECS (for example, see example in Section 6.2.4) into order requests for those same data. Insert Notifications are e-mail messages that inform a subscriber that new data has been inserted into the ECS archive. The S4PM user (under whose account S4PM strings are set up to run) is set up as just such a subscriber for data needed by one or more S4PM forward processing strings. Once received by the S4PM user, the Insert Notifications are processed by a procmail filter and sent to the Subscription Notify station as NOTIFICATION work orders.

A sample procmail filter for converting Insert Notifications into NOTIFICATION work orders and sending them to the Subscription Notify station is shown below:

```
MAILDIR=$HOME/Mail
LOGFILE=$MAILDIR/from
DEFAULT=$MAILDIR/mbox
SHELL=/bin/sh
:0
* ^Subject:.*ECS Notification for Event
$HOME/sub_notify/DO.NOTIFICATION.$$.wo
```

Figure 5-1. Procmail filter used for converting Insert Notifications from ECS into NOTIFICATION work orders directed to the Subscription Notify station.

If there is more than one forward processing S4PM string, a single Subscription Notify station functions as a switchboard of sorts; it directs the requests for data only to those S4PM string(s) that need them. For this to work, all strings needing this capability must see the same Subscription Notify station. This is typically done by having the station reside under the home directory of the S4PM user and having this home directory cross-mounted across all boxes that are running S4PM forward strings. A symbolic link is then created so that this station appears along side the other S4PM stations in the strings that use it.

The Subscription Notify station does checking of NOTIFICATION work orders to ensure that the data start and stop times are nominal. If they are not, the job fails. The **Force Request** failure handler can be used to force the request through regardless.

The output work order from Subscription Notify is a REQUEST_DATA_*datatype* work order which is sent to the Request Data station where datatype is the data type contained in the input NOTIFICATION work order. The data type is appended to the work order name so that, based on the data type, the REQUEST_DATA work order can be sent to the S4PM string(s) needing that data type.

For S4PM strings configured exclusively for reprocessing, where data are ordered via the Compose Request Tool interface, the Subscription Notify station is not shown.

5.2 Split Services (split_services)

Processing Domains:	On-Demand
Scripts:	s4pm_split_services.pl, s4pm_split_order.pl
Configuration Files:	station.cfg, s4pm_split_services.cfg, s4pm_allocate_disk.cfg, Multiple s4pm_select_data_*service*.cfg files
Log Files:	station.log
Databases:	ECS MSS Database
Failure Handlers:	Restart Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-4. Split Services Station Components

On-Demand Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	SERVICE	ODL	EDG, Other Client
	ORDER	ODL	Split Services
Output:	ORDER	ODL	Split Services
	TRACK_REQUEST	PDR	Track Requests
	FIND_*service*	PDR	Find Data
	REQUEST_DATA	PDR	Request Data
	ORDER_FAILURE	PDR	Track Requests
	EXPECT	Pathname	Track Data

Table 5-5. Split Services Station Work Orders

The Split Services station is used only in on-demand processing.

The Split Services station accepts the initial SERVICE work order and breaks it up into separate orders based upon media type. In addition, if the number of data files to service exceed a configurable number (set in the s4pm_split_services.cfg file), Split Services will split the input work order up into smaller pieces. Each such piece is sent as an ORDER work order back to the Split Services station itself.

5.2.1 Processing the REQUEST_DATA Work Order

Data needed by each ORDER work order are sent to the Request Data station as REQUEST_DATA work orders. These work orders are in the form of PDRs, identical in form to those generated by Compose Data Request tool, with the information on the input data encoded in the UR field of each FILE_GROUP representing the individual data files.

The naming of the REQUEST_DATA work orders is important. The job_id field of the original SERVICE work order name will comprise the REQUEST_ID, provided by the search/order client (via the V0 Data Gateway) and the REQUEST_PART_NUMBER,

separated by an underscore. The V0 Data Gateway provides a unique REQUEST_ID; orders coming in via other clients (*e.g.* the WHOM interface at the GES DISC) currently does not, so the Split Services station will need to generate one.

5.2.2 Processing the TRACK_REQUEST Work Order

However, it is also important to pull the output products together into a single order for ease of use at the user's end. Thus, Split Services station must send downstream information on the pieces that make up the original order, in addition to the basic information about the order itself, such as the user information needed to fill the request. It does this by prepending the original ODL request with information mapping each requested file with the REQUEST_PART_NUMBER to which it corresponds, with a line of delimiters in between, as shown in Figure 5-1.

```
1: MODOCL2B.A2001010.2055.041.2003278051811.hdf
SC:MODOCL2B.004:33868119
1: MOD28L2.A2001010.2055.041.2003278045549.hdf SC:MOD28L2.004:33867551
1: MOD03.A2001010.2055.004.2003004170017.hdf SC:MOD03.004:20450665
2: MODOCL2B.A2001081.2100.041.2003284013539.hdf
SC:MODOCL2B.004:34184741
2: MOD28L2.A2001081.2100.041.2003284011956.hdf SC:MOD28L2.004:34183384
2: MOD03.A2001081.2100.004.2003068211917.hdf SC:MOD03.004:23850626
=========================================================
GROUP = PRODUCT_REQUEST
   MESSAGE_ID = "90312"
   REQUEST_ID = "90312"
   DATA_CENTER_ID = "GSFC"
   GROUP = MONITOR
      TX_CLIENT = "1069203596"
   END_GROUP = MONITOR
   GROUP = CONTACT_ADDRESS
      FIRST_NAME = "DOE"
...
etc.
```

Figure 5-1. Example of a TRACK_REQUEST work order.

The first 6 lines map individual files to the REQUEST_PART_NUMBER. Note the inclusion of both the Local Granule ID and the Universal Reference (UR). Though apparently redundant, this supports the addition of future more advanced tracking and debugging capabilities, as the UR is the identifier by which it is identified in Distribution, while the Local Granule ID is the eventual file name the file will have when distributed to S4PMOD.

5.2.3 Processing the FIND Work Order

In a request-driven system, the triggering file is the request itself. Processing cannot proceed until the input data files have arrived. The Find Data station is specifically designed for this situation. However, whereas Find Data is accustomed to receiving only data type and time criteria to identify required additional data, in this case the Local

Granule ID is fully known. Thus, in place of such fields as BEGIN_DATE, etc., the FILE_ID of the FILE_SPEC group is filled in with the Local Granule ID. (The DIRECTORY_ID field is filled in with a stub identifier, INSERT_DIRECTORY_HERE.)

5.2.4 Processing the EXPECT Work Order

One area where request-driven processing differs from data-driven processing is that multiple service requests may legitimately require the same input data. In the data-driven version of the system, such a situation is a non-fatal error, and duplicate requests for the same data are suppressed. Since only one set of algorithms is used on a given file, the number of times it is used is fixed. In the request-driven system, however, the number of uses is a function of the number of different services requested since the data were first requested. As a result, the Track Data station needs to update the number of uses for a given file according to the number of services requested. Because these can come in as multiple unrelated requests, the concept of an EXPECT work order is added. This allows the Track Data station to avoid prematurely purging input data when subsequent work orders arrive for the same data.

```
FileId=MODOCL2B.A2001010.2055.041.2003278051811.hdf Uses=1
FileId=MOD28L2.A2001010.2055.041.2003278045549.hdf Uses=1
FileId=MOD03.A2001010.2055.004.2003004170017.hdf Uses=1
FileId=MODOCL2B.A2001081.2100.041.2003284013539.hdf Uses=2
FileId=MOD28L2.A2001081.2100.041.2003284011956.hdf Uses=2
FileId=MOD03.A2001081.2100.004.2003068211917.hdf Uses=2
```

Figure 5-2. Example of an EXPECT work order.

5.2.5 The ORDER_FAILURE Work Order

When a job downstream of Split Services fails and cannot be easily remedied, a common failure handler is to generate a ORDER_FAILURE work order and send it to the Track Requests station.

5.2.6 The PSPEC File

The SERVICE work order is in ODL format and it contains the criteria for processing the request. The portion of the work order that contains this information is copied by Split Services into a separate file called a Processing Specification (PSPEC) file. This PSPEC file is placed in the input area for the string and treated by S4PM as an input file. In fact, all algorithms (or services) running in an on-demand string are configured to treat the PSPEC file as the trigger data type. Thus, the PSPEC file's arrival in the input area triggers on-demand services to begin preparations for a run.

5.2.7 Additional Features

One of the most common failure modes in on-demand processing is the discovery that the input data are not available. In turn, the two most common reasons for data unavailability are:

(1) Replacement by another data file covering the same time period and
(2) Failure to successfully read the archive tape.

The first case can be handled straightforwardly by replacing the data file to be requested. (The latter is more difficult.) As a result, it is useful for Split Services to first check the database to check the DeleteFromArchive flag of the DsMdGranules table to see that the data are still available in the archive. (Since this query uses dbID, a unique primary key, it is a very fast query.) If the DeleteFromArchive flag is 'Y', the Split Services job automatically replaces the data file with one that does exist for the same time period.

5.3 Poll PDR (poll_pdr)

Processing Domains:	All (S4PA only)
Scripts:	s4p_remote_polling_pdr.pl
Configuration Files:	station.cfg
Log Files:	station.log
Databases:	oldlist.txt
Failure Handlers:	Restart Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-6. Acquire Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	POLL_PDR_*identifier*	N/A	Poll PDR
Output:	ACQUIRE_DATA	PDR	Acquire Data

Table 5-7. Poll PDR Station Work Orders.

The Poll PDR station is used only when the external archive system is providing data to S4PM via PDRs. Although designed to handle the MODIS LAADS system, it should be configurable for any external system that places PDRs in a specific location on a remote host.

The station does its work by periodically polling the remote site for PDRs via FTP or Secure Shell FTP. It keeps track of PDRs already polled via an ASCII file named oldlist.txt. Polled PDRs undergo a few transformations (dictated by command line arguments) and sends the transformed PDRs to the Acquire Data station as ACQUIRE_DATA work orders.

By default, Poll PDR will send everything it finds to the Acquire Data station. If the –w option is included on the command line in the Acquire Data station, however, data types not explicitly configured in the string will be quietly deleted.

5.4 Acquire Data (acquire_data)

Processing Domains:	All (S4PA only)
Scripts:	s4pm_acquire_data.pl, s4pm_tk_compose_request.pl, s4pm_tk_compose_single_request.pl, s4pm_request_subscription.pl
Configuration Files:	station.cfg, s4pm_allocate_disk.cfg, s4pm_tk_compose_request.cfg, s4pm_tk_fillhole_request.cfg
Log Files:	station.log,
Databases:	s4pm_allocate_disk.db
Failure Handlers:	Restart Remove Job
Manual Overrides:	None
Interfaces:	Compose Data Request, Compose Single File Request, Fill Hole Restart All Failed Jobs

Table 5-8. Acquire Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	ACQUIRE_DATA	PDR	Compose Request Tool
	DN	PDR	S4PA
Output:	REGISTER	PDR	Register Data

Table 5-9. Acquire Data Station Work Orders. Although in principle On-Demand should work with Acquire Data, it has not been tested and is therefore not shown.

The Acquire Data station is used only when the external archive system is S4PA and it functions in a role similar to Request Data for when the archive is ECS. There are, however, some key differences other than the station name.

The input Distribution Notification (DN) work orders are PDRs, but unlike with Request Data, they do not arrive at Acquire Data from Receive DN. Instead, S4PA pushes data notifications directly into the Acquire Data station directory. After a data notification is received, the Acquire Data station creates symbolic links to those data in the INPUT disk pool (if the S4PA is local) or it transfers the data over to the INPUT disk pool (if the S4PA is remote). An output PDR is then sent to Register Data.

Note that future releases of S4PM may subsume Request Data functionality into the Acquire Data station. At that point, the need for a separate Request Data station may go away.

5.5 Request Data (request_data)

Processing Domains:	All
Scripts:	s4pm_request_data.pl, acquire, s4pm_tk_compose_request.pl, s4pm_tk_compose_single_request.pl, s4pm_request_subscription.pl
Configuration Files:	station.cfg, s4pm_allocate_disk.cfg, s4pm_tk_compose_request.cfg, s4pm_tk_fillhole_request.cfg
Log Files:	station.log, SCLI.log
Databases:	s4pm_allocate_disk.db
Failure Handlers:	Restart Remove Job
Manual Overrides:	None
Interfaces:	Compose Data Request, Compose Single File Request, Fill Hole Restart All Failed Jobs

Table 5-10. Request Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	REQUEST_DATA	PDR	Compose Request Tool
	REQUEST_DATA_*datatype*	PDR	Subscription Notify
Output:	None	N/A	N/A
On-Demand Processing			
Input:	REQUEST_DATA	PDR	Split Services
Output:	REQUEST_DATA	PDR	Track Requests

Table 5-11. Request Data Station Work Orders

The Request Data station feeds data requests to the ECS. For each UR/LGID in the REQUEST_DATA or REQUEST_DATA_*datatype* work order, the station checks to see if the data have already been requested, by looking for a "stub" file in the REQUESTS subdirectory under the station directory. It also checks to see if there is enough room in the input area. If these conditions are met, it submits an ACQUIRE request to ECS using the SCLI (see Section 10.2). The destination directory for the data is the input pool directory. It also creates stub files for each data file requested indicating that the data are "on the way". This prevents duplicate requests from being submitted.

5.5.1 Standard Processing

In standard processing, the Request Data station generates data requests from one of two sources.

(1) Requests can be specifically generated by an operator using the Compose Data Request Tool (see Section 8.3, an ordering interface that allows data to be ordered by data type and time).
(2) Requests may be generated via the Subscription Notify station which are triggered by Insert Notifications from the ECS (see 5.1).

The interfaces available are:

(1) Compose Data Request which simply invokes the **Compose Data Request** tool.
(2) Compose Single File Request which invokes the **Compose Single File Request** tool that allows smaller amounts of data to be ordered (smaller than in the **Compose Data Request** tool).
(3) Fill Hole which invokes an interface similar to the **Compose Data Request** tool, but with smaller time increments.

5.5.2 On-Demand Processing

In on-demand processing, requests for data are driven solely by the REQUEST_DATA work order from Split Services station. In addition, a REQUEST_DATA work order is sent to Track Services to notify it that a request for data has been made successfully.

5.5.3 Data Mining Processing

In Data Mining strings, the Request Data station functions as it does in standard processing. But only in data mining is the script s4pm_request_subscription used. This script interfaces with the Web front-end and allows a user to place a subscription to data for his or her data mining string. The script generates an e-mail that is sent to User Services requesting the subscription.

5.6 Auto Request (auto_request)

Processing Domains:	All
Scripts:	s4pm_auto_request.pl
Configuration Files:	station.cfg, s4pm_tk_compose_request.cfg
Log Files:	station.log
Databases:	None
Failure Handlers:	Restart, Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-12. Auto Request Station Components

Standard Processing (ECS)			
	Work Order Name	**Format**	**To / From Station**
Input:	AUTO_REQUEST	P=V	Auto Request Tool
Output:	REQUEST_DATA	PDR	Request Data
Standard Processing (S4PA)			
Input:	AUTO_REQUEST	P=V	Auto Request Tool
Output:	ACQUIRE_DATA	PDR	Acquire Data

Table 5-13. Auto Request Station Work Orders

The Auto Request station is associated with the Auto Request Tool. Together they automate the process of ordering or acquiring data on a daily basis to feed the S4PM processing string.

One of the drawbacks of retrospective processing has been that one had to manually bring up the Compose Data Request tool, enter in a year and day of year, and then select which data to retrieve from the archive system for processing in S4PM. This would have to be done each day since only one day's worth of input data could be ordered at a time. The goal of the Auto Request Tool is to automate this process.

The output from the Auto Request Tool is the AUTO_REQUEST work order which contains a list of parameters and their values. This becomes the input work order to the Auto Request station.

The output work order is either a REQUEST_DATA work order going to the Request Data station (if in ECS) or an ACQUIRE_DATA work order going to the Acquire Data station.

5.7 Poll Data (poll_data)

Processing Domains:	Standard
Scripts:	s4pm_poll_data.pl, s4p_repeat_work_order.pl
Configuration Files:	station.cfg, s4pm_poll_data.cfg, s4pm_allocate_disk.cfg
Log Files:	station.log
Databases:	*datatypeproc*.db
Failure Handlers:	Restart Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-14. Poll Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	POLL	N/A	N/A
Output:	REGISTER	PDR	Register Data

Table 5-15. Poll Data Station Work Orders

The Poll Data is used when all input data is to come from the ECS Datapool rather than from the ECS archive. As the name indicates, the Poll Data station polls the appropriate Datapool directories for data that have arrived since the last time the station polled. When new data are found on the Datapool, the station creates symbolic links to the new data in the S4PM input disk pool and sends a PDR to the Register Data station containing the new files. The station is configured to poll the Datapool once a day. The station keeps track of what data have already been polled in files named datatypeproc.db and datatypestat.txt, for example: MOD021KMproc.db and MOD021KMstat.txt.

There is no input work order per se. Only a seed POLL work order is needed. The output work order is a REGISTER which is a PDR containing the new files imported from the Datapool.

The configuration file s4pm_poll_data.cfg contains a number of parameters that control the station:

- $cfg_poll_data_dp_dir - The root directory of the Datapool.
- $cfg_poll_data_max_days - The maximum number of day's worth of data to import at a time. This is applicable only if polling has fallen behind (e.g. station was off for some time) or for the initial polling cycle.
- $cfg_poll_data_max_files_daily - Maximum number of files to expect per day (288 for MODIS).

