

4.11 EMD General Process Failure Recovery Concepts

During EMD processing, client or server failures can occur. These failures cause certain recovery events to take place within the EMD. To understand the General Process Failure Recovery of the EMD processes, several key concepts must be described. These failure recovery concepts are:

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

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

4.11.1 Client-Server Rebinding

EMD uses a socket-based infrastructure to provide an rpc-like capability for a distributed object environment. In the infrastructure, the client applications call proxy objects that represent a server's server objects. A proxy object uses the socket name service to find its server object by its known name. The socket name service returns an internal reference to the server object, known as its "binding handle". The process itself is called "binding".

There are two possible failure situations to be discussed for rebinding. These failure situations are:

- System Startup Failures
- Server Crashes

It is conceivable that the initial attempt to "bind" with a server fails, for example, because the server is not up or because the socket name server is not running. The EMD socket-based infrastructure provides parameters for a number of automatic retries of a binding attempt and a retry interval.

The internal reference to a server object can become invalid, for example, if the server is shutdown and re-started. When this happens, the client needs to obtain a new reference. This process is called "rebinding." EMD client libraries contain code that makes an automatic attempt at rebinding with the server to support failure recovery.

Without such an automatic rebinding attempt, all client applications of a server would have to be brought down and re-started if a server fails (the re-started application can, of course, "bind" again). And if these applications have client applications, the client applications would need to be re-started and so on down the process chain. The result would be that the failure of a single server could ripple through most of the EMD and require the shutdown and re-start of a large portion of the system.

With automatic rebinding this is not usually the case. Only rarely is it necessary to bring down a server to allow it to get a new "binding handle". When a server goes down (i.e., crashes), the other application(s), which communicate with it lose their "binding handle." However, the application(s) do continually try to rebind to the "downed" server. If the "downed" server comes back up before the number of retries are exhausted, the application(s) do eventually get a new valid "binding handle" for the re-started server and communications can continue. Operations may notice a brief pause in the execution of some applications, but as soon as the failed server is back on-line, the system reverts to a normal state.

Client-server rebinding is generally done in the client library. Any configurable parameters are contained in the client libraries included in the client applications. The configurable parameters have defaults.

4.11.2 Sybase Reconnecting

A similar approach has been implemented by the EMD infrastructure that provides the interface with the Sybase ASEs. For example, an EMD application may attempt to connect to its Sybase ASE while the server is still in the process of starting up. The connection attempt fails, but the infrastructure code attempts the connection for a configurable number of times, waiting for a configurable amount of time between each connection attempt.

Most EMD applications obtain a connection for the duration of a transaction and relinquish it when they are done with it. These applications have been directed to implement the following recovery behavior: if they get a Sybase error that requires the transaction be re-done (for example, a deadlock error), they release and then re-request the connection. If the cause of the error was a Sybase ASE fault, this connection attempt fails, but causes the infrastructure code to enter the connection re-try loop. If the Sybase ASE is restarted before the retries are exhausted, the application continues normally and now completes the transaction in progress when the Sybase fault occurred.

However, operations should be aware of the following facts:

- Not all EMD applications are able to use the EMD Sybase interface code. For example, the Science Data Server does not, for performance reasons. In addition, the MSS order tracking server uses its own DB connection API for performance reasons.

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

4.11.3 Request Identification

EMD generates a unique identifier for each type of request that requires fault-handling provisions. These "recoverable requests" fall into one of these two categories:

- 1) User Requests: their request identifiers are generated by the System Management Subsystem (MSS) when request-tracking information is created.
- 2) System Requests: their identifiers are system generated and referred to as RPC ID. They are based on the Universal Unique Identifier (UUID), a mechanism for creating system-wide unique identifications.

Some examples of EMD processes that use the User Requests are the V0 Gateway, E-mail Parser Gateway, and ASTER Gateway.

The following describes how request identification is used during recovery. As a request propagates through the system, each associated client/server exchange is assigned a unique RPC ID. However, the RPC ID for each interaction is derived from the previous RPC ID received by the client for this request. Thus, all RPC IDs associated with a given request have a common portion, which relates the various client/server calls to one another. More importantly, given the previous RPC ID, clients consistently reproduce the same RPC ID that was submitted to the server on the subsequent event. The concept of reproducible RPC IDs is central to the EMD fault recovery capability. When requests are retried from client to server, they are always submitted with the same RPC ID as was used in the original submission of the request, even if either the client or server has crashed between retries.

RPC IDs are also central to the check-pointing aspect of fault recovery. As requests arrive at fault recovery-enabled servers, they are recorded in a persistent store (typically, a database), tagged with the RPC ID, which identifies the request. As the request is serviced, check-pointing state information may be updated in the persistent store, up to and including the completion status of the request. This allows the servers to resume servicing from the last check-pointed state, particularly upon re-submission from a client.

Many kinds of requests do not pose recovery issues and thus, do not employ request identifiers. For example, if a search is submitted to the Science Data Server, and no response is received, the client application can simply re-submit the search. However, some types of requests do pose recovery issues.

4.11.4 Senior Clients

A Senior Client is an EMD client process that originates an EMD request, has fault recovery requirements and may lead to a chain of sub-requests. The Senior Client assigns the original request identifier (rpcid). It is responsible for re-submitting the request if it gets a retry error or no response. It is responsible for reassigning the same rpcid upon re-submission of a request.

Senior Clients include the Ingest Granule Server, the ASTER Gateway, the V0 Gateway, the E-mail Parser Gateway, the Subscription Server, and the Science Data Server.

Senior Clients that send requests and receive acknowledgments of receipt of their requests from the receiving servers can expect to receive an outcome (a response, a code, data, or messages). If no acknowledgments are received from the receiving server, the Senior Client must re-submit the request with the same RPCID as the initial request after a failure recovery. The unique RPCID helps receiving servers to recognize duplicate requests so these duplicate requests can be acknowledged or ignored.

There is one exception to this re-submission rule. Senior Clients are not responsible for recovery of the process environment and completion of the requests, if they are cold started. If the restart is a cold start, there are no automatic restarts for any previous requests and all requests are submitted as new requests.

In essence, a Senior Client takes on the role of the "end user" for system requests. If anything happens to a request upstream, it has the ultimate responsibility for deciding what to do with the request (retry or suspend/abort it and tell the operator, i.e., "the buck stops" with the Senior Client).

4.11.5 Request Responsibility

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

Synchronous Requests

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

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

Asynchronous Requests

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

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

4.11.6 Queues

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

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

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

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

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

4.11.7 Request Responses

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

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

Where this is not possible (e.g., system generated requests, or if a data order runs into an error after it was already accepted and the user/operator is no longer connected), errors are logged. They are brought to the attention of the DAAC operations staff for action if there is a corresponding server GUI for the operator. However, not all EMD servers are associated with an operator GUI. In these cases, operators need to monitor the server logs for errors on a regular basis.

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

- Fatal errors,
- Retry errors and
- Warnings

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

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

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

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

- a. Did it reach the server?
- b. Did the server start the request but not complete it?
- c. Did the server complete the request with an error but was not able to send the error response?
- d. Did the server process it successfully?

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

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

Table 4.11-1 summarizes the five categories of request responses, and the specific requirements for the application or server currently responsible for the request. EMD servers have been directed to classify their responses accordingly.