- $cfg_poll_data_wait_age - The age in days to wait before processing a particular directory. This allows time for the full set of data to populate a give data day's directory.
- %cfg_poll_datatypes - Hash where the keys are the data types to poll and the values are strings containing the data type root directory and the version number. For example: $cfg_poll_datatypes{MYD03} = 'MOGA 004';

5.8 Receive DN (receive_dn)

Processing Domains:	All (ECS only)
Scripts:	s4pm_receive_dn.pl, s4pm_resubmit_data_request.pl, acquire
Configuration Files:	station.cfg, ACQParmfile
Log Files:	station.log
Databases:	None
Failure Handlers:	Remove Job, Resubmit Request
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-16. Receive DN Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	Distribution Notification	E-Mail	ECS
Output:	REGISTER	PDR	Register Data
On-Demand Processing			
Input:	Distribution Notification	E-Mail	ECS
Output:	REGISTER	PDR	Register Data

Table 5-17. Receive DN Station Work Orders

The job of Receive DN station is to process Distribution Notification (DNs) from ECS which are produced when requested data has been distributed to S4PM. DNs are sent via e-mail and, as with Insert Notifications, are disseminated using procmail to the proper S4PM strings(s). A sample DN is shown in here in Section 6.2.4.

The DN contains information on what requested data were distributed successfully and what were not due to some failure. Receive DN has the capability to make a new request for any data that did fail to get distributed while allowing the rest through via the Resubmit Request failure handler. Note that the Restart is not available in this station.

The output from Receive DN is a REGISTER work order that is sent to Register Data. Note that in S4PM strings that interoperate with S4PA instead of ECS, the Receive DN station is not active.

5.9 Register Data (register_data)

Processing Domains:	All
Scripts:	s4pm_register_data.pl, s4pm_purge_bad_data.pl, Various QA scripts
Configuration Files:	station.cfg, s4pm_register_data.cfg, s4pm_allocate_disk.cfg
Log Files:	station.log
Databases:	None
Failure Handlers:	Restart, Remove Job, Bypass QA, Fix L0 Time, Purge Bad QA Data, Purge Ragged L0
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-18. Register Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	REGISTER	PDR	Receive DN
		PDR	Acquire Data
Output:	INSERT	Pathname	Track Data
	SELECT_*algorithm*	PDR	Select Data
On-Demand Processing			
Input:	REGISTER	PDR	Receive DN
Output:	INSERT	Pathname	Track Data

Table 5-19. Register Data Station Work Orders

The Register Data station receives input REGISTER work orders from Receive DN or Acquire Data. The job of Register Data is to rename data to follow a consistent file naming convention, move these data to the proper data subdirectories, and create a UR file (with the .ur extension) for each. The UR file serves as a signal file to any process on the system watching for these data to arrive.

The default file naming convention used is based upon the MODIS convention, but extended to all files. Note that as of S4PM 5.7.0, file naming in S4PM is configurable (within certain limits). See the S4PM 5.15.0 Installation and Configuration Guide for a discussion on this new feature. Here, we will assume the default file naming convention which is:

Datatype[:RegionID].AYYYYDDD.HHMM.VVV.YYYYDDDHHMMSS.hdf

where

- Datatype is the data type name (ESDT ShortName in ECS)
- RegionID (optional) is a regional spatial subset identifier
- YYYYDDD.HHMM is the start date and time of the data file
- VVV is a three-digit data type version (ESDT VersionID in ECS)
- YYYYDDDHHMMSS is the production date/time

Multi-file files (*i.e.* multi-file granules), typically Level-0 data, are maintained as a bundle on S4PM by placing each of its files in a subdirectory under the main input directory, one named for the data file as described above.

Output SELECT_algorithm work orders are sent to the Select Data station. Using information in its configuration files, Register Data includes the name of the algorithm that will be using the data as part of the SELECT_algorithm work order file name.

In addition, Register Data sends a INSERT work order to the Track Data station so that the new data may be registered and tracked within S4PM.

Register Data supports a number of failure handlers in addition to the standard ones. The station can run various quality assessment scripts on incoming data. Such scripts are specified in the station.cfg file using the %quality_assessment hash. When data fail the quality assessment, the job fails. The Bypass QA failure handler will override the quality assessment failure and allow the data into the system. Alternatively, the Purge Bad QA Data failure handler will remove the data responsible for the failed job.

For Level-0 data, Register Data ensures that data times are on expected boundaries. If they are not, the job fails. The Fix L0 Time failure handler causes the station to adjust the data times in the metadata to be on the expected time boundaries and then allows the data into the system. This "fixing" of the time boundaries can be useful when Level-0 data are short. Alternatively, the Purge L0 will purge the offending Level-0 data.

5.10 Select Data (select_data)

Processing Domains:	Standard
Scripts:	s4pm_select_data.pl, s4pm_preselect_data.pl, s4pm_airs_L0_check.pl, s4pm_tk_pwbox.pl, s4pm_check_n_spec.ksh
Configuration Files:	station.cfg, s4pm_airs_L0_check.cfg, s4pm_allocate_disk.cfg s4pm_select_data_*algorithm*.cfg
Log Files:	station.log
Databases:	None
Failure Handlers:	Restart, Remove Job
Manual Overrides:	Modify Threshold, Modify Timer, Release Job Now
Interfaces:	Restart All Failed Jobs

Table 5-20. Select Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	SELECT_*algorithm*	PDR	Register Data, Register Local Data
Output:	FIND_*algorithm*	PDR	Find Data

Table 5-21. Select Data Station Work Orders

The Select Data is not used in on-demand processing strings, although Select Data configuration files are still required.

The Select Data station is responsible for determining what input data are needed for a particular algorithm run. The station is driven by configuration files, one per algorithm, that embody the production rules for that algorithm. These configuration files are algorithm-specific and are named: s4pm_select_data_algorithm.cfg and reside in the select_data_cfg subdirectory under the select_data station directory itself.

When particular data arrive or are produced within S4PM, the Select Data receives notification of those new data via a SELECT_algorithm work order. Based upon the characteristics of the data (usually time coverage and data type), and the production rules used for that algorithm, Select Data determines what other data are needed in order for that algorithm to run. Both required and optional data are determined. For optional data, Select Data determines which of possibly several data sets may satisfy an optional need.

The output of Select Data is a FIND_algorithm work order that describes the additional data needed or desired by this algorithm for this particular run. For optional or alternate inputs, Select Data lists all possibilities in the output work order since it does not know which of these data actually are (or will be) available for a particular run. The FIND_algorithm work order is sent to the Find Data station.

For algorithms that need to accumulate many input files (e.g. an algorithm that produces a daily Level-3 composite based on many Level-2 files), the Input Accumulation production rule may be used. In this case, the s4pm_preselect_data.pl script is run first. Its job is to periodically poll for the input data until a minimum number of files are available within the time allotted. Only then is control passed to the s4pm_select_data.pl script which evaluates the rest of the algorithm's input needs. While polling, the job spins in the Select Data station (stays green). The Modify Threshold, Modify Timer, and Release Job Now manual overrides allow control of jobs running under s4pm_preselect_data.pl. Modify Threshold and Modify Timer allow the minimum number of files and the maximum wait time to be adjusted up or down. The Release Job Now causes polling to be halted; processing is immediately passed to the nominal s4pm_select_data.pl script.

5.10.1 Production Rules

The following sections describe the production rules that are supported in S4PM via the Select Data station. Although represented individually, the supported production rules can generally be combined, although there are exceptions.

In all S4PM processing, each algorithm is triggered by the arrival of a particular input data type. The arrival of this data type essentially tells S4PM that it can begin gathering the rest of the input data needed by that algorithm. This trigger data type can be any of the inputs needed by the algorithm. In some cases, it may be useful to set the trigger data type to one not actually used by the algorithm as input.

See S4PM 5.15.0 Installation and Configuration Guide for a full (and gory) description of how the production rules are described in the Stringmaker algorithm configuration files.

5.10.1.1 Simple Time-Based

The input files staged have data coverages that match the processing time. If the processing period is longer than the input's temporal coverage, more than one input will be matched to cover the entire period.

5.10.1.2 Optional Input with Expiration Timer

One or more optional files may be associated with a single Process Control File (PCF) logical unit number (LUN). Each is associated with an expiration timer and ranked according to preference. If an optional file is available within the time set by the timer, it is used by the algorithm. If it is not available, Select Data will seek the next desirable

choice (if specified). This process continues until either all options have failed to be retrieved within the time allowed or an option is found. If no option is found, the algorithm is allowed to run without that LUN being fulfilled.

5.10.1.3 Required Input with Expiration Timer

One or more required files may be associated with a single Process Control File (PCF) logical unit number (LUN). Each is associated with an expiration timer and ranked according to preference. If a required file is available within the time set by the timer, it is used by the algorithm. If it is not available, Select Data will seek the next desirable choice (if specified). This process continues until either all choices have failed to be retrieved within the time allowed or a choice is found. If no choice is found, the algorithm will not be allowed to run.

5.10.1.4 Required-If Input with Expiration Timer

This production rule says that an input file is required, but only if a test on the trigger input file (or its associated metadata file) is successful. Otherwise, the input is dropped all together. This rule can be used, for example, by an algorithm that requires an input that is only generated for nighttime. The rule would say, in essence, that this input is required only if the trigger input is a nighttime data file. Otherwise, the input is not needed.

The current method for handling this situation is to make the input optional and setting a wait timer on it sufficiently large to catch most of the data that will eventually show up. This, however, is imprecise and some needed data may be missed (resulting in lower quality output) or the job may wait unnecessarily for data that will never show up.

5.10.1.5 Previous n Input File

An input can be for a time earlier than the time indicated by the processing period by n times the input's temporal coverage, where n = 1, 2, 3,..

Such a file may be either optional or required.

5.10.1.6 Following n Input File

An input can be for a time later than the time indicated by the processing period by n times the input's temporal coverage, where n = 1, 2, 3,..

Such a file may be either optional or required.

5.10.1.7 Nearest File in Past

Look backward in time for the nearest input file to the current processing time, with the nearest being the file matching the current processing time. Limits are placed on where to begin looking back and how far to look back.

5.10.1.8 Nearest File in Future

Look forward in time for the nearest input file to the current processing time, with the nearest being the file matching the current processing time. Limits are placed on where to begin looking forward and how far to look forward.

5.10.1.9 Input Accumulation

Some algorithms require many data files of a particular type. For example, an algorithm that aggregates a day's worth of Level-2 files into a single Level-3 daily composite may need hundreds of input files. Using the nominal production rules, the algorithm has two choices: list all the inputs as optional or all as required. In the former case, there is the risk that the compositing algorithm will run with too few inputs (a minimum may be needed for scientifically valid data); in the latter case, if a few inputs never arrive the algorithm will not run even if the data quality would not have been significantly impacted.

To get around this limitation, the input aggregation production rule has been set up. This is supported in S4PM by specifying the minimum number of inputs to accumulate and a maximum time to accumulate. These values can be adjusted "on the fly" during run time. If, for example, a particular day's worth of input is known to be sparse, the minimum threshold can be lowered for that single run to allow the algorithm to run.

5.10.1.10 Spatial Based Processing

S4PM supports rudimentary spatial processing whereby the output product is of a single ESDT over a single time coverage, but over one or more spatial regions. The output file names, is this case, include the Region ID (see file naming convention). The limitations on this support are that the output products cannot be used as input to downstream processing within S4PM.

5.10.1.11 Processing Offsets

By default, the temporal coverage of the output products is determined by the temporal coverage of the trigger input data type. Processing offsets may be used to add an offset in either direction.

5.10.1.12 On-Demand Processing

Although the Select Data station itself is not used in On-Demand processing, the Stringmaker algorithm configuration files are still important components.

In on-demand processing, the term service is used rather than algorithm since the ODL request that comes into the on-demand S4PM system is service based rather than product based. At this stage in the evolution of S4PM, however, the terms service and algorithm are interchangeable since the service name is the name of the algorithm servicing that request.

The trigger data type for all on-demand processing is PSPEC. This is a pseudo-data type since it is really the ODL Processing Specification that initiates processing. As a data type, the PSPEC file is an entry in all PCFs and is assigned to LUN 911. This entry must be specified in the Select Data configuration file along with other inputs needed.

For algorithms that use PCFs (not all are required if they can use the request ODL instead), there must be an association between the specialized criteria supplied in the request (parameters for the algorithm to use) and LUN numbers. This association is done in the algorithm configuration files via the %specialized_criteria hash as illustrated below in Figure 5-3:

```
%specialized_criteria = (
    '20100' => 'CHANNELS|MOD021KM.00[34]',
    '20200' => 'CHANNELS|MOD02HKM.00[34]',
);
```

Figure 5-3. Example from an algorithm configuration file setting specialized criteria.

In Figure 5-3, the LUNs are mapped to specialized criteria names. In this case, the PSPEC is assumed to contain specialized criteria with both CRITERIA_NAMEs set to 'CHANNELS'. If the data type matches the pattern: MOD021KM.00[34] in the PSPEC file, that criteria value is placed in the PCF at LUN 20100. If the data type matches the pattern: CHANNELS|MOD02HKM.00[34] instead, that criteria value is placed in the PCF at LUN 20200. The PCF template, also indicated in Figure 5-3 must also contain entries for these LUNs.

5.10.2 Proxy Data Types

Proxy data types, mainly used in on-demand processing, allow a single data type to represent more than one data type. For example, the data type MODL1B can be a proxy representing the data types MOD021KM, MOD02HKM, and MOD02QKM. When any one of these three data types arrives in S4PM, they are all handled via the MODL1B data type. In fact, MODL1B is seen by S4PM as a fully legitimate data type and can be used as such in, for example, the Select Data configuration files.

Services in on-demand processing typically operate on one of many possible input data types. For example, the GdAIRL1B service can subset any one of several hundred data types. Rather than configure Select Data explicitly for each possible data type, a proxy data type can be configured to represent any one of these. This drastically simplifies the Select Data configuration since only one input data type needs to be configured, the proxy data type.

5.11 Track Data (track_data)

Processing Domains:	All
Scripts:	s4pm_track_data.pl, s4pm_tk_delete_data.pl
Configuration Files:	station.cfg
Log Files:	station.log, transaction.log
Databases:	path.db, uses.db, expect.db
Failure Handlers:	Restart, Remove Job
Manual Overrides:	None
Interfaces:	View/Delete Data, Restart All Failed Jobs

Table 5-22. Track Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	UPDATE	Pathname	Register Data, Run Algorithm, Repeat Hourly, Repeat Daily
	INSERT	Pathname	Register Data, Register Local Data
Output:	SWEEP	Pathname	Sweep Data
On-Demand Processing			
Input:	UPDATE	Pathname	Ship Data, Run Algorithm, Repeat Hourly, Repeat Daily
	INSERT	Pathname	Register Data, Register Local Data
	EXPECT	Pathname	Split Services
Output:	SWEEP	Pathname	Sweep Data

Table 5-23. Track Data Station Work Orders

The Track Data station keeps track of all data in the system using a DBM database.

At least, two work order types are processed by Track Data. The first is the INSERT work order which is used to notify Track Data of new data on the S4PM system. Such new data can arrive from external sources (e.g. via ECS subscription) or can be data created within S4PM from an algorithm. When an INSERT work order is received, Track Data adds that file to its database along with the number of uses expected for that file. Uses are determined from configuration files.

The second type of input work order is the UPDATE work order. This simply informs Track Data that another use has been made against a particular data file in its database. Track Data decrements the number of outstanding uses appropriately. Once the number of outstanding uses has been reduced to zero, that data file is deleted from the database and a SWEEP_DATA work order is sent to the Sweep Data station for physical deletion of the file and reallocation of disk pool space reserved for it. The UPDATE work order may also contain the wildcard character '*'. In this case, the entry is interpreted as a file name pattern that is matched against zero or more files in the database. This is currently used when algorithms are installed dynamically as a way of modifying the uses of all files of a particular data type.

An optional third type of work order is the EXPECT work order. It is mainly used in on-demand processing strings where the following sequence of events would lead to premature deletion of data:

1. Request A is received to apply service A to data file 123
2. The Request Data station submits a request for file 123
3. Request B is received to apply service B to data file 123
4. The Request Data station finds an existing stub file and decides not to re-request file 123
5. File 123 arrives and is inserted, with Uses=1
6. Service A is executed, and Uses is decremented to 0, resulting in the file's deletion
7. Service B fails to find the input file and fails

In order to prevent this, the concept of expecting data is added. Each Split Services invocation sends an EXPECT work order to Track Data, which increments the number of expectations in an expect.db file. When the INSERT work is received from Register Data, the number of expectations is substituted for the Uses that come with the INSERT work order. If a subsequent EXPECT work order is received, the number of uses is incremented.

The Sweep Data interface is a GUI that allows operators to delete individual files from the S4PM system. See Section 8.4 for more information.