Table 4.11-1. Request Responses

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

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

EMD processes that encounter an error or receive an error from a server request may either pass the error back to a higher-level client or present it to the operator for operator intervention. The fault handling policies are detailed in Table 4.11-2.

Table 4.11-2. Fault Handling Policies (1 of 2)

CI	Client Processes	Fault Handling Policy
DPLINGEST	EcDIInPollingService	<p>Retry errors: Errors are retried a configurable number of times for resources, then the the resource is suspended. Examples of retrieable errors are connection failures and timeouts for file transfers.</p> <p>Fatal errors: Resources are suspended immediately for non-transient errors. Examples of non-transient errors are host address does not exist, and login to host failed.</p>
	EcDIInProcessingService	<p>Retry errors: Errors are retried a configurable number of times for resources, then the the resource is suspended. Examples of retrieable errors are connection failures and timeouts for quick server operations such as ODL to XML conversion and Data Pool Insertion.</p> <p>Fatal errors: Resources are suspended immediately for non-transient errors. Examples of non-transient errors are quick server not running and failures to login to a transfer host.</p>
	EcDIInNotificationService	<p>Retry errors: Errors are retried a configurable number of times for resources, then the the resource is suspended. Examples of retrieable errors are connection failures and timeouts for file transfers.</p> <p>Fatal errors: Resources are suspended immediately for non-transient errors. Examples of non-transient errors are host address does not exist, and login to host failed.</p>
SDSRV	EcDsScienceDataServer	<p>Retry errors: Errors are retried a configurable number of times, then passed back to the calling client process unchanged. The default retry policy for SDSRV servers is “retry forever”. For async Acquire requests that involve subsetting, retry errors encountered with the HDF servers are not returned to the client. Instead the request is queued for future execution.</p> <p>Fatal errors: Errors are logged and the request is passed on to the Data Distribution with the appropriate error information. Data Distribution uses this error information in the distribution notification sent to users to inform them of the errors.</p> <p>After a SDSRV fault, the operator restarts the HDF servers manually.</p>
OEA	EcOwOgcEchoAdaptor	<p>Fatal errors: Errors are logged and request is marked as “FAILED”, error responses are sent to users to inform them of the errors.</p>

Table 4.11-2. Fault Handling Policies (2 of 2)

CI	Client Processes	Fault Handling Policy
SSS	EcNbSubscribedEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	All errors are logged. Failed attempts to connect to Sybase are retried. Failed database queries are retried if the reason for failure was deadlock.
DMS	EcDmV0ToEcsGateway	All errors are logged. Fatal errors not retried, reported back to customer.
OMS	EcOmOrderManagerServer	All errors are logged. Failed attempts to connect to Sybase are retried. Retry errors: Errors are retried a configurable number of times, then passed back to the calling process. Fatal errors: Errors are logged and the request is suspended and operator intervention is generated. Operator then have a choice to hold, fail or resubmit the request.
BMGT	EcBmBMGT	Fatal Errors: All Errors are logged but not retried. If BMGT is run as a cron job and it fails to complete, BMGT can be run manually for that particular day by using EcBmBMGTStart script. The operator needs to update EcBmBMGTUserParams.xml file in the config directory by setting <begindate> day prior to the date of failure</begindate> <enddate>day of the failure</enddate>
BMGT	EcBmBulkURL	Fatal Errors: All Errors are logged but not retried. If BulkURL is run as a cron job and it fails to complete, BulkURL can be run manually for that particular day by using EcBmBulkURLStart <MODE> Insert. The operator needs to update EcBmBulkURLConfigParams.xml file in the config directory by setting <begindate> day prior to the date of failure</begindate> <enddate>day of the failure</enddate> <doPreviousFlag>>false</doPreviousFlag>
DPL	TBD	TBD

4.11.8 Duplicate Request Detection

The above scheme for handling requests in cases of faults poses a potential problem. The request could have been re-submitted because there was no response returned by the server. But, in fact, the server completed the request but was unable to get the status back to the client (e.g., because of communications problems or a machine crash). The following measures are intended to deal with this situation:

- **Trivial duplicate requests.** There are many interfaces where sending a new request to retry a service whose outcome is unknown either has no or negligible impact on the

EMD. This is because many EMD services have been designed with this goal in mind. For example, after a failure, the Science Data Server CSCI can send a duplicate request for inserting a new collection to the Data Dictionary CSCI or the Advertising Service CSCI. The Data Dictionary CSCI or the Advertising Service CSCI simply interprets the second request as an update for the (now) existing collection. When the SDSRV exports the same event more than once to the Subscription Service Computer Software Component (CSC), it assumes it is meant as a replacement for the previous one. This made designing the recovery for an ESDT update fairly simple. If the update fails, it can always be re-started at the beginning. Any duplicate requests issued to dictionary, or subscription services are of no consequence.

- **Recognize non-trivial duplicate requests.** Where executing the same request more than once can have undesirable consequences, EMD provides a mechanism for recognizing re-submitted requests. Each request is tagged with a unique identifier (see Request Identification Section). Upon submission of a request, the receiving server of the request must check the identifier and recognize when it is a re-submission of a previous request it received. For example, the server may realize the request has been completed and simply acknowledges the successful completion. Yet another example is the OMS CSCI recognizing a duplicate request originating from the V0ToECSSGateway if the gateway is configured not to allow duplicates.

4.11.9 Server Crash and Restart

- **Server Crash**

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

- **Server Restart**

When a server restarts, it may perform various re-synchronization activities in order to recover from an unexpected termination. In the event of a server cold start or cold restart, the server also cancels all outstanding requests and reclaims all associated resources. Note that the distinction between cold start and cold restart is described in the section above on Start Temperature. Specifics of server startup behavior are detailed in Table 4.11-3. Unless otherwise stated, existing request queues are always retained for warm restarts and cleared for cold starts or cold restarts.

Table 4.11-3. Server Response versus Restart Temperature

CI	Server(s)	Special Behavior on Warm Restart	Special Behavior on Cold Start or Cold Restart
DPLINGEST	EcDIInPollingService	None.	None.
DPLINGEST	EcDIInProcessingService	All ingest requests that did not reach a terminal state in the previous processing run will be re-queued in processing and executed from their last persisted state.	All ingest requests that did not reach a terminal state in the previous processing run will be moved to the state 'TERMINATED'. They will not be re-queued.
DPLINGEST	EcDIInNotificationService	None.	None.
OEA	EcOwOgcEchoAdaptor	N/A	N/A
SSS	EcNbSubscribedEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	N/A	N/A
DMS	EcDmV0ToEcsGateway	None. No persistence.	None.
OMS	EcOmOrderManagerServer	N/A	N/A

- **Request Re-submission**

Upon restarting a crashed client or server, requests are typically re-submitted. If the restarted process was started warm, the fault recovery capabilities permit the server to resume processing of the request from its last check-pointed state. This prevents needless repetition of potentially time-consuming activities. Specific behavior of servers upon re-submission of a request is detailed in Table 4.11-4. Note that a cell value of N/A means the server either has no clients or the clients do not re-submit requests.