5.12 Find Data (find_data)

Processing Domains:	All
Scripts:	s4pm_find_data.pl, s4pm_bootstrap_dprep.pl, s4pm_tk_pwbox.pl, s4pm_failed_find_data_handler.pl
Configuration Files:	station.cfg, s4pm_allocate_disk.cfg
Log Files:	station.log, find_data.log
Databases:	None
Failure Handlers:	Restart, Remove Job, Bootstrap, Fail Order
Manual Overrides:	Expire Current Timer, Ignore Optional, Ignore Required
Interfaces:	Zoom In, Restart All Failed Jobs

Table 5-24. Find Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	FIND_*algorithm*	PDR	Select Data
Output:	PREPARE_*algorithm*	PDR	Prepare Run
On-Demand Processing			
Input:	FIND_*service*	PDR	Split Services
Output:	PREPARE_*service*	PDR	Prepare Run

Table 5-25. Find Data Station Work Orders

5.12.1 Standard Processing

In standard processing strings, the input FIND_*algorithm* work order is distinguished from the FIND_*service* for on-demand processing by the presence of filled out directory locations for all files and only one FILE_ID filled out (that for the trigger data type). In this mode, the Find Data station attempts to locate the required and optional input data based upon the FIND_*algorithm* work order it receives from Select Data. It does so by formulating file name patterns for each required or optional data and polls the data area for files matching those patterns. In cases where the file names are fully specified, a simple test for the file's existence is made rather than running a pattern glob on the entire directory. In on-demand processing, this is the case (see Section 5.11.2). But if the file name pattern used in S4PM is configured to not contain any production date/time information, file names are fully specified even in standard processing and no pattern globbing is performed. Refer to the S4PM 5.15.0 Installation and Configuration Guide.

Several types of polling are available in the Find Data station:

Type of Polling	Description
Standard	In standard polling (the default for standard processing strings), polling of the disk pools for input data is done at a fixed frequency. Currently, the frequency is set to 30 seconds; every 30 seconds, the job polls the file system and sleeps in between pollings.
Smart Polling	In smart polling, the polling frequency can be configured to change over time such that the longer data are not found, the less frequently the station polls for them. Once a data file is found, the polling frequency is reset to the highest frequency (*e.g.* 30 seconds) under the theory that the likelihood of finding another data file goes up.
Recycle	In recycle mode, polling for all data is done once. If not all data are found, the job exits immediately and is requeued to run again. This is the default mode for on-demand processing and it generally increases performance through the Find Data station.

Table 5-26. Types of polling available in the Find Data station.

The input FIND_*algorithm* work order from which Find Data works fully describes which data are required and which are optional. It also specifies how long Find Data should wait before giving up. For required data, the timer starts once the FIND_*algorithm* work order is received; this is essentially the time when the trigger data type arrives. For optional data, the timer starts once all required data have been found by Find Data

Find Data is also cognizant of the order of preference for required and optional data for algorithms that are configured to use more than one possible input. The station looks first for the data with the highest preference first. If that data is not found within the time allotted, it looks for the next highest preferred data, etc.

If, after all timers have expired, all required data have been located, Find Data outputs a PREPARE_*algorithm* work order that contains all the data found, both required and optional. If on the other hand, any required data have not been found, that job in Find Data fails.

While polling for data, jobs in the Find Data station can run for extended period of time (for some algorithms wait timers are in weeks). For added capability, Find Data supports a number of manual overrides, in the form of signal files, that override wait timers. For example, if all required data has been found and a job is waiting on optional data that an operator knows will never arrive, the operator can force the job to timeout using the Expire Current Timer. To ignore all remaining optional data, the Ignore Optional manual override can be used or the Ignore Required for required data.

Manual Override	Description
Ignore Optional	Expire all wait timers on optional data, look one last time for them, and then end.
Ignore Required	Expire all wait timers on required data, look one last time for them, and then end. In this case, the algorithm must be able to handle missing required data.
Expire Current Timer	Expire only the current timer (the one ticking away now) and move onto looking for next data.

Table 5-27. Current manual overrides supported by the Find Data station and their meanings.

The Zoom In starts a new Tkstat GUI (see Section 8.1) to be brought up that zooms in on the station with which it is associated showing the breakout of input work order types as if they were multiple instances of the same station. See this example. In the Find Data station, the zoom feature allows one to easily see which work order subtypes are being processed where subtype equates to algorithm.

The Bootstrap failure handler is unique to DPREP algorithms running in the GES DAAC. It is used only in the case where DPREP needs to run without having the benefit of any look-behind input data. Invoking this failure handler triggers DPREP into running in an alternate mode where it can bootstrap without the need for look-behind data.

5.12.2 On-Demand Processing

In on-demand processing, the input FIND_*service* work order is already complete with all input file names known and specified (FILE_ID). The directory locations, however, contain placeholders (INSERT_DIRECTORY_HERE). In addition, the trigger data in all work orders is PSPEC. When Find Data sees such work orders, it knows that it is running in an on-demand processing mode.

An additional failure handler in on-demand processing is Fail Order. When Find Data cannot find all the data it needs within the allotted time, it fails. If the data problem cannot be resolved, an operator has the option of failing the order by invoking this failure handler. A ORDER_FAILURE work order is sent to the Track Requests station to notify it of an unrecoverable failure.

5.13 Prepare Run (prepare_run)

Processing Domains:	All
Scripts:	s4pm_prepare_run.pl
Configuration Files:	station.cfg, s4pm_prepare_run.cfg, Multiple s4pm_select_data_*algorithm*.cfg
Log Files:	station.log
Databases:	None
Failure Handlers:	Restart, Remove Job
Manual Overrides:	None
Interfaces:	Modify PCF Runtime Parameters, Restart All Failed Jobs

Table 5-28. Prepare Run Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	PREPARE_*algorithm*	PDR	Find Data
Output:	ALLOCATE_*algorithm*	PCF	Allocate Disk
On-Demand Processing			
Input:	PREPARE_*service*	PDR	Find Data
Output:	ALLOCATE_*service*	PCF	Allocate Disk, Track Requests

Table 5-29. Prepare Run Station Work Orders

5.13.1 Standard Processing

In standard processing, the Prepare Run station receives a PREPARE_*algorithm* work order from Find Data that contains all the data found for a particular algorithm run and generates a Process Control File (PCF), using a algorithm-specific PCF template. This PCF template contains all possible LUNs used by that algorithm (input files, output files, support files, and user defined parameters). The PCF output generated is a ALLOCATE_*algorithm* work order and it contains the file names and directory locations of all input data for a particular algorithm run. It also contains the file names of all output data to be produced. No output directories, however, are set. Instead, Prepare Run uses placeholders.

The output file names follow the file naming convention described earlier (see Section 5.4).

The PCF output work order is then sent to the Allocate Disk station which will complete the PCF.

The Modify Runtime Parms is a GUI that allows PCF runtime parameters to be modified. It is used in cases where the runtime parameter is a switch that, for instance, turns on or off a particular algorithm behavior or process.

5.13.2 On-Demand Processing

While all of the algorithms in standard processing S4PM strings run using ECS Process Control Files, in on-demand processing strings, S4PM is designed to also handle algorithms that use ODL request files as input. These files are known as Processing Specification (PSPEC) files and they are handled simply by referencing them in the PCF as an input data file. Since the PSPEC is treated as an input data type, Prepare Run will place it in the output PCF with a pre-determined Logical Unit Number (LUN). The LUN is assigned in the Select Data configuration file as with other data types.

On the other hand, for on-demand algorithms that do use PCFs, the Prepare Run station will fill in the runtime parameters with the information in the PSPEC file. This is driven by the Select Data configuration file for each algorithm that maps LUNs to CRITERIA_NAMEs.

5.14 Allocate Disk (alloc_disk)

Processing Domains:	All
Scripts:	s4pm_allocate_disk.pl, s4p_repeat_work_order.pl, s4pm_tk_disk_alloc.pl
Configuration Files:	station.cfg, s4pm_allocate_disk.cfg, s4pm_allocate_disk_pool.cfg
Log Files:	station.log
Databases:	s4pm_allocate_disk.db
Failure Handlers:	Restart, Remove Job
Manual Overrides:	None
Interfaces:	View Disk Allocation and Usage, Zoom In, Restart All Failed Jobs

Table 5-30. Allocate Disk Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	ALLOCATE_*algorithm*	PCF	Prepare Run
	UPDATE_POOLS	Perl	Stringmaker
Output:	RUN_*algorithm*	PCF	Run Algorithm
On-Demand Processing			
Input:	ALLOCATE_*service*	PCF	Prepare Run
	UPDATE_POOLS	Perl	Stringmaker
Output:	RUN_*service*	PCF	Run Algorithm, Track Requests

Table 5-31. Allocate Disk Station Work Orders

The Allocate Disk station receives a ALLOCATE_*algorithm* work order from Prepare Run. This work order is an incomplete PCF. Still missing are the directory locations of output files to be produced. Allocate Disk maintains a database of available disk pools and the current space available in each. Using the input ALLOCATE_*algorithm* work order, Allocate Disk determines how much disk space is needed for each output file from this algorithm using configuration files that define typical or maximum file sizes. If space for the output data is available in the corresponding disk pool, Allocate Disk reserves a block of space against that pool in the database. Once space is available for all output data to be produced, the output directory placeholders in the input PCF are replaced by actual directory locations.

If all disk space needed by a algorithm cannot be satisfied, the strategy employed is to have that job immediately terminate and get requeued for a later attempt (using the repeat_work_order.pl script). This allows other jobs, whose disk pools may have plenty of space, to get their turn.

With all directory placeholders in the PCF filled in, the output PCF becomes a RUN_*algorithm* work order, a fully qualified PCF. That work order is sent on to the Run Algorithm station.

The View Disk Allocation and Usage interface is a visual display showing the current used and free space in all the disk pools configured. See Section 8.6. The Zoom In interface plays the same role as it does with the Find Data station in Section 5.11.

The UPDATE_POOLS is a special work order. Its content is identical to that of the s4pm_allocate_disk_pools.cfg file, that is, Perl. The file contains the sizes in bytes for each of the disk pools configured in the string. When the station sees an UPDATE_POOLS work order, it will incorporate any changes to sizes in the work order into the s4pm_allocate_disk_pools.cfg file. Thus, the UPDATE_POOLS work order can be used to modify on-the-fly the size of disk pools.

5.14.1 On-Demand Strings

On-demand strings are configured by default with only two disk pools: INPUT and OUTPUT. All output products go into the OUTPUT disk pool. In addition, the View Disk Allocation and Usage tool only shows the OUTPUT pool; it does not display the INPUT pool since all files in the INPUT pool are symbolic links to data on the Datapool. Because of this, no space is actually allocated for input data and, therefore, space usage in the INPUT pool is not tracked.

Note also that on-demand strings are configured by default to take advantage of smart allocations. With this option, the space allocated for output data is based on a predicted file size, but once that file is produced, the allocation is adjusted based on the actual file size. This feature is enabled by default for all on-demand strings via the $smart_allocation parameter in the Stringmaker string configuration file (see the S4PM 5.15.0 Installation and Configuration Guide).

5.15 Run Algorithm (run_algorithm)

Processing Domains:	All
Scripts:	s4pm_run_algorithm.pl, s4pm_run_easy.pl, s4pm_failed_pge_handler.pl, clean_pge_output.pl
Configuration Files:	station.cfg, s4pm_allocate_disk.cfg, QC.cfg, *algorithm*.cfg, dataSize_*mission*.cfg
Log Files:	station.log, checksum*.txt, *.rul, STATS*.txt
Databases:	None
Failure Handlers:	Restart, Remove Job, Punt, Fail Order, QC Continue
Manual Overrides:	None
Interfaces:	Zoom In, Restart All Failed Jobs

Table 5-32. Run Algorithm Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	RUN_*algorithm*	PCF	Allocate Disk
Output:	SWEEP	Pathname	Sweep Data
	EXPORT	PDR	Export
	REGISTER	PDR	Register Local Data
	UPDATE	Pathname	Track Data
On-Demand Processing			
Input:	RUN_*service*	PCF	Allocate Disk
Output:	SWEEP	Pathname	Sweep Data
	EXPORT	PDR	Track Requests
	REGISTER	PDR	Register Local Data
	UPDATE	Pathname	Track Data

Table 5-33. Run Algorithm Station Work Orders

The Run Algorithm station is responsible for actually running the algorithm using the fully qualified PCF from the Allocate Disk station.

After an algorithm is run, Run Algorithm checks its exit code. A zero is considered success; anything else is considered failure. If the algorithm was successful, several actions are performed:

1. The data produced by the algorithm marks the arrival of new data in the system. Thus, an REGISTER work order is generated by Run Algorithm and sent to the Register Local Data station, essentially a clone of Register Data but for locally produced data. This in turn may trigger additional processing on the output data.
2. An UPDATE work order is also generated and sent to Track Data so that the uses associated with the input files can be appropriately decremented.
3. A Production History (PH) tar file is generated containing the logs and other information resulting from the algorithm run. It also contains a S4PM chain log tracking the progress of data through the S4PM system up to and including the run of the algorithm.
4. An EXPORT work order is sent to the Export station for those data that are to be sent into the ECS archive. An export counts as a use against that data for tracking purposes. For PH and Browse files, special handling is required to ensure that the data products have been successfully ingested by ECS before they are exported themselves. To accomplish this, PH and Browse files are placed in a "limbo" directory. Once the data with which they are associated have been ingested successfully into the ECS, they are released to the Export station.
5. Run Algorithm also tracks various performance statistics including the average volume rates and throughput. These are updated after every algorithm run.

If the algorithm fails, a Failed PGE tar file is generated. It is similar to the PH and can be exported to the ECS as any other type of data via the Punt failure handler.

The Run Algorithm can be configured to perform quality control (QC) on data products it produces. Common QC checks (such as file size checking) and be configured for all products while other QC checks can be associated with only particular data types. A product that fails QC checking can be blocked from export to the ECS archive, blocked from being used in downstream processing, or both. In addition, QC failures can be deemed fatal, in which case the algorithm job fails. Alternatively, the QC failure could simply be logged while the algorithm job itself succeeds. All this is done in the QC.cfg configuration file.

If an algorithm is configured to fail when one of its products does not pass QC checking, the QC Continue failure handler allows an operator to have the good products (if any) continue on (e.g. to Export or downstream processing) and the bad products which failed QC checking to be removed from the system.

The Zoom In interface plays the same role as it does with the Find Data station in Section 5.11.

5.15.1 On-Demand Processing

PCF-driven algorithms require no changes to the Run Algorithm software. However, for PSPEC-file driven algorithms (recognized via a command-line switch), a shell (pspec_svc.pl) is used to extract the path of the PSPEC file from the PCF and run the

algorithm from the command line. The syntax by which pspec_svc.pl calls the algorithm is:

algorithm PSPEC_FILE

The algorithm is responsible for opening and parsing the PSPEC file to obtain the necessary subsetting criteria.

One further change to the software is the suppression of Production History file creation via another command line switch.

The configuration of Run Algorithm is slightly different in that the work order that normally goes to Export now goes to Track Requests.

In the event of a algorithm failure in on-demand processing, the Punt failure handler is replaced with a Fail Order failure handler. It performs very similar tasks to the Punt except that rather than sending a EXPORT_FAILPGE PDR work order the Export station, it sends the same PDR as a ORDER_FAILURE work order to the Track Requests station. Both Punt and Fail Order failure handlers are handled by the same script.

5.16 Track Requests (track_requests)

Processing Domains:	On-Demand
Scripts:	s4pm_track_requests.pl, ignore_failure.pl
Configuration Files:	station.cfg, s4pm_track_requests.cfg
Log Files:	station.log
Databases:	None
Failure Handlers:	Restart, Remove Job, Ignore Order Failure
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-34. Track Requests Station Components

On-Demand Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	TRACK_REQUEST	ODL	Split Services
	PREPARE_*service*	PDR	Find Data
	ALLOCATE_*service*	PCF	Prepare Run
	RUN_*service*	PCF	Allocate Disk
	EXPORT	Pathname	Run Algorithm
	CLOSE	ODL	Ship Data
	N/A	PAN	Receive PAN
	ORDER_FAILURE	Mixed	Multiple Stations
Output:	SHIP	PDR	Ship Data
	EXPORT_EPD	PDR	Export

Table 5-35. Track Requests Station Work Orders

The Track Requests station is used only in on-demand processing.

The Track Requests station's primary responsibility is to handle the other end of Split Services, namely the joining of the outputs back into a single request for shipping to the user. Because it can do this only when it has determined that all of the outputs are ready, it takes on the responsibility of tracking the progress of each individual part of the request as it progresses through the system.

The initial input work order for a request is the concatenated TRACK_REQUEST work order, which Track Requests splits into its two constituent parts. It creates a directory named after the REQUEST_ID for each request, and places the ODL part of the file in that directory for future reference. It also sets up subdirectories under the ACTIVE_REQUESTS directory for each order being actively tracked using the jobid of the original TRACK_REQUESTS work order. Within the ACTIVE_REQUESTS/jobid directory, a number of subdirectories are set up to track status as shown in Table 5-32.

A file is generated for each REQUEST_PART, and named accordingly, e.g., 92312_1.txt for the first part of request 92312, and placed in the A.waiting_for_REQUEST_DATA/ directory. Subsequently, several S4PM station configurations are modified to include the Track Requests station as a downstream station for their output work orders. As Track Requests receives each one, it identifies the REQUEST_ID and the REQUEST_PART_NUMBER by parsing the job_id of the work order name. It then updates the status by moving the REQUEST_PART file from the previous directory into the next status up. The correspondence of output work orders to the new status is as follows:

This Work Order	From This Station	Routed To This Subdirectory	Means This…
TRACK_REQUEST	Split Services	A.waiting_for_REQUEST_DATA	Order request initiated, waiting for data
PREPARE_*service*	Find Data	B.waiting_for_TRIGGER	Data received
ALLOCATE_*service*	Prepare Run	C.waiting_for_GETDISK	Waiting for disk
RUN_*service*	Allocate Disk	D.waiting_for_RUN	Running
EXPORT	Run Algorithm	E.waiting_for_EXPORT	Running completed.
N/A	N/A	F.service_complete	Request completed.
N/A	N/A	G.waiting_for_CLOSE	All requests for this order have been completed and order is ready for distribution to user.
CLOSE	Ship Data	H.distribution_completed	All requests for this order have been distributed.
ORDER_FAILURE	Multiple	Z.order_failure	Order has failed.

Table 5-36. Mapping of work order to progress through the Track Request station's subdirectories.

The handling of output work orders from Run Algorithm is special in two respects. Firstly, it parses the output work order (EXPORT) to obtain the full paths of the output files, saving them in an ASCII file (one for each REQUEST_PART). Secondly, each time a file is moved to the E.waiting_for_EXPORT/ (running completed) status directory, Track Requests looks to see if the whole request is complete. If so, it concatenates the output file lists for each REQUEST_PART and generates an output work order for the GetID station (eventually to be combined with Ship to "replace" Export).

This whole process can be monitored using the main S4PM Monitor, as all the information is included in directories and ASCII files. However, a graphical user interface is provided to view and troubleshooting orders. In future releases, Track Requests will also communicate with the ECS system via the ORDER_STATUS message (see ICD for ECS Interfaces That Support External Subsetters Located at DAACs).

The Ignore Order Failure failure handler is used to ship an order when there is partial failure.

For requests coming into S4PM via the V0 Data Gateway, Track Requests ships data to the user via the External Product Dispatcher (EPD) method. Support for EPD is new in version 5.8.1. For such requests, Track Requests send a EXPORT_EPD work order to the Export station which reviews the PDR and places it in the PDR directory. Upon detection of the PDR by ECS, the data are placed in ECS distribution and a PAN is sent back to the Receive PAN station. The Receive PAN station then sends the PAN onto Track Requests so that it can close out the order.

For requests not coming in via the V0 Data Gateway, a SHIP work order is instead sent to the Ship Data station as in S4PM releases prior to 5.8.1.

5.17 Sweep Data (sweep_data)

Processing Domains:	All
Scripts:	s4pm_sweep_data.pl
Configuration Files:	station.cfg, s4pm_allocate_disk.cfg
Log Files:	station.log
Databases:	s4pm_allocate_disk.db
Failure Handlers:	Restart, Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-37. Sweep Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	SWEEP	Pathname	Run Algorithm, Track Data
Output:	None	N/A	N/A
On-Demand Processing			
Input:	SWEEP	Pathname	Run Algorithm, Track Data
Output:	None	N/A	N/A

Table 5-38. Sweep Data Station Work Orders

The Sweep Data station physically deletes files from the system based on its input and send a work order to Allocate Disk so that disk space can be appropriately freed up. If there are "stub" files associated with any data to be deleted, these are deleted too. Stub files are created for data ordered from the ECS via the Compose Request interface (see Section 8.3).

5.18 Register Local Data (register_local_data)

Processing Domains:	All
Scripts:	s4pm_register_data.pl, s4pm_block_daily.pl
Configuration Files:	station.cfg, s4pm_register_data.cfg, s4pm_allocate_data.cfg
Log Files:	station.log, block.log
Databases:	None
Failure Handlers:	Restart, Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-39. Register Local Data Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	REGISTER	PDR	Run Algorithm
Output:	INSERT	Pathname	Track Data
	SELECT_algorithm	PDR	Select Data
On-Demand Processing			
Input:	REGISTER	PDR	Run Algorithm
Output:	INSERT	Pathname	Track Data

Table 5-40. Register Local Data Station Work Orders

The Register Local Data station is a clone of the Register Data station, but used for internally generated data. See 5.5 for details on this station.

In on-demand processing there is no algorithm chaining. Therefore, no SELECT work orders are sent out.

5.19 Export (export)

Processing Domains:	Standard
Scripts:	s4pm_export.pl
Configuration Files:	station.cfg, s4pm_export.cfg
Log Files:	station.log
Databases:	None
Failure Handlers:	Restart, Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-41. Export Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	EXPORT	PDR	Run Algorithm
	EXPORT_EPD	PDR	Track Requests
Output: **Output:**	None	PDR	ECS Polling Directory
	None	PDR	Insert Datapool
	INSERT	Pathname	Track Data

Table 5-42. Export Station Work Orders

In previous releases of S4PM, the Export was not used in on-demand processing strings. As of release 5.8.1, the Export station is used to support the External Product Dispatcher (EPD).

The Export is responsible exporting data produced by the algorithm to either the ECS archive or the ECS Datapool via the SIPS interface. In on-demand strings, Export is used to send data into the ECS Distribution subsystem for distribution to the data requestor. The s4pm_export.cfg lists the data types processed by the Export station as well as their destinations: archive, Datapool, EPD, or ersatz ("fake" ingest). The input EXPORT work order contains those files to be exported.

If an incoming PDR contains more than one destination (i.e. some output products are going to Datapool while others are going to the archive), Export will split the PDR into two output PDRs, one for each destination. For products to be exported to the ECS archive, the station will place the PDR directly (rather than having Stationmaster do it) into the ECS polling directory set up for that PDR interface. For products to be exported to the ECS Datapool, the station will place the PDR directly in the Insert Datapool station directory. For data being distribute to users (in on-demand strings), the station will also place those PDRs directly into a PDR directory. Thus, there are no output work orders per se from the Export station.

For testing purposes, the Export can be configured to export files to a fake interface, one that mimics at a crude level the response of the ECS interface. Such exported files simply disappear.

There is technically no downstream station from Export so no output work order is created.

5.20 Insert Datapool (insert_datapool)

Processing Domains:	Standard
Scripts:	s4pm_insert_datapool.pl, GdDlBatchInsert.pl
Configuration Files:	station.cfg
Log Files:	station.log
Databases:	None
Failure Handlers:	Restart, Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-43. Insert Datapool Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	None	PDR	Export
Output:	None	PAN	Receive PAN

Table 5-44. Insert Datapool Station Work Orders

The Insert Datapool station is responsible for inserting output products to the ECS Datapool. Products inserted to the Datapool in this manner are handled as non-ECS data, that is, their data types do not require ESDTs.

The actual job of inserting the data to the Datapool is carried out by the GdDlBatchInsert.pl script which must be run on the Datapool machine. This script, in turn, uses an ECS-provided interface for inserting non-ECS data into the Datapool. In actuality, data are inserted into an "action" queue. Once in the action queue, the inserts are handled the same way they are for ECS data (those with ESDTs).

The Insert Datapool station was designed to produce the same short or long PANs that are produced by the ECS Ingest subsystem. The advantage of this design was that the Receive PAN station did not need to be redesigned to handle a new interface.

5.21 Ship Data (ship_data)

Processing Domains:	On-Demand
Scripts:	s4pm_ship_data.pl, gd_ProcIncrementEcAcRequestId.ksh, gd_ProcIncrementOrderId.ksh
Configuration Files:	station.cfg, s4pm_ship_data.cfg
Log Files:	station.log, DCLI.log
Databases:	None
Failure Handlers:	Restart, Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-45. Ship Data Station Components

On-Demand Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	SHIP	PDR	Track Requests
Output:	CLOSE	PDR	Track Requests
	UPDATE	Pathname	Track Data

Table 5-46. Ship Data Station Work Orders

The Ship Data station is used only in on-demand processing. Note that by default, as of release 5.8.1, on-demand requests coming from the V0 Data Gateway use the Export station rather than Ship Data (the Ship Data station will likely get subsumed into the Export station in some future release).

The role of Ship Data is to ship data processed on behalf of a user's request to that user. The Ship Data station receives a SHIP work order from Track Requests when an order has been completed. Using information on the user contained within the SHIP work order, Ship Data obtains an ECS order ID and request ID from the MSS database and using these IDs, submits a distribution request via DCLI (distribution command-line interface), an ECS-provided tool. The DCLI then routes the data to the user using the distribution services in ECS.

5.22 Receive PAN (receive_pan)

Processing Domains:	Standard
Scripts:	s4pm_receive_pan.pl
Configuration Files:	station.cfg
Log Files:	station.log
Databases:	None
Failure Handlers:	Resubmit PDR, Restart, Remove Job
Manual Overrides:	None
Interfaces:	Restart All Failed Jobs

Table 5-47. Receive PAN Station Components

Standard Processing			
	Work Order Name	**Format**	**To / From Station**
Input:	PAN, PDRD	Unique	ECS Ingest
	PAN, PDRD	Unique	ECS Distribution
	PAN, PDRD	Unique	Insert Datapool
Output:	UPDATE	Pathname	Track Data

Table 5-48. Receive PAN Station Work Orders

In previous releases of S4PM, the Receive PAN was not used in on-demand processing strings. As of release 5.8.1, the Receive PAN station is used to support External Product Dispatcher (EPD).

The Receive PAN station is responsible for closing the loop on data products exported to the ECS via the Export station. It does so by waiting for a Product Acceptance Notification indicating successful ingestion of data into ECS archive, insert to the ECS Datapool, or distribution via EPD. On receipt, it sends an UPDATE work order to the Track Data station allowing the data to be cleaned (if all other uses have been satisfied).

There are several types of notifications that can be received by the Receive PAN station: short PAN, long PAN, and a PDRD or Product Delivery Record Discrepancy. A short PAN reports an all or nothing status of the data products exported via a three line file. Either all products successfully ingested (or distributed via EPD) or they all failed. A long PAN is received when some products failed ingest or EPD while others succeeded; it will contain the status of each. A PDRD means that none of the products were ingested or distributed and usually indicates a syntax bug in the PDR or an improper configuration of ECS ingest. Such a notification only comes from ECS Ingest, not from Insert Datapool.

When a short PAN indicating total failure or a long PAN indicating partial failure is received in Receive PAN, the job fails. The operator has the option of running the Resubmit PDR failure handler to have the PDR get rerouted through the Export station again, presumably after the ingest problem has been resolved. This failure handler has the capability to only resubmit the products that failed, in the case of a long PAN, and omit the products that already succeeded.

5.23 Repeat Hourly (repeat_hourly) and Repeat Daily (repeat_daily)

Repeat Hourly and Repeat Daily are utility stations. They simulate cron jobs, repeating scripts on a hourly or daily interval to execute a number of tasks within S4PM. These tasks are currently configured:

- Clean expired data:
  Data not cleaned out due to the uses never reaching zero are periodically cleaned out. Each data type is configured for a maximum residence time. This ensures that data do not build up over time.
- Statistics roll up:
  Periodically, the performance statistics generated by Run Algorithm need to be rolled up into summary statistics and stored as small text files.
- Clean PH files:
  Production History files generated by Run Algorithm are not tracked the same way as data files. Therefore, they have to be periodically cleaned out.
- Clean other assorted files:
  Other files, such as certain log files, need to be periodically clean out beyond a certain age.

6. S4PM Work Orders

Generally, each work order will have a unique script associated with it. Before getting into the details of work order formats, the table below lists work order formats and work orders. Work order formats are discussed in more detail next.

Format	Work Orders
PDR	DAN (from S4PA) EXPORT FIND_*algorithm* FIND_*service* SELECT_*algorithm* REQUEST_DATA REGISTER PREPARE_*algorithm* PREPARE_*service*
PCF	ALLOCATE_*algorithm* ALLOCATE_*service* RUN_*algorithm* RUN_*service*
Pathname List	SWEEP INSERT UPDATE EXPECT
E-Mail	Distribution Notification (DN) Insert Notification Product Acceptance Notification (PAN)
ODL	SERVICE
Other	UPDATE_POOLS RECONFIG RESTART STOP MODIFY_MAX_JOBS

Table 6-1. Formats used by S4PM work orders

6.1 Work Order Names

Input work order names are typically of the form DO.<jobtype>.<jobid>.wo. Exceptions are made for the suffix where a work order is naturally another type of file, such as a Process Control File (.pcf). The jobtype field is used by stationmaster to decide which script to use. The jobid field is used to ensure uniqueness. In most cases, output work orders should propagate the jobid in the output work orders sent to downstream stations.

Example work order names:

- DO.FIND_MoPGE03.2003283100500.wo
- DO.SELECT_MoPGE03.2003283105500.wo
- DO.ALLOCATE_GdPGE02B.2003283111000.pcf

- DO.CLEAN_PRODHIST.0.6901473.wo

6.2 Work Order Formats

Generally speaking, work order formats should be as simple and as readable as possible (Simplicity, Design-for-Trouble rules). Also, the number of work order formats should be kept to a minimum by having multiple stations use the same format.

All work orders are ASCII files. Lines beginning with a '#' sign are comments, and are ignored by the stations reading them.

The work order types used in S4PM are PDR or PDR-style, Process Control File (PCF), pathname list, and others.

6.2.1 PDR Format

This format is based on the ECS Product Delivery Record which uses ODL (Object Data Language) to define FILE_GROUP and FILE_SPEC objects. For S4PM, however, a number of attributes have been added at the PDR and FILE_GROUP levels to allow more information to be passed:

	New Attribute	Description
PDR	PROCESSING_START	Algorithm processing start time and date
	PROCESSING_STOP	Algorithm processing stop time and date
	PRE__PROCESSING_OFFSET	Offset to be applied to the processing time before that time is used to compute the times of other input data needed
	POST_PROCESSING_OFFSET	Offset to be applied to the processing time only after that time is used to compute the times of other input data needed
FILE_GROUP	UR	UR is either the Universal Reference retrieved from the ECS Science Data Server or, for data produced within S4PM, the data file's LocalGranuleID (LGID)
	NEED	Represents whether the input file is required (NEED=REQ*n*) or optional (NEED=OPT*n*) where *n* represents the order of preference with 1 being the most desired
	LUN	LUN refers to the logical unit number for that input file in the process control file (PCF) used during the running of the algorithm
	DATA_START	Start date and time of the data file
	DATA_END	End date and time of the data file
	TIMER	The number of seconds to wait for that data before giving up on it. For required input, the timer starts once the trigger data file arrives; for optional input, the timer starts once all of the required data files have arrived.
	CURRENCY	Indicates the temporal coverage of the data relative to the processing period over which the algorithm is to be run. The processing period is aligned to the temporal coverage of the trigger data file.

		CURRENCY has values like CURR (for current), PREVn (for previous), and FOLLn (for following) where n is an integer indicating how far away from the processing period it must be. Thus, PREV1 is the file just prior to the current processing period, PREV2 is the one just before that, etc.
	BOUNDARY	Sets a reference in time against which processing periods of algorithms are determined, valids include START_OF_DAY, START_OF_HOUR, START_OF_6HOUR, etc.

Table 6-2. Additional PDR attributes used by S4PM.

Stations that use PDR-Style work orders as input are Select Data, Find Data, Prepare Run, Register Local Data, and Register Data.

6.2.1.1 EXPORT

Below is an example EXPORT work order that is sent to the Export station. This particular example is the EXPORT work order produced by the run of MoPGE02 which produces five products.

```
ORIGINATING_SYSTEM=S4PM10_MO_FW;
TOTAL_FILE_COUNT=4;
EXPIRATION_TIME=2003-12-05T05:31:40Z;
OBJECT=FILE_GROUP;
    DATA_TYPE=MOD02HKM;
    DATA_VERSION=004;
    NODE_NAME=g0spg10.gsfcb.ecs.nasa.gov;
    OBJECT=FILE_SPEC;
        FILE_TYPE=SCIENCE;
        DIRECTORY_ID=/usr/ecs/terra/forward/DATA/MOD02HKM;
        FILE_SIZE=275066715;
        FILE_ID=MOD02HKM.A2003336.0235.004.2003336101752.hdf;
    END_OBJECT=FILE_SPEC;
    OBJECT=FILE_SPEC;
        FILE_TYPE=METADATA;
        DIRECTORY_ID=/usr/ecs/s4ins/terra/forward/DATA/MOD02HKM;
        FILE_SIZE=17089;
        FILE_ID=MOD02HKM.A2003336.0235.004.2003336101752.hdf.met;
    END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
OBJECT=FILE_GROUP;
    DATA_TYPE=MOD02OBC;
    DATA_VERSION=004;
    NODE_NAME=g0spg10.gsfcb.ecs.nasa.gov;
    OBJECT=FILE_SPEC;
    FILE_TYPE=SCIENCE;
        DIRECTORY_ID=/usr/ecs/ terra/forward/DATA/MOD02OBC;
        FILE_SIZE=58947413;
        FILE_ID=MOD02OBC.A2003336.0235.004.2003336101752.hdf;
    END_OBJECT=FILE_SPEC;
    OBJECT=FILE_SPEC;
        FILE_TYPE=METADATA;
        DIRECTORY_ID=/usr/ecs/ terra/forward/DATA/MOD02OBC;
        FILE_SIZE=19026;
        FILE_ID=MOD02OBC.A2003336.0235.004.2003336101752.hdf.met;
    END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
```

Figure 6-1. Sample EXPORT work order.