Table 4.11-4. Server Response for Request Re-submission

CI	Server(s)	Behavior on Request Re-submission
DPLINGEST	EcDllnPollingService EcDllnNotificationService	N/A
DPLINGEST	EcDllnProcessingService.	The newly resubmitted request will have the same requestid and continue being processed from the last check-pointed state from the last processing run.
OEA	EcOwOgcEchoAdaptor	The newly resubmitted request will be using a different referenceld and resultSetNamen (otherwise error will be generated), therefore, OEA server will treat it as a new request.
SSS	EcNbEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	There is no resubmission of requests. EcNbRecoverDriver monitors the SSS database for events or actions that did not run to completion and re-enqueues them.
DMS	EcDmV0ToEcsGateway	No resubmission of requests.
OMS	EcOmOrderManagerServer	Incomplete requests in OMS are picked up and processed upon restarting OMS Server. The incomplete requests have the same requestid. If the request has already been staged the first time around, the granules should be inData Pool already and does not need to be staged again.

4.11.10 Client Crash and Restart

- **Client Crash**

When a client crashes in the EMD system, fault recovery-enabled servers have several possible responses. Servers may continue to service client requests, independent of the client's status. Servers may choose to suspend processing of client requests, but permit the requests to be resumed upon client recovery. Or, servers may terminate servicing of the client requests, canceling all work done on the requests. The behavior of each CI is detailed in Table 4.11-5. Note that the behavior of a server in the event of a client crash does not vary from client to client.

Table 4.11-5. Server Responses to Client Failures

CI	Server(s)	Behavior on Client Crash
DPLINGEST	EcDIIInProcessingService	Requests in process are serviced to completion.
	EcDIIInNotificationService	
	EcDIIInPollingService	N/A
SDSRV	EcDsScienceDataServer	Requests in process are serviced to completion.
OEA	EcOwOgcEchoAdaptor	Requests in process are serviced to completion.
SSS	EcNbSubscribedEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	Processing is database driven and not influenced by outside processes.
DMS	EcDmV0ToEcsGateway	Requests in process are serviced to completion.
OMS	EcOmOrderManagerServer	Requests in process are serviced to completion.

- **Client Restart**

When a client restarts in the EMD system, it sends a restart notification to each server with which it interacts. Clients notify servers they have come up “cold” or “warm”, and do not differentiate between cold start and cold restart. Generally, the notification temperature sent to the server matches the temperature at which the client process is restarted.

Table 4.11-6 shows exceptions to the general behavior for client submission of restart notification:

Table 4.11-6. Client Restart Notification Exceptions (1 of 2)

Client Processes	Server Processes	Restart Notification
EcDIIInPollingService EcDIIInProcessingService EcDIIInNotificationService	N/A	N/A
EcDsScienceDataServer	EcDsStArchiveServer	Always sent warm (Also see Note 1 below)
	EcDsStStagingDiskServer	Always sent cold (Also see Note 1 below)

Table 4.11-6. Client Restart Notification Exceptions (2 of 2)

Client Processes	Server Processes	Restart Notification
N/A	EcNbSubscribedEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	N/A
EcDmV0ToEcsGateway	N/A	N/A
EcOmOrderManagerServer	N/A	N/A

The default server behavior in response to a startup notification from a client is as follows:

- Warm Notification: Outstanding requests for the restarted client are left available in the persistent store. These requests may be re-submitted by the client, and serviced to completion upon re-submission. Associated resources are left allocated until the requests are completed.
- Cold Notification: All outstanding requests for the restarted client are cancelled. If the client re-submits any cancelled request using the same RPC ID (e.g., by pressing the Retry button from an operator GUI), it failed with a fatal error due to the client cold startup notification. Any resources associated with the cancelled requests are released and reclaimed by the system.

Specific aspects of server behavior upon receipt of a client restart notification are detailed in Table 4.11-7:

Table 4.11-7. Server Responses to Client Notification

CI	Server(s)	Behavior on Cold Notification	Behavior on Warm Notification
DPLINGEST	EcDIInPollingService EcDIInProcessingService EcDIInNotificationService	N/A	N/A
SDSRV	EcDsScienceDataServer	N/A	N/A
OEA	EcOwOgcEchoAdaptor	N/A	N/A
SSS	EcNbSubscribedEventDriver EcNbActionDriver EcNbDeleteRequestDriver EcNbRecoverDriver	N/A	N/A
DMS	EcDmV0ToEcsGateway	N/A	N/A
OMS	EcOmOrderManagerServer	N/A	N/A

Some known limitations within the EMD are:

- a.) Requests with many sub-requests can experience timing problems because of nested retries or because one of the requests is suspended.
- b.) Coding errors can cause unanticipated fault behavior that is different from what is described above (and such occurrences should be reported as NCRs).
- c.) System engineers and designers may have made mistakes in classifying errors (e.g., as fatal versus retry).
- d.) Not all EMD applications use the error recovery capabilities of the EMD Sybase interface infrastructure.

4.12 Product Distribution System (PDS) Subsystem Overview (REMOVED)

4.13 Spatial Subscription Server (SSS) Subsystem Overview

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

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

Spatial Subscription Server (SSS) Context

Figure 4.13-1 is the Spatial Subscription Server context diagram. Table 4.13-1 provides descriptions of the interface events in the Spatial Subscription Server context diagram.