6.2.1.2 FIND_*algorithm*

Below is an example FIND_*algorithm* work order that is sent to the Find Data station in standard processing. Only the first file group in this PDR file is completely filled out with file names, directory paths, and UR. That is because this first file group represents the trigger data type (in this case, MOD03) for this algorithm (in this case, MoPGE02). The remaining file groups have placeholders for file names, directory paths, and UR. The job of Find Data is to fill out these file groups. It does so by making use of the information that is provided in these file groups (e.g. DATA_START, DATA_END, CURRENCY, NEED) to generate a file name pattern with which to search for and wait on data.

```
TOTAL_FILE_COUNT=5;
PROCESSING_START=2003-11-30T01:55:00Z;
PROCESSING_STOP=2003-11-30T02:00:00Z;
POST_PROCESSING_OFFSET=0;
PRE_PROCESSING_OFFSET=0;
OBJECT=FILE_GROUP;
        DATA_TYPE=MOD03;
        DATA_VERSION=004;
        NODE_NAME=g0spg10.gsfcb.ecs.nasa.gov;
        UR=LGID:MOD03:004:MOD03.A2003334.0155.004.2003334094940.hdf;
        NEED=REQ1;
        LUN=600000;
        DATA_START=2003-11-30T01:55:00Z;
        DATA_END=2003-11-30T02:00:00Z;
        TIMER=0;
        CURRENCY=CURR;
        BOUNDARY=START_OF_DAY;
        OBJECT=FILE_SPEC;
                FILE_TYPE=SCIENCE;
                DIRECTORY_ID=/usr/ecs/terra/forward/DATA/MOD03;
                FILE_SIZE=60664640;
                FILE_ID=MOD03.A2003334.0155.004.2003334094129.hdf;
        END_OBJECT=FILE_SPEC;
        OBJECT=FILE_SPEC;
                FILE_TYPE=METADATA;
                DIRECTORY_ID=/usr/ecs/ forward/DATA/MOD03;
                FILE_SIZE=12245;
                FILE_ID=MOD03.A2003334.0155.004.2003334094129.hdf.met;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
OBJECT=FILE_GROUP;
        DATA_TYPE=MOD01;
        DATA_VERSION=004;
        UR=INSERT_UR_HERE;
        NEED=REQ1;
        LUN=500001;
        DATA_START=2003-11-30T01:55:00Z;
        DATA_END=2003-11-30T02:00:00Z;
        TIMER=7200;
        CURRENCY=CURR;
        BOUNDARY=START_OF_DAY;
        OBJECT=FILE_SPEC;
                FILE_TYPE=SCIENCE;
                DIRECTORY_ID=INSERT_DIRECTORY_HERE;
                FILE_ID=INSERT_FILE_HERE;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
OBJECT=FILE_GROUP;
        DATA_TYPE=MOD01;
        DATA_VERSION=004;
        UR=INSERT_UR_HERE;
        NEED=OPT1;
        LUN=500000;
        DATA_START=2003-11-30T01:50:00Z;
        DATA_END=2003-11-30T01:55:00Z;
        TIMER=7200;
        CURRENCY=PREV1;
```

```
        BOUNDARY=START_OF_DAY;
        OBJECT=FILE_SPEC;
                FILE_TYPE=SCIENCE;
                DIRECTORY_ID=INSERT_DIRECTORY_HERE;
                FILE_ID=INSERT_FILE_HERE;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
```

Figure 6-2. Sample FIND_algorithm work order.

6.2.1.3 FIND_*service*

Below is an example of a FIND_*service* work order that is sent to the Find Data station in on-demand processing. The FIND_*service* work order differs from the standard FIND_algorithm work order in these respects:

1. The trigger data type is always PSPEC.
2. All file names are fully qualified although, except for the PSPEC, the directory locations are not.
3. The processing start and processing stop times are not needed, so they are fixed to 00:00 on January 1, 1970.

```
TOTAL_FILE_COUNT=2;
PROCESSING_START=1970-01-01T00:00:00Z;
PROCESSING_STOP=1970-01-01T00:00:00Z;
POST_PROCESSING_OFFSET=0;
PRE_PROCESSING_OFFSET=0;
OBJECT=FILE_GROUP;
        DATA_TYPE=PSPEC;
        DATA_VERSION=001;
        UR=PSPEC_GdMODl1B.100_1;
        NEED=REQ1;
        LUN=911;
        TIMER=7200;
        CURRENCY=CURR;
        BOUNDARY=START_OF_DAY;
        OBJECT=FILE_SPEC;
                FILE_ID=PSPEC_GdMODL1B.100_1;
                FILE_TYPE=SCIENCE;

DIRECTORY_ID=/data2/on_demand/stations/neutral/DATA/INPUT;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
OBJECT=FILE_GROUP;
        DATA_TYPE=MOD021KM;
        DATA_VERSION=004;
        UR=INSERT_UR_HERE;
        NEED=REQ1;
        LUN=22210;
        TIMER=7200;
        CURRENCY=CURR;
        BOUNDARY=START_OF_DAY;
        OBJECT=FILE_SPEC;
                FILE_ID=MOD021KM.A2003001.0235.004.2003001111017.hdf;
                FILE_TYPE=SCIENCE;
                DIRECTORY_ID=INSERT_DIRECTORY_HERE;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
```

Figure 6-3. Sample FIND_*service* work order.

6.2.1.4 SELECT_*algorithm*

Below is an example SELECT_*algorithm* work order that is sent to the Select Data station in standard processing. The work order in this example is SELECT_MoPGE02.

```
TOTAL_FILE_COUNT=2;
OBJECT=FILE_GROUP;
        DATA_TYPE=MOD03;
        DATA_VERSION=004;
        NODE_NAME=g0spg10.gsfcb.ecs.nasa.gov;
        UR=LGID:MOD03:004:MOD03.A2003334.0155.004.2003334094940.hdf;
        DATA_START=2003-11-30T01:55:00Z;
        DATA_END=2003-11-30T02:00:00Z;
        OBJECT=FILE_SPEC;
                DIRECTORY_ID=/usr/ecs/terra/forward/DATA/MOD03;
                FILE_TYPE=SCIENCE;
                FILE_SIZE=60664640;
                FILE_ID=MOD03.A2003334.0155.004.2003334094129.hdf;
        END_OBJECT=FILE_SPEC;
        OBJECT=FILE_SPEC;
                DIRECTORY_ID=/usr/ecs/ terra/forward/DATA/MOD03;
                FILE_TYPE=METADATA;
                FILE_SIZE=12245;
                FILE_ID=MOD03.A2003334.0155.004.2003334094129.hdf.met;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
```

Figure 6-4. Sample SELECT work order.

6.2.1.5 REQUEST_DATA

Below is an example REQUEST_DATA work order that is sent to the Request Data station. The work order in this example is REQUEST_DATA_AM1EPHN0. This work order could have been generated automatically via a Subscription Notification going to the Subscription Notify station or via the use of the Compose Request tool.

```
ORIGINATING_SYSTEM=S4PM;
TOTAL_FILE_COUNT=1;
OBJECT=FILE_GROUP;
        DATA_TYPE=AM1EPHN0;
        DATA_VERSION=001;
UR=UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF]:24:SC:AM1EPH.001:32;
        DATA_START=2003-12-03T04:00:00Z;
        DATA_END=2003-12-03T05:59:59Z;
        OBJECT=FILE_SPEC;
                FILE_TYPE=SCIENCE;
                DIRECTORY_ID=INSERT_DIRECTORY_HERE;
                FILE_SIZE=474158;
                FILE_ID=AM1EPHN0.A2003337.0400.001.2003337144738;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
```

Figure 6-5. Sample REQUEST_DATA work order.

6.2.1.6 REGISTER

Below is an example REGISTER work order that is sent to the Register Data station. Here, it is for the arrival of Level-0 data. The REGISTER work orders going to the Register Local Data station are similar.

```
TOTAL_FILE_COUNT=6;
OBJECT=FILE_GROUP;
        DATA_TYPE=MOD000;
        DATA_VERSION=001;
UR=UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF]:22:SC:MOD0.001:85;
        OBJECT=FILE_SPEC;
                FILE_TYPE=SCIENCE;
                DIRECTORY_ID=/usr/ecs/terra/forward/DATA/INPUT;
                FILE_SIZE=1999647270;
                FILE_ID=P0420064AAAAAAAAAAAAAA03248064247001.PDS;
        END_OBJECT=FILE_SPEC;
        OBJECT=FILE_SPEC;
                FILE_TYPE=SCIENCE;
                DIRECTORY_ID=/usr/ecs/forward/DATA/INPUT;
                FILE_SIZE=3068;
                FILE_ID=P0420064AAAAAAAAAAAAAA03248064247000.PDS;
        END_OBJECT=FILE_SPEC;
        OBJECT=FILE_SPEC;
                FILE_TYPE=METADATA;
                DIRECTORY_ID=/usr/ecs/forward/DATA/INPUT;
                FILE_SIZE=28762;
                FILE_ID=P0420064AAAAAAAAAAAAAA03248064247001.PDS.met;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
```

Figure 6-6. Sample REGISTER work order.

6.2.1.7 PREPARE_*algorithm*

Below is an example PREPARE_*algorithm* work order that is sent to the Prepare Run station in standard processing. In this case, it is a work order that will result in the running of MoPGE02. All fields in the work order have been filled out. The job of Prepare Run is to allocate space for the output files (which are not in this work order) and generate a runtime PCF with which MoPGE02 will run.

```
TOTAL_FILE_COUNT=4;
PROCESSING_START=2003-09-05T03:35:00Z;
PROCESSING_STOP=2003-09-05T03:40:00Z;
PROCESSING_OFFSET=0;
POST_PROCESSING_OFFSET=0;
PRE_PROCESSING_OFFSET=0;
OBJECT=FILE_GROUP;
        DATA_TYPE=MOD03;
        DATA_VERSION=004;
        NODE_NAME=g0spg10.gsfcb.ecs.nasa.gov;
        UR=LGID:MOD03:004:MOD03.A2003248.0335.004.2003248094932.hdf;
        NEED=REQ1;
        LUN=600000;
        DATA_START=2003-09-05T03:35:00Z;
        DATA_END=2003-09-05T03:40:00Z;
        TIMER=0;
        CURRENCY=CURR;
        BOUNDARY=START_OF_DAY;
        OBJECT=FILE_SPEC;
                FILE_TYPE=SCIENCE;
                DIRECTORY_ID=/usr/ecs/terra/forward/DATA/MOD03;
                FILE_SIZE=60664637;
                FILE_ID=MOD03.A2003248.0335.004.2003248092611.hdf;
        END_OBJECT=FILE_SPEC;
        OBJECT=FILE_SPEC;
                FILE_TYPE=METADATA;
                DIRECTORY_ID=/usr/ecs/terra/forward/DATA/MOD03;
                FILE_SIZE=12242;
                FILE_ID=MOD03.A2003248.0335.004.2003248092611.hdf.met;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
OBJECT=FILE_GROUP;
        DATA_TYPE=MOD01;
        DATA_VERSION=004;
        UR=LGID:MOD01:004:MOD01.A2003248.0335.004.2003248094105.hdf;
        NEED=REQ1;
        LUN=500001;
        DATA_START=2003-09-05T03:35:00Z;
        DATA_END=2003-09-05T03:40:00Z;
        TIMER=7200;
        CURRENCY=CURR;
        BOUNDARY=START_OF_DAY;
        OBJECT=FILE_SPEC;
                FILE_TYPE=SCIENCE;
                DIRECTORY_ID=/usr/ecs/terra/forward/DATA/MOD01;
                FILE_SIZE=574137409;
                FILE_ID=MOD01.A2003248.0335.004.2003248092611.hdf;
        END_OBJECT=FILE_SPEC;
        OBJECT=FILE_SPEC;
                FILE_TYPE=METADATA;
                DIRECTORY_ID=/usr/ecs/terra/forward/DATA/MOD01;
                FILE_SIZE=7356;
                FILE_ID=MOD01.A2003248.0335.004.2003248092611.hdf.met;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
```

Figure 6-7. Sample PREPARE_algorithm work order.

6.2.1.8 PREPARE_*service*

Below is an example PREPARE_*service* work order that is sent to the Prepare Run station in on-demand processing. In this case, it is a work order that will result in the running of the GdMODL1B subsetting service. The PREPARE_*service* work order is essentially identical to the PREPARE_*algorithm* work order except that in a PREPARE_*service* work order you will always find the PSPEC file associated with LUN 911.

```
TOTAL_FILE_COUNT=3;
PROCESSING_START=2002-07-05T01:00:00Z;
PROCESSING_STOP=2002-07-05T01:05:00Z;
PROCESSING_OFFSET=0;
POST_PROCESSING_OFFSET=0;
OBJECT=FILE_GROUP;
        DATA_TYPE=MOD021KM;
        DATA_VERSION=004;
UR=UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF]:22:SC:MOD021KM.004:2
3;
        NEED=REQ;
        LUN=22210;
        TIMER=7200;
        CURRENCY=CURR;
        BOUNDARY=START_OF_DAY;
        OBJECT=FILE_SPEC;
                FILE_SIZE=142713283;
                FILE_TYPE=SCIENCE;
                FILE_ID=MOD021KM.A2002186.0100.004.2003119204955.hdf;
                DIRECTORY_ID=/usr/ecs/on_demand/neutral/DATA/INPUT;
        END_OBJECT=FILE_SPEC;
        OBJECT=FILE_SPEC;
                FILE_SIZE=56465;
                FILE_TYPE=METADATA;

FILE_ID=MOD021KM.A2002186.0100.004.2003119204955.hdf.met;
                DIRECTORY_ID=/usr/ecs/on_demand/neutral/DATA/INPUT;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
OBJECT=FILE_GROUP;
        DATA_TYPE=PSPEC;
        DATA_VERSION=001;
        UR=INSERT_UR_HERE;
        NEED=TRIG;
        LUN=911;
        TIMER=7200;
        CURRENCY=CURR;
        BOUNDARY=START_OF_DAY;
        OBJECT=FILE_SPEC;
                FILE_TYPE=SCIENCE;
                FILE_ID=PSPEC_GdMODL1B.100_1;
                DIRECTORY_ID=/usr/ecs/on_demand/neutral/DATA/INPUT;
        END_OBJECT=FILE_SPEC;
END_OBJECT=FILE_GROUP;
```

Figure 6-8. Sample PREPARE_service work order.

6.2.2 PCF Format

This format is based upon the ECS Toolkit Process Control File or PCF format. Stations that use this format for their input work orders are Allocate Disk and Run Algorithm.

6.2.2.1 ALLOCATE_*algorithm*

Below is an example ALLOCATE_*algorithm* work order that is sent to the Allocate Disk station in standard processing. Note that the PRODUCT INPUT FILES section is filled out completely with file names and directories while the PRODUCT OUTPUT FILES section is not. There, placeholders have been inserted where directory names are needed. The job of Allocate Disk station is to allocate disk for these output files and then replace these placeholders with actual directory names.

```
#
?   PRODUCT INPUT FILES
#
!  ~/runtime
#
#Lookup Tables
#
700050|MOD02_Refl.hdf.coef|../MoPGE02/4.2.0/pge/static||MOD02_Refl.hdf.
coef||1
700060|MOD02_Emis.hdf.coef|../MoPGE02/4.2.0/pge/static||MOD02_Emis.hdf.
coef|1
700070|MOD02_QA.hdf.coef|../MoPGE02/4.2.0/pge/static||MOD02_QA.hdf.coef
||1
#
#Geolocation
#
600000|MOD03.A2003248.0335.004.2003248092611.hdf|/usr/ecs/terra/forward
/DATA/MOD03||LGI
D:MOD03:004:MOD03.A2003248.0335.004.2003248094932.hdf|MOD03.A2003248.03
35.004.2003
248092611.hdf|1
#
# L1A input files below (Also see 70020,700223,700222,201001)
#
500000|MOD01.A2003248.0330.004.2003248092611.hdf|/usr/ecs/terra/forward
/DATA/MOD01||LGI
D:MOD01:004:MOD01.A2003248.0330.004.2003248093943.hdf|MOD01.A2003248.03
30.004.2003
248092611.hdf|1
500001|MOD01.A2003248.0335.004.2003248092611.hdf|/usr/ecs/terra/forward
/DATA/MOD01||LGI
D:MOD01:004:MOD01.A2003248.0335.004.2003248094105.hdf|MOD01.A2003248.03
35.004.2003
248092611.hdf|1
500002|MOD01.A2003248.0340.004.2003248092611.hdf|/usr/ecs/terra/forward
/DATA/MOD01||LGI
D:MOD01:004:MOD01.A2003248.0340.004.2003248093923.hdf|MOD01.A2003248.03
40.004.2003
248092611.hdf|1
#
```

```
?   PRODUCT OUTPUT FILES
#
! ./
#
700000|MOD02QKM.A2003248.0335.004.2003248095325.hdf|INSERT_DIRECTORY_HE
RE||LGID
:MOD02QKM.A2003248.0335.004.2003248095325.hdf|MOD02QKM.A2003248.0335.00
4.20032480
95325.hdf.met|1
700001|MOD02HKM.A2003248.0335.004.2003248095325.hdf|INSERT_DIRECTORY_HE
RE||LGID
:MOD02HKM.A2003248.0335.004.2003248095325.hdf|MOD02HKM.A2003248.0335.00
4.20032480
95325.hdf.met|1
700002|MOD021KM.A2003248.0335.004.2003248095325.hdf|INSERT_DIRECTORY_HE
RE||LGID
:MOD021KM.A2003248.0335.004.2003248095325.hdf|MOD021KM.A2003248.0335.00
4.20032480
95325.hdf.met|1
700010|MOD02OBC.A2003248.0335.004.2003248095325.hdf|INSERT_DIRECTORY_HE
RE||LGID
:MOD02OBC.A2003248.0335.004.2003248095325.hdf|MOD02OBC.A2003248.0335.00
4.20032480
95325.hdf.met|1
700100|MOD021QA.A2003248.0335.004.2003248095325.hdf|INSERT_DIRECTORY_HE
RE||LGID
:MOD021QA.A2003248.0335.004.2003248095325.hdf|MOD021QA.A2003248.0335.00
4.200324809
5325.hdf.met|1
# MOD_PR02BR (Browse) Output File:
99201|Browse.A2003248.0335.001.2003248095325.hdf|INSERT_DIRECTORY_HERE|
|LGID:Brows
e.A2003248.0335.001.2003248095325.hdf|Browse.A2003248.0335.001.20032480
95325.hdf.met|1
?   USER DEFINED RUNTIME PARAMETERS
#
?   TEMPORARY I/O
#
! ./
#
?   END
```

Figure 6-9. Sample ALLOCATE_*algorithm* work order.

6.2.2.2 ALLOCATE_*service*

Below is an example ALLOCATE_*service* work order that is sent to the Allocate Disk station in on-demand processing. In all respects, it is similar to the ALLOCATE_*algorithm* work order used in standard processing. One difference, however, is the LUN 911 for the PSPEC file. The other difference is in the USER DEFINED RUNTIME PARAMETERS section where two LUNs, 20100 and 21212, are specifically set aside for specialized criteria. Two specialized criteria in the incoming SERVICE work order (going into the Split Services station) have names CHANNELS and FORMAT. The values associated with these names in that SERVICE work order

have been used to populate LUNs 20100 and 21212 in the runtime PCF. This task was performed by Prepare Run in building this ALLOCATE_*service* work order.

```
#
?   PRODUCT INPUT FILES
#######################################################################
##
!  ~/runtime
#
# At launch format
911|PSPEC_GdMODL1B.100_1|/usr/ecs/TS2/CUSTOM/g0dus02/s4ins/on_demand/ne
utral/DATA/IN
PUT||INSERT_UR_HERE|PSPEC_GdMODL1B.100_1|1
22210|MOD021KM.A2002186.0100.004.2003119204955.hdf|/usr/ecs/TS2/CUSTOM/
g0dus02/s4ins/o
n_demand/neutral/DATA/INPUT||UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:
[GSF:DSSDSRV
]:22:SC:MOD021KM.004:220643|MOD021KM.A2002186.0100.004.2003119204955.hd
f|1
22215|README|../GdMODL1B/1.0.0/pge/static||README||1
#---------------------------------------------------------------------
--
10250|MCF|||||1
10251|dump|./||||1
10252|MCFWrite.temp|./||||1
10254|GetAttr.temp|./||||1
#
22223|MOD021SS.mcf|../GdMODL1B/1.0.0/pge/static||||1
#
?   PRODUCT OUTPUT FILES
#######################################################################
##
# Next line is the default location for PRODUCT OUTPUT FILES
!  ./
22222|MOD021SS.A2002186.0100.001.2004035204113.hdf|INSERT_DIRECTORY_HER
E||LGID:MOD02
1SS.A2002186.0100.001.2004035204113.hdf|MOD021SS.A2002186.0100.001.2004
035204113.hdf.met|1
?   SUPPORT INPUT FILES
#######################################################################
##
# Next line is the default location for SUPPORT INPUT FILES
!  ~/runtime
#
?   SUPPORT OUTPUT FILES
#######################################################################
##
# Next line is default location for SUPPORT OUTPUT FILES
! ./
10100|LogStatus|||||1
10101|LogReport|||||1
10102|LogUser|||||1
#
?   USER DEFINED RUNTIME PARAMETERS
#######################################################################
##
20100|CHANNELS|00000000000000100000100000010000000000
```

```
21212|FORMAT|HDFEOS
#
?   INTERMEDIATE INPUT
#####################################################################
##
#
?   INTERMEDIATE INPUT
#####################################################################
##
#
# This section is intended for intermediate input files, i.e., files
# which are output by an earlier PGE but which are not standard
# products.
#
# Each logical ID may have only one file instance.
# Last field on the line is ignored.
#
#####################################################################
##
#
# Next line is default location for INTERMEDIATE INPUT FILES
!  ~/runtime
#
?   INTERMEDIATE OUTPUT
#####################################################################
##
# Next line is default location for INTERMEDIATE OUTPUT FILES
!  ~/runtime
#
?   TEMPORARY I/O
#####################################################################
##
# Next line is default location for TEMPORARY FILES
!  ./
```

Figure 6-10. Sample ALLOCATE_*service* work order.

6.2.2.3 RUN_*algorithm*

Below is an example RUN_*algorithm* work order that is sent to the Run Algorithm station. It is, in fact, the same work order that went to the Allocate Disk station above, but now the output directories have been filled out. The work order is now a full-fledged PCF.

```
#
?   PRODUCT INPUT FILES
#
!  ~/runtime
#
#Lookup Tables
#
700050|MOD02_Refl.hdf.coeff|../MoPGE02/4.2.0/pge/static||MOD02_Refl.hdf
.coef||1
700060|MOD02_Emis.hd.coef|../MoPGE02/4.2.0/pge/static||MOD02_Emis.hdf.c
oef||1
```

```
700070|MOD02_QA.hd.coef|../MoPGE02/4.2.0/pge/static||MOD02_QA.hdf.coef|
|1
#
#Geolocation
#
600000|MOD03.A2003248.0335.004.2003248092611.hdf|/usr/ecs/terra/forward
/DATA/MOD03||LGID:
MOD03:004:MOD03.A2003248.0335.004.2003248094932.hdf|MOD03.A2003248.0335
.004.2003248
092611.hdf|1
#
# L1A input files below (Also see 70020,700223,700222,201001)
#
500000|MOD01.A2003248.0330.004.2003248092611.hdf|/usr/ecs/terra/forward
/DATA/MOD01||LGID:
MOD01:004:MOD01.A2003248.0330.004.2003248093943.hdf|MOD01.A2003248.0330
.004.2003248
092611.hdf|1
500001|MOD01.A2003248.0335.004.2003248092611.hdf|/usr/ecs/terra/forward
/DATA/MOD01||LGID:
MOD01:004:MOD01.A2003248.0335.004.2003248094105.hdf|MOD01.A2003248.0335
.004.2003248
092611.hdf|1
500002|MOD01.A2003248.0340.004.2003248092611.hdf|/usr/ecs/terra/forward
/DATA/MOD01||LGID:
MOD01:004:MOD01.A2003248.0340.004.2003248093923.hdf|MOD01.A2003248.0340
.004.2003248
092611.hdf|1
#
?   PRODUCT OUTPUT FILES
#
! ./
#
700000|MOD02QKM.A2003248.0335.004.2003248095325.hdf|/usr/ecs/terra/forw
ard/DATA/MOD02Q
KM||LGID:MOD02QKM.A2003248.0335.004.2003248095325.hdf|MOD02QKM.A2003248
.0335.004.2
003248095325.hdf.met|1
700001|MOD02HKM.A2003248.0335.004.2003248095325.hdf|/usr/ecs/terra/forw
ard/DATA/MOD02H
KM||LGID:MOD02HKM.A2003248.0335.004.2003248095325.hdf|MOD02HKM.A2003248
.0335.004.2
003248095325.hdf.met|1
700002|MOD021KM.A2003248.0335.004.2003248095325.hdf|/usr/ecs/terra/forw
ard/DATA/MOD021
KM||LGID:MOD021KM.A2003248.0335.004.2003248095325.hdf|MOD021KM.A2003248
.0335.004.2
003248095325.hdf.met|1
700010|MOD02OBC.A2003248.0335.004.2003248095325.hdf|/usr/ecs/terra/forw
ard/DATA/MOD02O
BC||LGID:MOD02OBC.A2003248.0335.004.2003248095325.hdf|MOD02OBC.A2003248
.0335.004.2
003248095325.hdf.met|1
700100|MOD021QA.A2003248.0335.004.2003248095325.hdf|/usr/ecs/terra/forw
ard/DATA/MOD021
QA||LGID:MOD021QA.A2003248.0335.004.2003248095325.hdf|MOD021QA.A2003248
.0335.004.2
003248095325.hdf.met|1
```

```
# MOD_PR02BR (Browse) Output File:
99201|Browse.A2003248.0335.001.2003248095325.hdf|/usr/ecs/terra/forward
/DATA/Browse||LGID:
Browse.A2003248.0335.001.2003248095325.hdf|Browse.A2003248.0335.001.200
3248095325.hdf.met|1
?   USER DEFINED RUNTIME PARAMETERS
#
?   TEMPORARY I/O
#
! ./
#
?   END
```

Figure 6-11. Sample RUN_*algorithm* work order.

6.2.2.4 RUN_service

Figure 6-12. Sample RUN_*service* work order.

6.2.3 Pathname List Format

This type of work order is simply a list of full pathnames of files to be processed. This type of work order is used by Sweep Data.

6.2.3.1 SWEEP

Below is an example SWEEP work order that is sent to the Sweep Data station. This one in particular resulted from the successful export of products from MoPGE03. The work order merely contains the full pathnames of those products. The Sweep Data station will physically delete each from disk after reclaiming reserved disk space.

```
/usr/ecs/OPS/CUSTOM/g0spg10/s4ins/terra/forward/DATA/MODCSR_G/MODCSR_G.
A2003331.061
0.004.2003331144742.hdf
/usr/ecs/OPS/CUSTOM/g0spg10/s4ins/terra/forward/DATA/MOD35_QC/MOD35_QC.
A2003331.061
0.004.2003331144742.hdf
/usr/ecs/OPS/CUSTOM/g0spg10/s4ins/terra/forward/DATA/MOD07_L2/MOD07_L2.
A2003331.0610
.004.2003331144742.hdf
/usr/ecs/OPS/CUSTOM/g0spg10/s4ins/terra/forward/DATA/MODVOLC/MODVOLC.A2
003331.0610
.004.2003331144742.hdf
/usr/ecs/OPS/CUSTOM/g0spg10/s4ins/terra/forward/DATA/MOD35_L2/MOD35_L2.
A2003331.0610
.004.2003331144742.hdf
```

Figure 6-12. Sample SWEEP work order.

6.2.3.2 INSERT

Below is an example INSERT work order that is sent to the Track Data station. The one below resulted from the arrival of Level-0 data. The uses for each are the maximum number of times each file will be used. As each file is used, the uses for these files will be decremented appropriately. When the uses reach zero for any file, it will be deleted from the system.

```
FileId=/usr/ecs/TS2/CUSTOM/g0spg10/s4ins/terra/reprocessing/DATA/INPUT/
AM1EPHN0.A2000270.06
00.001.2001172213441 Uses=9
FileId=/usr/ecs/TS2/CUSTOM/g0spg10/s4ins/terra/reprocessing/DATA/INPUT/
GDAS_0ZF.A2000270.03
00.001.2000271023141 Uses=72
FileId=/usr/ecs/TS2/CUSTOM/g0spg10/s4ins/terra/reprocessing/DATA/INPUT/
MOD000.A2000270.0600
.001.2000285124229 Uses=9
FileId=/usr/ecs/TS2/CUSTOM/g0spg10/s4ins/terra/reprocessing/DATA/INPUT/
AM1ATTN0.A2000270.06
00.001.2001172213440 Uses=9
```

Figure 6-13. Sample INSERT work order.

6.2.3.3 UPDATE

Below is an example UPDATE work order that is sent to the Track Data station. In this example, the uses for each file are decremented by 1 in the station's database file, uses.db.

```
FileId=/g0spg10/s4ins/terra/reprocessing/DATA/INPUT/AM1ATTN0.A2000270.0
600.001.2001172213440 Uses=-1
FileId=/g0spg10/s4ins/terra/reprocessing/DATA/INPUT/MOD000.A2000270.060
0.001.2000285124229 Uses=-1
```

```
FileId=*MOD021KM*.hdf Uses=+5
FileId=*MOD02HKM*.hdf Uses=+1
```

Figure 6-14. Sample UPDATE work orders. The second example uses the wildcard character '*' to match zero or more files in the Track Data database. This latter type of work order is sent by the Algorithm Installation tool when a new algorithm uses input that is already extant in S4PM.

6.2.3.4 EXPECT

An EXPECT work order looks exactly like an INSERT or UPDATE work order except it is used to alert Track Data to expect a future INSERT work order. The EXPECT work order is used only in on-demand processing and it prevents unnecessary re-ordering of data that has already been ordered (say, by another on-demand user).

```
FileId=MOD021KM.A2002186.0445.004.2003044155500.hdf Uses=1
FileId=MOD021KM.A2002186.0105.004.2002248170101.hdf Uses=1
FileId=MOD021KM.A2002186.0100.004.2003119204955.hdf Uses=1
FileId=MOD021KM.A2003103.2115.004.2003104040923.hdf Uses=1
```

Figure 6-15. Sample EXPECT work order.

6.2.4 E-Mail Format

This is not really a format per se, but instead represents different formats that are distributed via e-mail (although other options exist such as FTP). Distribution Notifications and Insert Notifications are sent through e-mail and processed in S4PM via procmail whereas PANs (and their ilk) are FTP pushed.

6.2.4.1 Distribution Notification (DN)

Below is an example Distribution Notification work order that is sent to the Receive DN station. The DN is actually emailed to the S4PM user and placed into the Receive DN station directory as a work order by procmail.

```
From allmode@machine2.ecs.nasa.gov  Mon Aug 12 09:16:24 2002
Date: Mon, 12 Aug 2002 09:15:28 -0400 (EDT)
Message-Id: <200208121315.JAA41947@ecs.nasa.gov>
To: <s4pmops@machine2.ecs.nasa.gov>
Subject: ECS Notification

Dear User,

Your FTP product request has been processed for the data file(s)
listed below. The files associated with the product(s) have been sent
to your site using FTP push.

Useful documents are located at
http://daac.gsfc.nasa.gov/MODIS/index.sshtml
For more information or to report a problem, please contact the GES
DAAC User Services at

GES DAAC ECS User Services
NASA GSFC Code 610.2
Greenbelt, MD 20771
Telephone (301) 555-5393
FAX (301) 555-5218
daac_usg@gsfcsrvr4.gsfcmo.ecs.nasa.gov

++++++++++

ORDERID: NONE
REQUESTID: NONE
USERSTRING: S4PM10_MO_FW
FINISHED: 08/12/2002 09:14:32

MEDIATYPE: FtpPush
FTPHOST: g0spg10.gsfcb.ecs.nasa.gov
FTPDIR: /usr/ecs/OPS/CUSTOM/g0spg10/s4ins/terra/forward/DATA/INPUT
MEDIA 1 of 1
MEDIAID:

        GRANULE:
UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF:DSSDSRV]:24:SC:GDAS_0ZF.
001:13155625
        ESDT: GDAS_0ZF.001
```

```
FILENAME: gdas1.PGrbF00.020802.06z
FILESIZE: 22883916

FILENAME: gdas1.PGrbF00.020802.06z.met
FILESIZE: 14937
```

Figure 6-16. Sample Distribution Notification (DN) work order.

6.2.4.2 Insert Notification

Below is an example Insert Notification work order that is sent to the Subscription Notification station. As with a DN, the Insert Notification is emailed to the S4PM user, processed by procmail and deposited into the station directory.

```
From allmode@machine2.ecs.nasa.gov  Wed Jun  4 05:25:49 2003
Date: Wed, 4 Jun 2003 05:24:51 -0400 (EDT)
Message-Id: <200306040924.GAA03406@ecs.nasa.gov>
To: <s4pmops@machine2.ecs.nasa.gov>
Subject: ECS Notification for Event: RMT000.001:INSERT

Subscription Notification:
UR:10:DsShESDTUR:UR:15:DsShSciServerUR:13:[GSF:DSSDSRV]:22:SC:RMT000.00
1:28245773

ESDT Information:  RMT000.1:INSERT

User Information:  s4pmops

User String:  RMT000.001 Insert

Subscription ID:  5137

Qualifier List:

======METADATA SECTION======

dbID = 28245773
type = SC
subType = RMT000.001
primaryCollectionId = 15908651
processingHistoryId =
lastUpdate = Jun  4 2003  5:19AM
deleteEffectiveDate =
BeginningDateTime = Jun  3 2003  7:32PM
EndingDateTime = Jun  3 2003  7:37PM
DeleteFromArchive = N
insertTime = Jun  4 2003  5:19AM
ShortName = RMT000
VersionID = 1
SizeMBECSDataGranule = 377.42
ReprocessingPlanned =
ReprocessingActual =
LocalGranuleID = RMT000.A2003154.1932.001.2003154124422
DayNightFlag =
```

```
ProductionDateTime = Jun  3 2003 12:44PM
LocalVersionID =
PGEVersion =
RangeEndingTime = 19:37:13.5813140
RangeEndingDate = Jun  3 2003 12:00AM
RangeBeginningTime = 19:32:27.3266300
RangeBeginningDate = Jun  3 2003 12:00AM
TimeOfDay =
CalendarDate =
ZoneIdentifier =

DirectReadoutStation = ORST-OPEL

internalFileName = :SC:RMT000.001:28245773:1.CCSDS
filePath =
userDataFile = P0420064AAAAAAAAAAAAAA03154124422000.PDS
checkSum  = 0
fileSize = 384
creationDate =

internalFileName = :SC:RMT000.001:28245773:2.CCSDS
filePath =
userDataFile = P0420064AAAAAAAAAAAAAA03154124422001.PDS
checkSum  = 0
fileSize = 377419602
creationDate =
```

Figure 6-17. Sample Insert Notification work order.