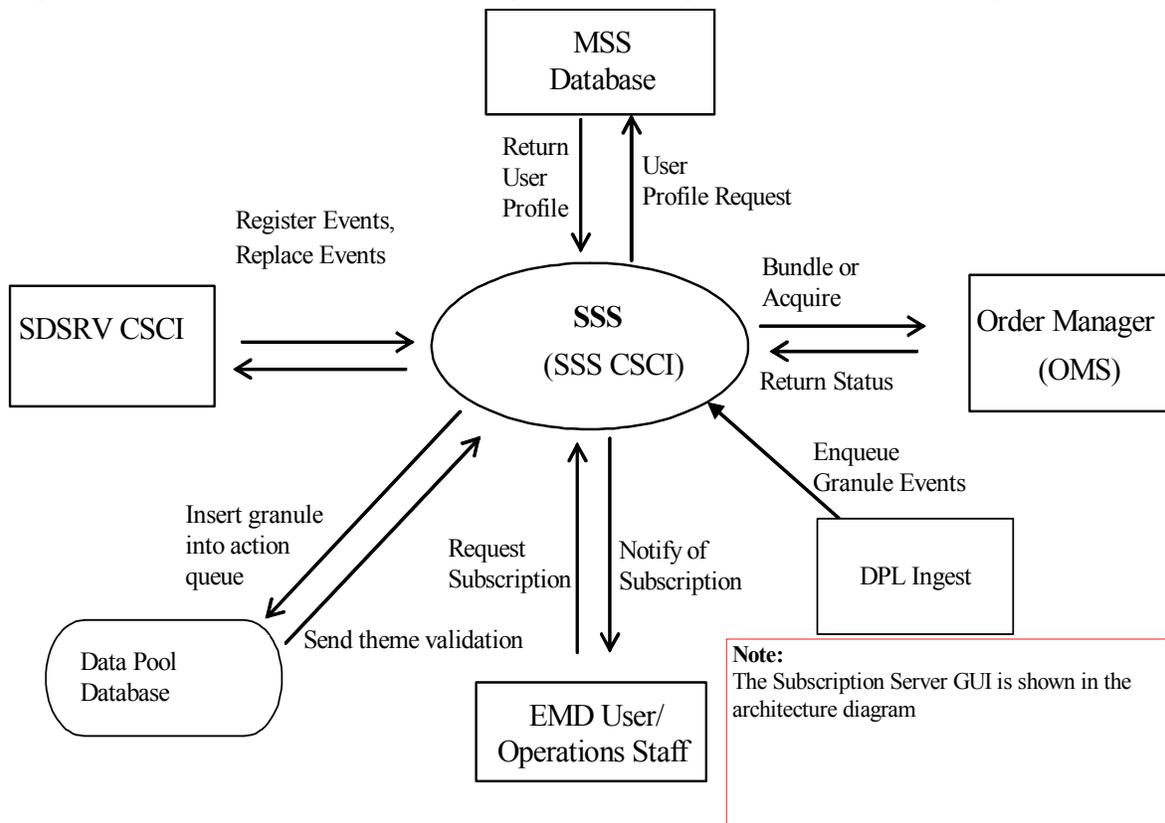


Figure 4.13-1. Spatial Subscription Server Context Diagram

Table 4.13-1. Subscription Server Interface Events

Event	Interface Event Description
User Profile Request	User Profile Request - MSS provides requesting CSCIs with User Profile parameters such as e-mail address and shipping address to support their processing activities. SSS will not enter a subscription for or distribute a granule to a user who does not have a valid MSS user profile.
Notify of Subscription	The SSS CSC sends email notification to the EMD User when the subscribed event occurs, provided that a notification action was requested in the subscription.
Request Subscription	A subscriber (EMD user requests Operations Staff to create the subscription) sends information (ESDT and, optionally, acceptable metadata values) with the subscription, specifying one or more actions (e.g., acquire and/or notification) to be taken when the subscribed event occurs.
Return status	Status returned by a stored procedure to indicate whether or not the call succeeded.
Register Events	The SDSRV CSCI sends information about an Earth Science Data Type (ESDT) to the SSS database when an ESDT is installed into the system.
Enqueue Granule Events	DPL Ingest will enqueue new granule events via an SSS stored procedure call.
Replace Events	The SDSRV CSCI notifies the SSS database when an ESDT is deleted from the system.
Return User Profile	MSS returns the user profile to SSS as part of user authentication.
Bundle or Acquire	SSS notifies OMS, via a stored procedure call, when a granule has matched a subscription. If the subscription is bundled, i.e. associated with an OMS bundling order, then the granule is inserted into the appropriate OMS bundle. If the subscription is not bundled, then an acquire request is sent to OMS. Whether an acquire is sent to SDS or OMS is determined by an SSS configuration parameter.
Insert granule into action queue	If a subscription has an associated Data Pool action, then SSS will insert a row into the Data Pool database action queue table, indicating that the granule that matched the subscription should be inserted into the Data Pool.
Send theme validation	If a subscription's Data Pool action is associated with a Data Pool theme, then the Data Pool will verify, via stored procedure call that the theme exists and is enabled for insert.

4.13.1 Spatial Subscription Server Architecture

Figure 4.13-2 is the Spatial Subscription Server architecture diagram. The diagram shows the events sent to the Spatial Subscription Server processes and the events the Subscription Server processes send to other processes.

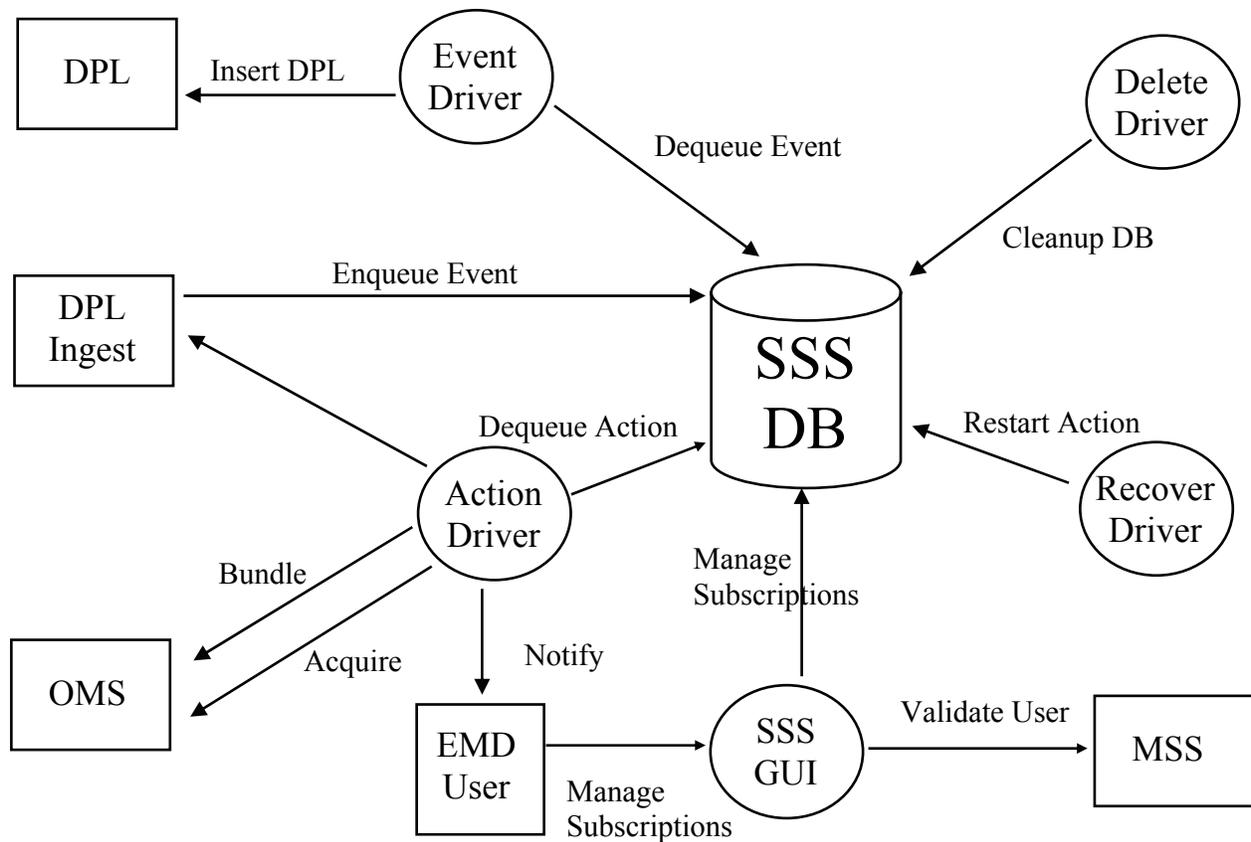


Figure 4.13-2. Spatial Subscription Server Architecture Diagram

Table 4.13-2 provides descriptions of the processes shown in the Spatial Subscription Server architecture diagram.

Table 4.13-2. Spatial Subscription Server Processes

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

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

4.13.1.1 Subscription Server Process Interface Descriptions

Table 4.13-3 provides descriptions of the interface events shown in the Subscription Server architecture diagram.

Table 4.13-3. Spatial Subscription Server Process Interface Events (1 of 2)

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

Table 4.13-3. Spatial Subscription Server Process Interface Events (2 of 2)

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

4.13.1.2 Subscription Server Data Stores

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

- **Attributes:** includes the ESDTs for which subscriptions can be created and the metadata attributes that can be used to qualify those subscriptions.
- **Subscriptions:** information about subscriptions that have been created for users, their associated qualifying expressions, and their associated actions.
- **Events:** information about newly arrived data granules, their metadata, and the subscriptions that match them.
- **Actions:** information about actions for matched subscriptions that need to be carried out, e.g. acquire or email notification.