6.2.4.3 Product Acceptance Notification (PAN)

Below is an example Product Acceptance Notification work order that is sent to the Receive PAN station. These work orders can be short PANs (as in this example), long PANs or PDRDs (the latter two indicating some sort of ingest failure).

```
MESSAGE_TYPE = SHORTPAN;
DISPOSITION = "SUCCESSFUL";
TIME_STAMP = 2003-10-19T03:43:04Z;
```

Figure 6-19. Sample Product Acceptance Notification (PAN) work order.

6.2.5 ODL Format

TBD

6.2.5.1 SERVICE

You may view this sample SERVICE work order here.

7. Cross-Station Dependencies

Generally, S4PM stations are completely independent of other stations, with the only information exchange happening via work orders. However, there are a few cases where two or more stations need to share information, or share information in addition to the work orders.

One way of implementing this is to actually use a single station, with the information saved somewhere within the directory, and accept two different job types. In some cases, this is not possible for performance reasons, and the stations must be separated.

7.1 Export and Receive PAN

The Export station submits a Product Delivery Record to ECS. If anything goes wrong with the data insert, ECS sends back a Product Delivery Record Discrepancy or a Product Acceptance Notification. If all the files were successful, ECS does not send back the list of successful files. In this case, the Receive PAN station will need to match the PAN with the PDR in order to determine which files have passed the archive step, so that the UPDATE work order can be constructed.

Share: Product Delivery Records

Data Store: Directory with PDRs

7.2 Allocate Disk, Request Data, and Sweep Data

All of these stations use the same resource allocation database. Allocate Disk and Request Data allocate volumes from these pools, while Sweep Data frees the disk in these pool.

Share: s4pm_allocate_disk.db, s4pm_allocate_disk.cfg

Data Store: DB_File database.

7.3 Request Data and Register Data

Request Data station creates a unique directory in the reprocessing case for the data to be sent to and deposits FIND_algorithm work orders there as a single file. Register Data splits the FIND_algorithm work orders and sends them downstream to Find Data when data are received in that directory.

Share: FIND_*algorithm* work orders

Data Store: UNIX directory, composite FIND_*algorithm* work order file.

8. User Interfaces

8.1 S4PM Monitor

S4PM Monitor is a window for monitoring a set of stations. It does this by reading the station configuration file, then monitoring for directories such as RUNNING.*, FAILED.* and pending work orders like DO.*.wo. It displays these as boxes in different colors. Each row of boxes usually represents a station, but by using the syntax station:jobtype on the command line, a row can be confined to one jobtype for a given station.

More than one S4PM string can be configured on a single box, each with its own combination of stations and each with its own S4PM monitor.

Clicking on a given job directory or station box brings up the drill-down tool, Job Monitor (see below).

In the example below, the Configurator and Insert Datapool stations are off (indicated by red background with white text). The stations that are red, but with grey text are disabled. This means that those stations don't participate in the current S4PM configuration (thus, they are disabled).

There is one job in the Configurator station that has failed (also indicated by red). The Find Data station has many jobs running (all of them are green). The Prepare Run station has one job running (green box) and one running but taking longer than expected (yellow box). The Running Algorithm71 station has only one job running, but many queued up (blue boxes).

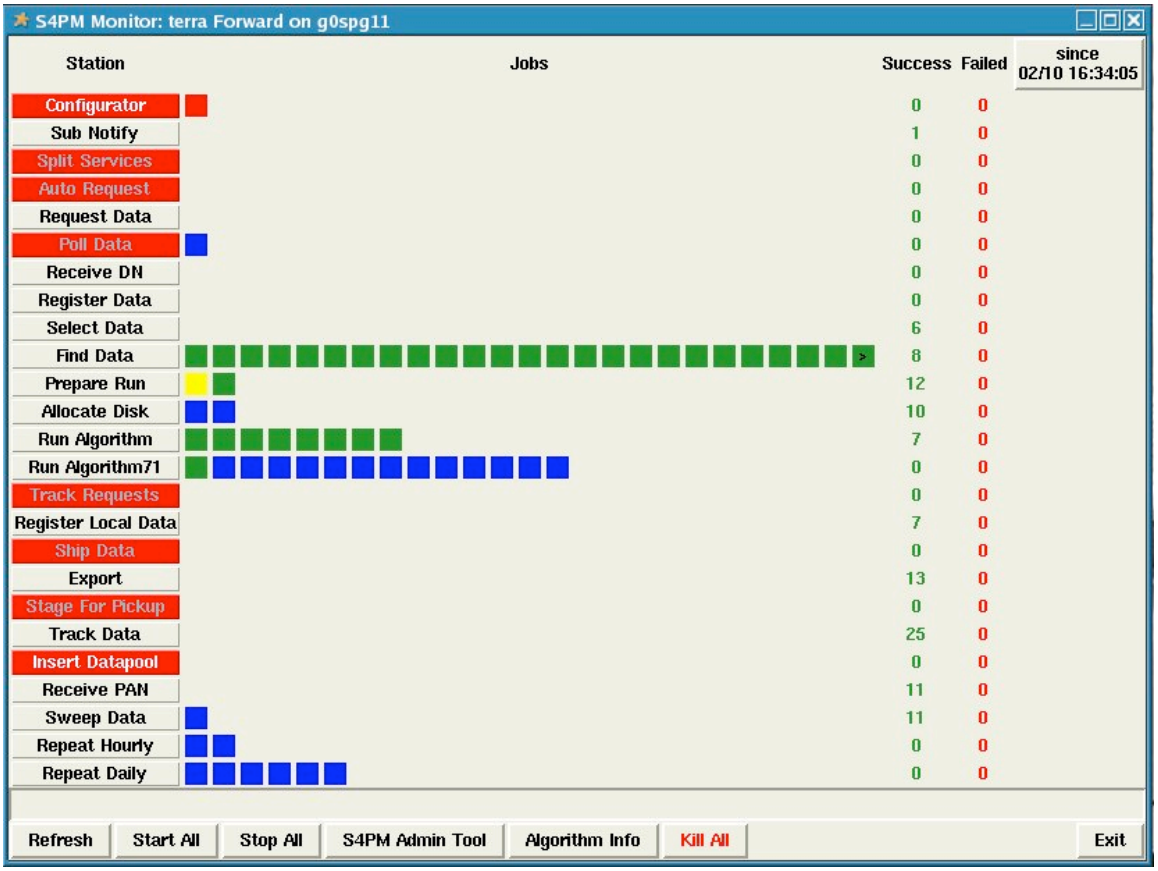


Figure 8-1. Example S4PM Monitor. Buttons along bottom are shortcuts to common tasks and tools. The Configurator and Insert Datapool stations are red indicating that those stations are not running. The Split Services, Auto Request, Poll Data, Track Requests, Ship Data, and Stage For Pickup stations are also all red, but the station names are in grey indicating that these stations are disabled in this string. The red job box in the Configurator station is a failed job. The green boxes in Find Data and Run Algorithm stations are running jobs and the blue boxes are jobs that are queued up waiting to be executed. The lone yellow box indicates that the job is taking longer to run than predicted.

8.2 Job Monitor

The Job Monitor shows the directories and files in a station or job directory. (It is actually based on a simple file browser.) In addition, it has some additional buttons specific to S4PM, allowing the operators to interface with the stations. Buttons are context-sensitive, that is, they only show up where they are applicable.

The example below shows the Job Monitor display for a particular running job in the Find Data station. Almost all stations display the Terminate, Suspend, and Resume buttons. But only in Find Data will you find the Expire Current Timers, Ignore Optional, and Ignore Required buttons since they are only applicable to this station.

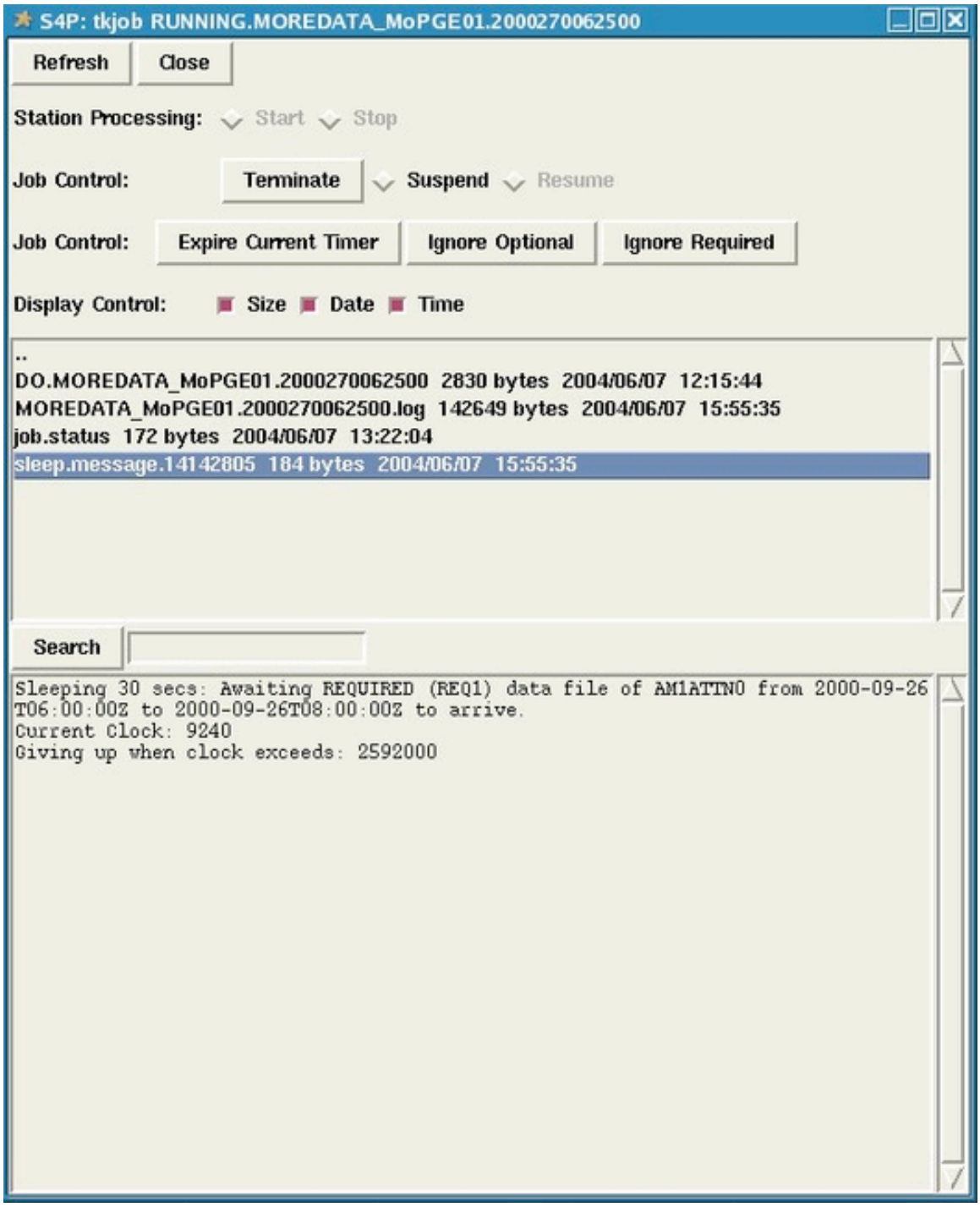


Figure 8-2. Sample Tkjob window resulting from clicking on one of the green boxes in Find Data. A file has been highlighted and its contents are shown.

8.2.1 Failure Handlers

Failure handlers are tasks to run when a job fails. The specifics of the task depend upon the station in which the failure occurs and the nature of the failure. The Job Monitor can

be configured to handle any arbitrary task(s) when a failure occurs and display that task as a button that shows up in the Tkjob display.

Adding a failure handler to a station is a simple modification of the station configuration file. Failure handlers are stored in the hash %cfg_failure_handlers. The example below illustrates how it is done in the Run Algorithm station:

```
%cfg_failure_handlers = (
        'Punt' => '../s4pm_failed_pge_handler.pl',
        'QC     Continue'     =>     '../failedQC_handler.pl     &&
remove_job.pl',
        'Remove Job' => 'remove_job.pl',
        'Restart'  =>  'clean_pge_output.pl  &&  restart_job.pl  &&
remove_job.pl'
);
```

Note: The failure handler buttons do not show up unless the job has failed. If the job is running (still green), failure handler buttons are not displayed.

The sections below are described some of the more important failure handlers currently in use by S4PM.

8.2.1.1 Remove Job

This simply removes the failed job directory. An alternate way to remove a job is to cd into the failed job directory and run remove_job.pl.

8.2.1.2 Restart Job

This failure handler is the most common and is generally available for any type of failure in any station. It simply resurrects the input work order and cycles it back into that station. This is useful in cases where the cause of the failure was easily remedied. Often, a restart job is bundled with a remove job at the same time. An alternate way to restart a job is to cd into the failed job directory and run restart_job.pl.

Note that to restart all failed jobs in a station, use the **Restart All Failed Jobs** interface (right click on any station button).

8.2.1.3 Punt Job

This failure handler is unique to Run Algorithm station. It bundles up the debris left behind by an algorithm failure (logs mostly) and exports it to the ECS archive as a failed algorithm tar file. It too is bundled with a remove job.

8.2.1.4 Resubmit PDR

This failure handler is used in the Receive PAN station. If a PDRD or a long PAN is received (both indicating a failure of ECS Ingest), this failure handler will resubmit the PDR in whole or in part (if there was a partial success) once the problem has been addressed.

8.2.2 Manual Overrides

Manual overrides are tasks that can be run while a job is still running. Like failure handlers, their specifics are varied. Buttons representing manual overrides are only displayed when jobs are still running.

Configuring a manual override is similar to configuring failure handlers. The hash involved is %cfg_manual_overrides. The example below is from the Find Data station:

```
%cfg_manual_overrides = (
        'Expire Current Timer' => '../tk_pwbox.pl -command "touch
EXPIRE_CURRENT_TIMER" -task "EXPIRE_CURRENT_TIMER" -label "Expire The
Current Timer"',
        'Ignore Optional' => '../tk_pwbox.pl -command "touch
IGNORE_OPTIONAL" -task "IGNORE_OPTIONAL" -label "Ignore Optional
Granules"',
        'Ignore Required' => '../tk_pwbox.pl -command "touch
IGNORE_REQUIRED" -task "IGNORE_REQUIRED" -label "Ignore Required
Granules"'
);
```

8.3 Compose Data Request Tool

The Compose Data Request tool is a window data ordering interface. See Figure 8-3.

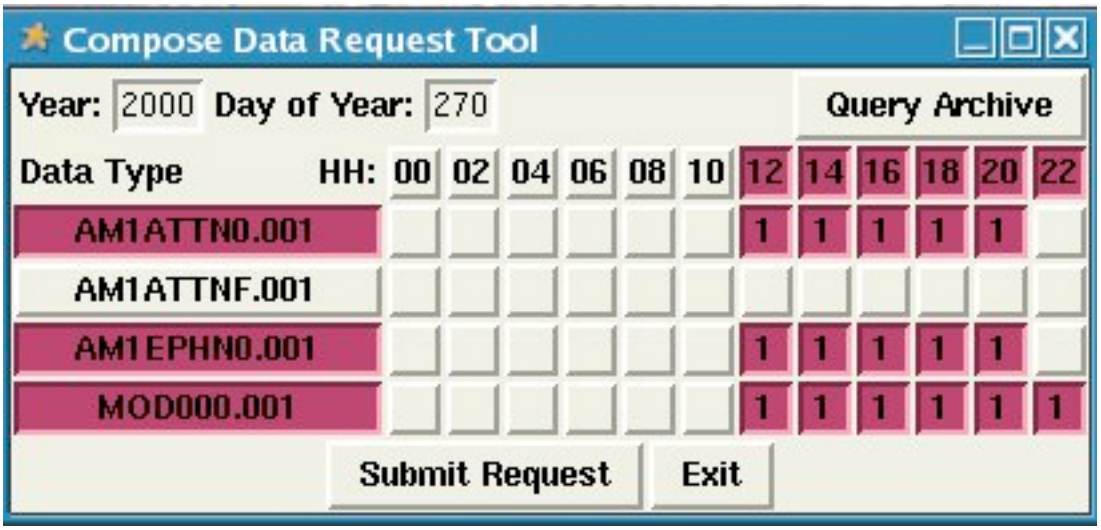


Figure 8-3. The Compose Data Request tool window. Data types are shown down the left side of the window. Those in red are selected; those in white have been deselected. Along the top, hours selected are in red and those deselected are in white. To acquire data, enter the year and day of year in the spaces indicated and then click on Query Archive. Once satisfied with selection, click on Submit Request.

To use, the operator specifies a year and day of year for the data to be ordered. The window is already configured to only show the data types relevant for that S4PM string. The operator can further refine the order by selecting or de-selecting time blocks (here configured as two hours per block) and data types. Once the **Query Archive** button is clicked, a query is made to the external archive system (either S4PA or ECS) for the data indicated. Hits show up as red boxes containing the number of files falling within each time block. In the above example, only one file is found for each time block.

If the number of files found is greater than one for any time block, a request for that data cannot be made since the tool cannot decide which of several files is the one desired. When this happens, all but the desired file must be deleted from the archive (or marked as deleted in ECS) and the query to the archive redone. Once satisfied with the results, the operator can click on **Submit Request** to create an input work order that either goes to the Acquire Data station or Request Data station. These stations, in turn, obtain the data.