Table 4.13-4 provides descriptions of the data found in the principal Sybase ASE data stores used by the Spatial Subscription Server. More detail on these and other data stores can be found in the Spatial Subscription Server Database Design and Schema Specifications for the EMD Project (311).

Table 4.13-4. Spatial Subscription Server Data Stores (1 of 2)

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

Table 4.13-4. Spatial Subscription Server Data Stores (2 of 2)

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

4.14 Data Pool Subsystem Overview

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

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

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

1. **Data Pool Insert:** inserts ECS data into the Data Pool. ECS data is copied from the ECS archive into the Data Pool, based on an ECS granule id. The Data Pool database inventory is updated for each granule inserted in the Data Pool. Data Pool Insert consists of six major subcomponents:
 - a) the Data Pool Action Driver (DPAD): a C++ executable which schedules Data Pool insert actions based on a queue of Data Pool insert actions populated by the Spatial Subscription Server, the Batch Insert Utility, Data Pool Ingest, or the Order Manager Server;
 - b) the New (7.20) Data Pool Insert Utility (NDPIU), a java executable which manages the registration and publication of an ECS data granule into the Data Pool;
 - c) the Data Pool Quick Server, a C++ executable which is installed on the ECS service hosts. The Quick Server is used by the DPAD to perform copy and checksum operations. It is also used by DPL Ingest and OMS to perform operations which are performed on ECS service hosts for load balancing reasons, or which cannot be performed on the local host due to lack of data access (mount points, etc.)
 - d) the Data Pool Metadata to XML generation tool (M2XT), a java executable which translates ECS granule metadata from the Science Data Server database into XML, for storage in the Data Pool directories;
 - e) the band extraction utility (bandtool), a C executable invoked by the DPAD, which extracts band information from HDF-EOS granules and stores the extracted information in a .BandHeader file in the temp area on the ESDT file system. The .BandHeader file is used by the NDPIU during granule registration. The bandtool is invoked only if the granule being inserted is from a collection eligible for conversion by the HDF-EOS to GeoTiff Conversion Tool (HEG);
 - f) the jpeg extraction utility (hdf2jpeg), a C executable invoked by the NDPIU, which extracts browse images (jpeg or raster) from a browse hdfEOS file on browse publication.
2. **Data Pool Cleanup and Validation (EcDICleanupDataPool.pl):** a perl utility, which cleans expired granules from the public Data Pool directories and database. This utility normally runs as a cron job. The utility may also be used to report on and correct inconsistencies between the Data Pool directories and the database (validation).

3. **Data Pool Web Access (EcDIWebAccess):** a java-based web application, which runs with the apache web server and related COTS. The Data Pool Web Access application allows end-users to perform drill-down searches for Data Pool data, to view metadata and browse images online, and to convert and/or order Data Pool data.
4. **Data Pool Maintenance GUI (EcDIDpm):** a perl-based web GUI that allows DAAC operations staff to monitor Data Pool insert activity and to control the Data Pool configuration.
5. **Data Pool Access Statistics utilities:** perl utilities which parse firewall ftp logs (EcDIRollupFwFtpLogs.pl) and Data Pool Web Access custom code logs (EcDIRollupWebLogs.pl) for accesses to the Data Pool directories, and then roll up access information for storage in the Data Pool database.
6. **Data Pool FTP Server:** customized wu-ftp daemon, which supports ftp access to Data Pool directories and also provides a checksum-on-download service.
7. **Data Pool Update Granule Expiration utility (EcDIUpdateGranule.pl):** a perl utility, which allows operations staff to update the Data Pool expiration date and retention priority for specified Data Pool granules.
8. **Data Pool Batch Insert Utility (EcDIBatchInsert.pl):** a perl utility, which allows operations staff to queue ECS data for insert into the Data Pool.
9. **Data Pool Most Recent Insert Utility (EcDIMostRecentInsert.pl):** a perl utility, which creates files at the file system and data collection level of the Data Pool directory structure which contain information about granules recently inserted into the Data Pool at those levels.
10. **Data Pool Collection Remapping Utility (EcDIRemap.pl):** a perl utility, which allows DAAC operations staff to remap all data in a Data Pool collection directory from one higher level collection group directory to another.
11. **Data Pool Move Collection Utility (EcDIMoveCollection.pl):** a perl utility, which allows DAAC operations staff to move a Data Pool collection from one file system to another.
12. **Data Pool Density Map Utility (EcDIDensityMapUtility.pl):** a perl utility, which calculates spatial density map information about Data Pool collections and stores this information in the Data Pool database. This utility normally runs as a cron job.
13. **Data Pool Statistics Table Population Utility (EcDIPopulateStatTables.pl):** a perl utility, which populates tables in the Data Pool database which maintain counts of granules by collection and collection group, for use by the Web Access drill down web pages. This utility normally runs as a cron job.
14. **Data Pool Hidden Scrambler Utility (EcDIHiddenScrambler.pl):** a perl utility, which creates new names for specified hidden directories, saves these names, renames the existing hidden directories, and updates existing FTP Pull links that point to the previous hidden directories to point to the corresponding renamed directory.
15. **Data Pool Database (DataPool[_<MODE>]):** a Sybase database which stores Data Pool inventory and configuration information.

4.14.1 Data Pool Subsystem Context

Figure 4.14-1 is the Data Pool Subsystem context diagram. The diagram shows the interaction of the Data Pool Subsystem with other EMD subsystems. Table 4.14-1 provides descriptions of the interface events shown in the Data Pool Subsystem context diagram.

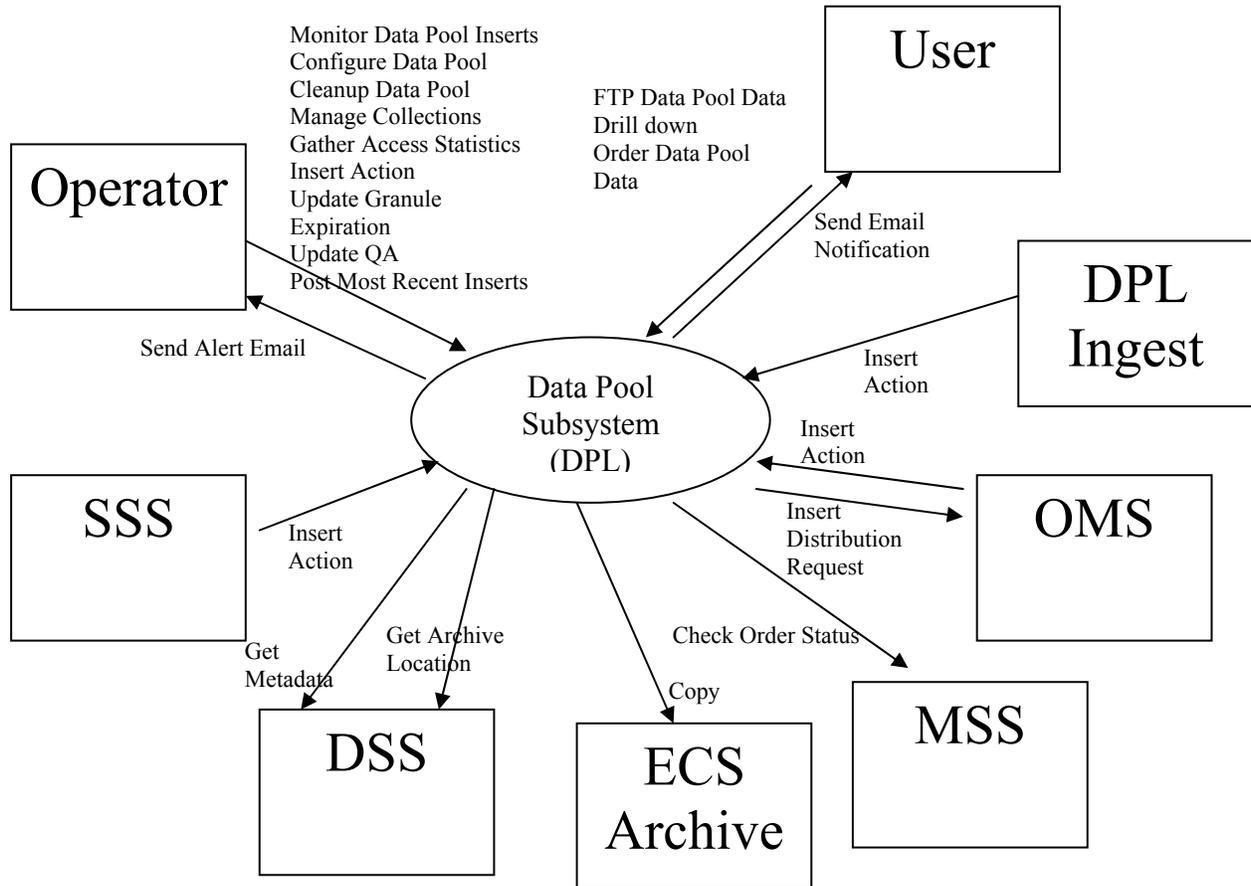


Figure 4.14-1. Data Pool Subsystem Context Diagram

Table 4.14-1. Data Pool Subsystem Interface Events (1 of 3)

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

Table 4.14-1. Data Pool Subsystem Interface Events (2 of 3)

Interface Event	Interface Event Description
Cleanup Data Pool	The operator runs the Data Pool Cleanup utility to clean expired granules out of the Data Pool, and to identify and cleanup granules, which are orphans (on Data Pool disk but not in the database) or phantoms (in the Data Pool database but not on disk).
Manage Collections	The operator uses the Data Pool Maintenance GUI to add, remove, or change specifications for Data Pool collections. The operator uses the Remap Collection utility to map a collection from one collection group to another. The operator uses the Move Collection utility to move a collection from one file system to another.
Gather Access Statistics	The operator uses the access statistics rollup scripts for the firewall ftp and web access logs to gather statistics about end user access to data pool files, and to store those statistics in the Data Pool database.
Insert Action	The operator uses the Batch Insert utility to insert historical data from the ECS archive into the Data Pool.
Update Granule Expiration	The operator uses the Update Granule Expiration utility to update the expiration date or retention priority for a Data Pool granule or set of granules.
Update QA	The operator uses the QA Update utility to propagate updates of QA information from the SDSRV to the Data Pool.
Post Most Recent Inserts	The operator uses the Most Recent Inserts utility to post information about recent Data Pool Inserts to the Data Pool ftp directories.
FTP Data Pool data	The end user uses the customized WU-FTP service to download Data Pool data.
Drill Down	The end user uses the Web Access web pages to perform searches for Data Pool data.
Order Data Pool data	The end user uses the Web Access web pages to order Data Pool data for ftp or media distribution. The end user may choose to convert, reformat, or subset the data using the HDF-EOS to GeoTiff Conversion Tool (HEG).
Send email notification	The DPL subsystem (Web Access component) sends email to the end user indicating that the user's Data Pool order has been submitted. Email is sent by the WebAccess component only for downloads without HEG conversion, and only if the user requests email. (OMS sends order acknowledgement and distribution notice emails).
Insert action	The Data Pool Ingest subsystem inserts a Data Pool insert action into the Data Pool Insert Action Queue (DIInsertActionQueue) for granules which are configured to be published in the Data Pool.
Insert action	The OMS subsystem inserts a Data Pool insert action into the Data Pool Insert Action Queue (DIInsertActionQueue) for granules to be staged to the Data Pool for ECS distribution requests. .
Insert Distribution Request	The DPL subsystem (WebAccess component) inserts distribution requests in the OMS database for Data Pool orders placed using the Data Pool Web Access web pages.

Table 4.14-1. Data Pool Subsystem Interface Events (3 of 3)

Interface Event	Interface Event Description
Check Order Status	The DPL subsystem (WebAccess component) checks status of orders in the MSS database.
Copy	The DPL subsystem copies data from the ECS Archive to the appropriate Data Pool file system.
Get Archive Location	The DPL subsystem looks up archive location information in the STMGT database, for granules which will be copied from the ECS archive to the Data Pool.
Get Metadata	The DPL subsystem gets metadata about ECS granules from the SDSRV database, and uses this metadata to store corresponding metadata in the Data Pool database and to create an xml metadata file on Data Pool disk.
Insert Action	The Spatial Subscription Server subsystem inserts Data Pool insert actions in the Data Pool Insert Action Queue (DIInsertActionQueue) for granules which are being inserted into the ECS inventory for which a Data Pool insert subscription is placed. Data Pool insert subscriptions are qualified subscriptions (unqualified Data Pool insert subscriptions have been replaced by DPL Ingest configuration of ESDTs for public Data Pool insert.)

4.14.2 Data Pool Hardware Context

Figure 4.14-2 is the Data Pool hardware context diagram. The diagram shows the interaction of the Data Pool custom code and COTS (in *italics*) with EMD hardware components.