8.4 View/Delete Data Tool

The View/Delete Data tool is a window that allows individual data files to be viewed or deleted cleanly from the S4PM system. The window is shown below in Figure 8.4. The numbers to the right of the file names are the current uses. When this number reaches zero, the file is automatically deleted.

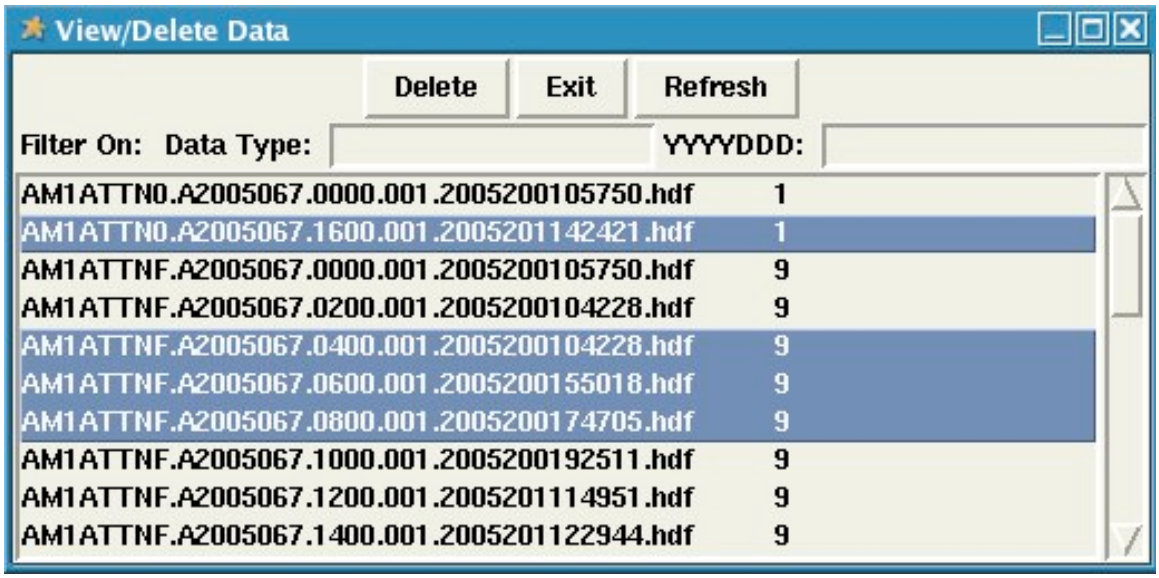


Figure 8-4. Sample Delete Data window showing files and current uses.

To delete files, an operator simply selects the files to delete and clicks on the Delete button. Rudimentary filtering is supported to aid the operator in finding the right files. This interface works by creating a SWEEP work order that is sent to the Sweep Data station.

8.5 S4PM Administration Tool

A number of clean up and other administrative tools have been placed in the S4PM Administration Tool and run as a separate window.

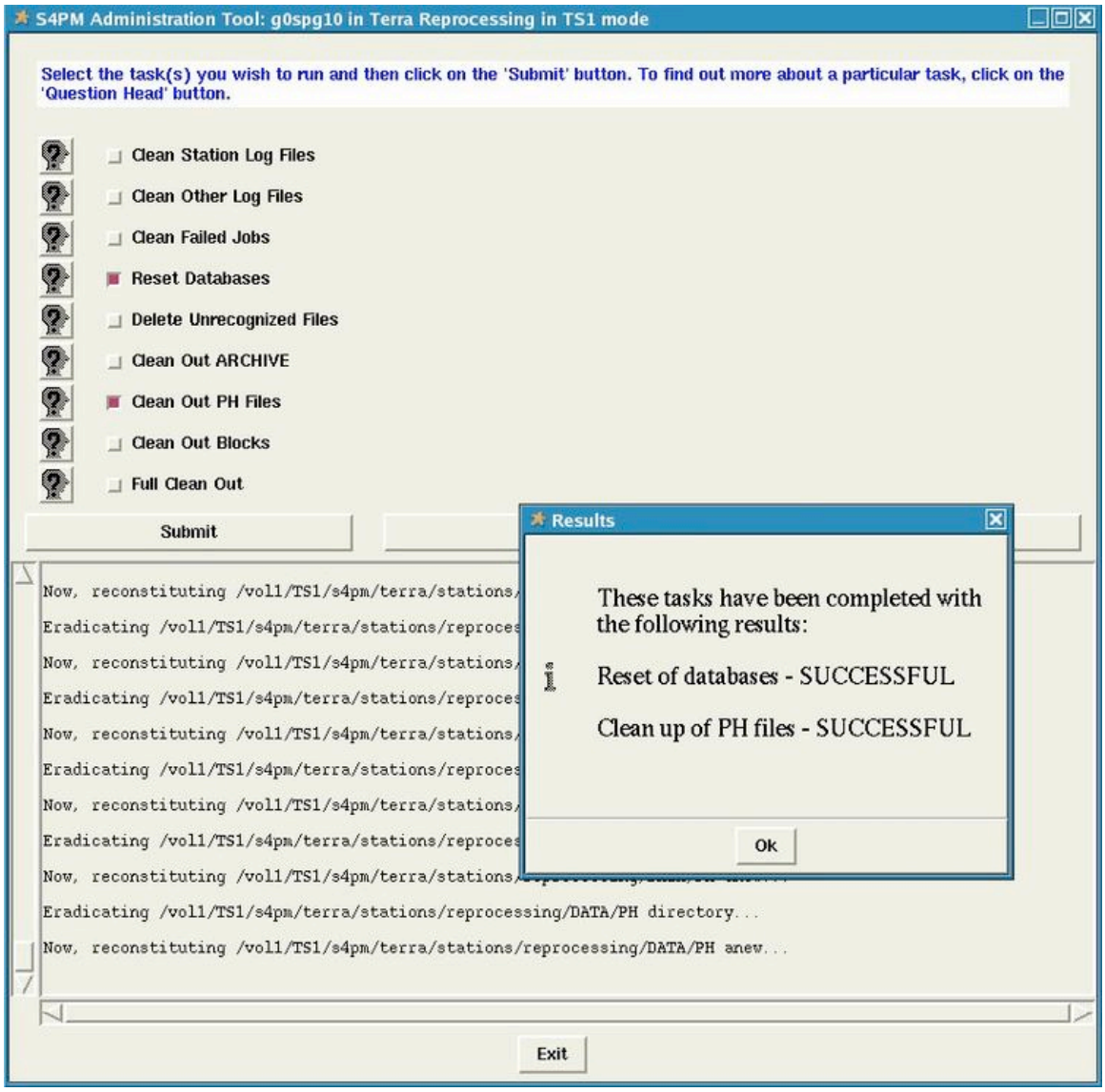


Figure 8-5. The S4PM Administration Tool. This sample shows the pop-up window that is displayed after a set of tasks have been completed (in this case, Reset Databases and Clean Out PH Files).

S4PM Administration Tool is currently configured to perform one or more tasks that an operator selects. The current list of selectable tasks includes:

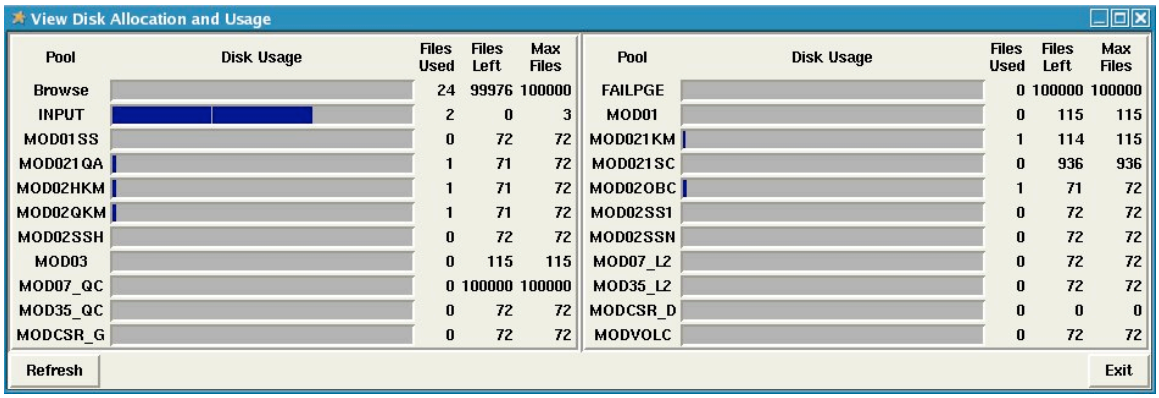
- Clean out station log files as well as other log files.
- Reset the Track Data and Allocate Disk databases to a clean, initialized state. It also removes any data referenced in those databases.
- Remove failed jobs in any station.
- Scan the S4PM file system for any unrecognized files and offer to delete them.
- Clean out the archived log files, log files which have been archived to a special area.
- Clean out Production History files. These files are not managed in the same way as data files. They therefore need to be periodically cleaned out manually.

- Remove data blocks.
- Do it all. This task bundles together all the tasks above plus some more. It effectively restores that S4PM string to its pristine state.

S4PM Administration Tool is designed to be extensible. Tasks can be added or deleted fairly easily as the need arises.

8.6 View Disk Allocate and Usage

The View Disk Allocation and Usage is a window which shows how much space is left in each disk allocation pool. An example is shown below.


View Disk Allocation and Usage

Pool	Disk Usage	Files Used	Files Left	Max Files	Pool	Disk Usage	Files Used	Files Left	Max Files
Browse		24	99976	100000	FAILPGE		0	100000	100000
INPUT		2	0	3	MOD01		0	115	115
MOD01SS		0	72	72	MOD021KM		1	114	115
MOD021QA		1	71	72	MOD021SC		0	936	936
MOD02HKM		1	71	72	MOD02OBC		1	71	72
MOD02QKM		1	71	72	MOD02SS1		0	72	72
MOD02SSH		0	72	72	MOD02SSN		0	72	72
MOD03		0	115	115	MOD07_L2		0	72	72
MOD07_QC		0	100000	100000	MOD35_L2		0	72	72
MOD35_QC		0	72	72	MODCSR_D		0	0	0
MODCSR_G		0	72	72	MODVOLC		0	72	72

Refresh Exit

Figure 8-6. Sample Disk Allocation and Usage window showing how much space is allocate to each disk pool and how much of that spaced is currently being used.

9. S4PM Logging

9.1 Purpose

The purpose of logging is to support:

- Quick and easy troubleshooting by operations personnel
- Metrics collection and reporting

The table below shows the types of logs, where and how they are saved, what process writes to them, and when/how they are cleaned up.

9.2 Logging Details

9.2.1 Chain Logs

Chain logs are log files that move from station to station along with work orders. As processing is done at each station along the way, messages from that station are concatenated to the chain log file. Thus, chain logs contain a history of a single thread of jobs moving through S4PM.

Chain logs adhere to a particular format. The break from one station to the next is indicated by a line containing all equal signs (=) followed on the next line by a date and time stamp, the process ID of the Stationmaster process for that station, and then the station name. For example:

```
=====================================================================
===
2005-07-26 11:40:22 41432948 Receive DN
```

Figure 9-1. Illustration of how each station section is introduced in the chain log file.

The next lines display the contents of the work order going into the listed station. The work order contents are terminated by a line of dashes.

Following the input work order contents are lines written by the station scripts. Each such line is preceded by the same date and time stamp and process ID as listed at the top of the section (Figure 9-1). Each such entry is preceded by a mnemonic indicating the severity of the message. These are described in Table 9-1:

Severity Mnemonic	Meaning
DEBUG	Only appears when debugging is enabled and is used solely for debugging purposes (debugging can be enabled by setting the environment variable OUTPUT_DEBUG to a non-zero value).
INFO	The message is for informational purposes only.
WARNING	A minor non-fatal warning condition was detected. These aren't generally serious except when they're followed soon by a failure.
ERROR	A major non-fatal error condition was detected. These are significant and often presage a fatal condition.
FATAL	A fatal error condition was detected and this results in a job failure.

Table 9-1. Standard S4PM error severity mnemonics and their meanings.

For example, Figure 9-2 shows a single line for the Find Data station. The INFO means that the message is informational only.

```
2005-07-26 11:43:12 41432956 INFO s4pm_find_data.pl: Looking for CURR
data file of AM1EPHN0 with a need of REQ1.
```

Figure 9-2. Sample line from the Find Data section of a chain log file.

Following the messages output by the station scripts is another line of equal signs indicating the beginning of a new station section. This indicates that the chain log has made a move from the preceding station to the next station along with a work order. Note that in moving from one station to the next, the chain log is also renamed to match the name of the work order it is paired with.

The level of logging in the chain logs can be controlled via the environment variable S4P_LOG_LEVEL (note, not S4PM_LOG_LEVEL). This sets the minimum severity of the messages written to the chain logs. The valids are FATAL, ERROR, WARNING, INFO, and DEBUG. Note that DEBUG messages are only written to the chain logs if the environment variable OUTPUT_DEBUG is set to a non-zero value (assuming that S4P_LOG_LEVEL allows it).

9.2.2 Find Data Log

The Find Data station maintains a separate log file named find_data.log. Each algorithm can be configured for required or optional input data. Production rules also allow for alternate inputs. The information in this log file shows how each job was resolved in terms of the data actually found. For every single job passing through the Find Data station, a single line is added to the Find Data log.

Each line in the Find Data log contains a number of parameter=value pairs. Table 9-2 describes each such parameter:

Parameter	Description
PGE	The name of the algorithm as configured in S4PM.
DATADATE	The processing start date and time for the job.
PRODDATE	The production date and time, that is, the current time on the machine in which the job is running.
datatype=currency …	One or more input data types with the currency as ultimately resolved by Find Data. Currency is a term that indicates how the time coverage of the data related to the processing period (whose start is DATADATE). CURR indicates that the data's time coverage is the same as the processing period; FOLL*n* indicates that the data's time coverage is *later* than the processing period by n steps; PREV*n* indicates that the data's time coverage is *earlier* than the processing period by n steps.

Table 9-2. Parameters in the Find Data log file and their meanings.

An excerpt from a Find Data log is show in Figure 9-3. In this figure, four entries are shown, each corresponding to a job in the Find Data station. The first is for the algorithm MoPGE03. For this algorithm, the Find Data station was able to locate the current MOD021KM, MOD03, GDAS_0ZF, SEA_ICE, and REYNSST input data files. For the NISE data, it was able to find the following one. For the MOD02QKM, no data file was located; this data type must have been configured as optional, otherwise the job in Find Data would have failed.

```
PGE=MoPGE03 DATADATE=2000-11-01T00:45:00Z PRODDATE=2005-06-13T16:15:35
MOD021KM=CURR MOD03=CURR MOD02QKM=NONE GDAS_0ZF=CURR SEA_ICE=CURR
REYNSST=CURR NISE=FOLL1
PGE=GdPGE02B DATADATE=2000-11-01T01:10:00Z PRODDATE=2005-06-13T16:20:26
MOD021KM=CURR MOD02QKM=NONE MOD02HKM=NONE
PGE=GdMOD02SS DATADATE=2000-11-01T01:25:00Z PRODDATE=2005-06-
13T16:27:52 MOD03=CURR MOD021KM=CURR
PGE=GdPGE02B DATADATE=2000-11-01T01:25:00Z PRODDATE=2005-06-13T16:27:52
MOD021KM=CURR MOD02QKM=CURR MOD02HKM=CURR
```

Figure 9-3. Excerpt from a Find Data log file showing 4 entries, each corresponding to a job.

9.2.3 Track Data Transaction Log

This log in the Track Data station directory logs all database transactions made. Each transaction includes a date and time stamp, the transaction type, the full path of the data file, and the current number of uses outstanding on that data file. The transaction type indicates the type of transaction. Possible values of transaction type and what they mean are shown in Table 9-3.

Transaction Type	Description
INSERT	A data file has been added to the Track Data database.
DELETE	A data file is marked for deletion in the Track Data database. A work order will be sent to the Sweep Data station to carry out the physical deletion of the file.
UPDATE	The uses for some data file has been updated.
UPDATE_GHOST	Indicates an update of a file that is no longer in the Track Data database.
EXPECT	Similar to a INSERT except that the data file hasn't yet arrived; it is merely expected to arrive. This type of transaction is only used in on-demand processing. This allows Track Data to ignore multiple requests for the same data file while at the same time allowing it to update its uses appropriately.

Table 9-3. Transaction types in the Track Data transaction log and what they mean.

9.2.4 Case-Based Reasoning Log

This log file is produced based on the theory of case-based reasoning for supporting autonomic computing. Although data are logged in a case-based reasoning log file, the data are not used by S4PM (not yet, at least).

The log file is named cbr.log and resides in the station root directory. What gets recorded in this log file is any manual operation by a human operator and includes things that operators do when intervening. This includes the failing of jobs, the restarting of failed jobs, the invocation of manual overrides (*e.g.* **Ignore Optional** in Find Data), etc. The eventual goal is to be able to use this information when case-based reasoning is added to S4PM.

Each entry contains a time stamp (formatted to be machine readable versus human readable), a letter indicating the type (F for failure, R for failure recover, or M for manual override), the exit code of the failure or the handler, the station name in which the intervention occurred, the work order type of the job on whose behalf the intervention was made, and a message string.