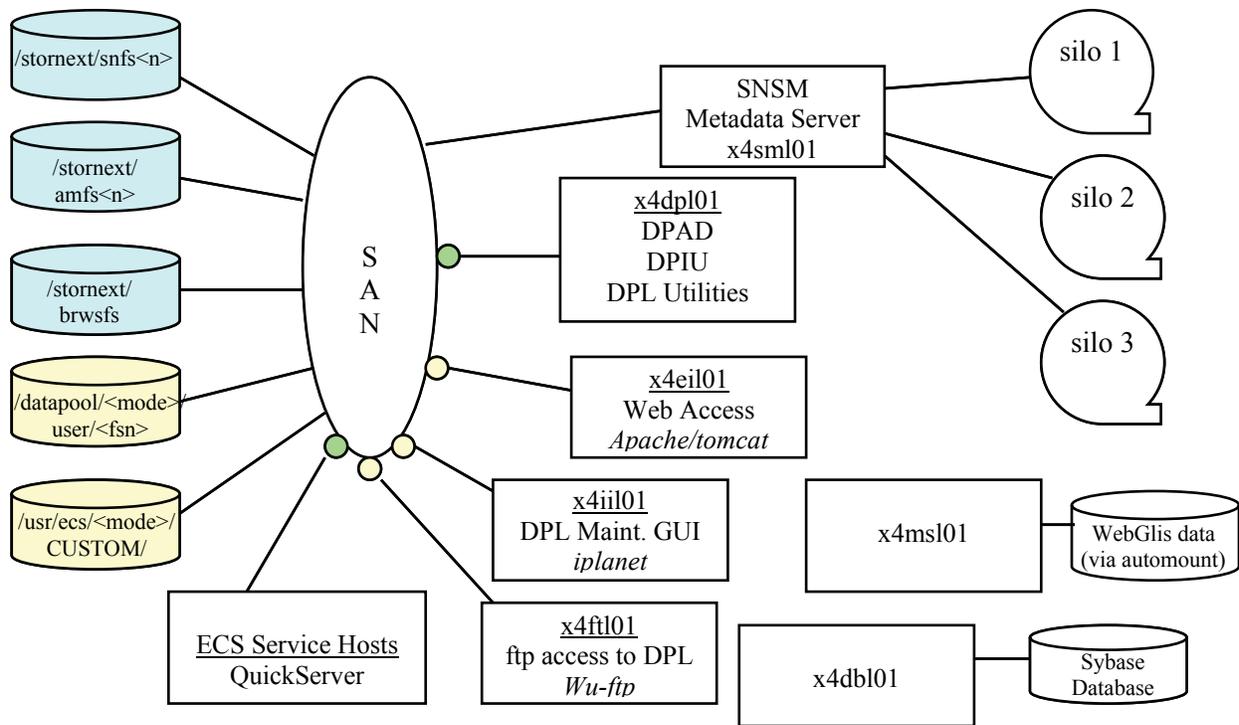


Figure 4.14-2. Data Pool Hardware Context

4.14.3 Data Pool Insert CSCI Functional Overview

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

A functional overview of the two-step Data Pool Insert process for ECS granules is shown below in two diagrams. The first diagram (Figure 4.14-3) shows the process for registration of a granule in the Data Pool. The second diagram (Figure 4.14-4) shows the process for publication of a granule in the Data Pool.

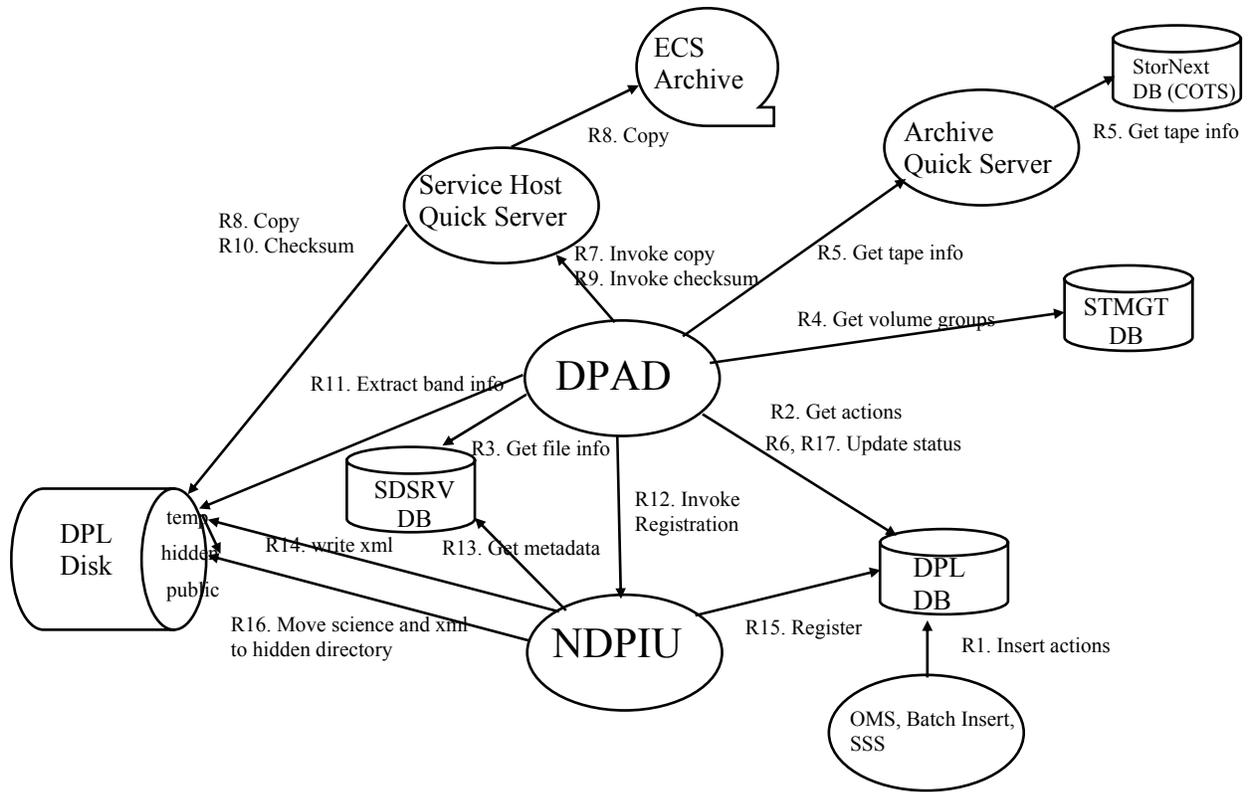


Figure 4.14-3. Data Pool Insert CSCI Architecture Diagram – Registration

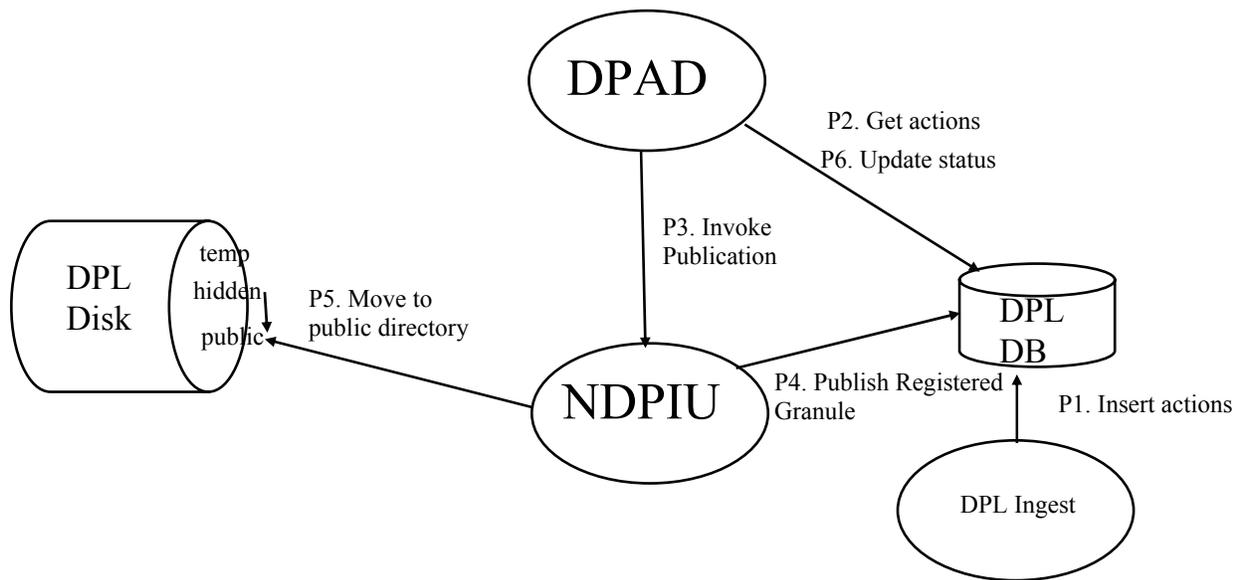


Figure 4.14-4. Data Pool Insert CSCI Architecture Diagram - Publication

There are four use cases for the Data Pool Insert process, one for each ECS process or component which requests insertion of a granule into the Data Pool. These use cases, and their relationship to the registration and publication steps, is shown in Table 4.14-2.

Table 4.14-2. Use Cases for Data Pool Insert

Requestor	Context	Data Pool Insert processes
OMS	stages granules in the hidden Data Pool for distribution	Registration (events R1 – R17 in Table 4.14-4)
Data Pool Ingest	requests publication of a granule in the Data Pool after the granule has been ingested and archived	Publication (events P1 – P6 in Table 4.14-4)
Batch Insert Utility	queues existing ECS granules for insertion into the public Data Pool	Registration Publication (events R1-R16, and P3-P6 in Table 4.14-4)
Spatial Subscription Server	queues granules for insertion into the public Data Pool as a result of a qualified subscription with a Data Pool insert action	Registration Publication (events R1-R16, and P3-P6 in Table 4.14-4)

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

Table 4.14-3 provides a process description for each of the major custom code components of the Data Pool insert process. Table 4.14-4 describes the interface events among the Data Pool insert process components for registration and publication.

Table 4.14-3. Data Pool Insert CSCI Process Description (1 of 2)

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcDIActionDriver	Server	DPLHW	Developed	EcDIActionDriver (DPAD) is a C++ server that is responsible for dispatching Data Pool insert actions for ECS granules from the insert action queue in the Data Pool database (DIInsertActionQueue).

Table 4.14-3. Data Pool Insert CSCI Process Description (2 of 2)

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcDIInsertUtility	Java utility	DPLHW	Developed	EcDIInsertUtility (NDPIU) is a java executable that is invoked by the EcDIActionDriver to register and publish ECS granules in the Data Pool. One copy of the EcDIInsertUtility is invoked for each ECS granule to be inserted in the Data Pool.
EcDIQuickServer	Server	ACMHW, DIPHW, DRPHW, SPRHW, MSSHW, AITHW, CSSHW, DMGHW, DPSHW, INTHW, DPLHW, OMSHW	Developed	The EcDIQuickServer (Service Host Quick Server) is a C++ server which performs CPU-intensive operations, such as copy and checksum, on ECS service hosts.
EcDIM2XT	Utility	DPLHW	Developed	Java utility that gets granule metadata from the SDSRV database and constructs an XML file.
hdf2jpeg	Utility	DPLHW	Developed	Java utility that extracts jpeg's from an HDFEOS granule.
bandtool	Utility	DPLHW	Developed	C utility that extracts band information from an HDFEOS granule.

Table 4.14-4. Data Pool ECS Insert CSCI Process Interface Events (1 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
R1. Insert Action	One per granule inserted in SDSRV which qualifies for existing subscription with Data Pool Insert action	Database: DataPool (DIInsertActionQueue)	Trigger: TrigInsEcNbDpEventDetails. sql	When a granule is inserted into SDSRV which matches an existing subscription with Data Pool Insert action, the trigger inserts a row into the DIInsertActionQueue in the Data Pool database with actionSource = null.

Table 4.14-4. Data Pool ECS Insert CSCI Process Interface Events (2 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
R1. Insert Action	One per granule in Syn IV order placed through Order Manager	Database: DataPool	Process: OmServer Stored Proc: OmInsDPLAction	When a granule is ordered in Syn IV mode via the Order Manager, OMS inserts a row into the DIInsertActionQueue in the Data Pool database, with actionSource = O.
R1. Insert Action	One per granule in input file for the Batch Insert utility.	Database: DataPool	Utility: EcDIBatchInsert.pl	For each valid granule in its input file, the Batch Insert Utility inserts a row into the DIInsertActionQueue in the Data Pool database, with actionSource = B.
R2. Get Action	Continuously, as long as there are actions in DIInsertActionQueue with status = null or status = RETRY. If no actions, once per configured time interval (IdleSleep in DIConfig)	Database: DataPool	Process: EcDIActionDriver (DPAD)	The DPAD gets batches of actions (with status = null or status = RETRY) from the DIInsertActionQueue.
R3. Get file info	Once per file per ECS granule to be inserted	Database: SDSRV (EcDsScienceData Server1) (DsMdUserDataFile)	Process: EcDIActionDriver (DPAD)	The DPAD gets file name information for each file in the granule from the SDSRV database.
R4. Get volume groups	Once per ECS granule to be inserted	Database: STMGT (DsStVolumeGroup)	Process: EcDIActionDriver (DPAD)	The DPAD gets the name of the open volume group for the granule's collection (shortname, versionid) from the STMGT database.
R5. Get tape info	Once per file per ECS granule to be inserted	Process: Java Quick Server Database: COTS StorNext metadata database	Process: EcDIActionDriver (DPAD)	The DPAD calls the Java Quick Server, which runs on the StorNext (COTS) metadata server, to retrieve tape label information for the granule files, based on the volume group information from the STMGT database.

Table 4.14-4. Data Pool ECS Insert CSCI Process Interface Events (3 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
R6. Update status	Once per ECS granule to be inserted	Database: Data Pool (DIInsertActionQueue, DIActiveInsertProcesses)	Process: EcDIActionDriver (DPAD)	The DPAD updates the status of the insert in the Data Pool database.
R7. Invoke copy	Once per science file per ECS granule	Process: EcDIQuickServer	Process: EcDIActionDriver (DPAD)	The DPAD chooses a QuickServer on an ECS Service Host to perform the file copy operation. The Service Host is chosen based on availability.
R8. Copy	Once per science file per ECS granule	Storage Device: Data Pool disk (managed by COTS StorNext Storage Area Network) Storage Device: ECS Archive tape or cache (managed by COTS StorNext)	Process: EcDIQuickServer EcUtCopyExec	The QuickServer, running on an ECS Service Host, uses the EcUtCopyExec to copy the science file from the ECS Archive (tape or cache) to the Data Pool file system associated with the granule's collection.
R9. Invoke checksum	Once per science file per ECS granule	Process: EcDIQuickServer	Process: EcDIActionDriver (DPAD)	The DPAD chooses a QuickServer on an ECS Service Host to perform the file checksum operation. The Service Host is chosen based on availability.
R10. Checksum	Once per science file per ECS granule	Storage Device: Data Pool disk (managed by COTS StorNext Storage Area Network) Storage Device: ECS Archive tape or cache (managed by COTS StorNext)	Process: EcDIQuickServer	The QuickServer, running on an ECS Service Host, checksums the science file on the Data Pool file system (temp directory).
R11. Extract band info	Once per ECS science granule, where the collection is enabled for HEG conversion (i.e., convertEnabledFlag is on for the collection)	Storage Device: temp directory in Data Pool file system	Process: EcDIActionDriver (DPAD) bandtool	The DPAD uses the bandtool utility to extract band information from the science granule, and writes band information to a temporary file in the Data Pool file system

Table 4.14-4. Data Pool ECS Insert CSCI Process Interface Events (4 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
R12. Invoke registration	Once per ECS science granule	Process: EcDIInsertUtility (NDPIU)	Process: EcDIActionDriver (DPAD)	The DPAD invokes an instance of the NDPIU from a pool to perform the granule registration.
R13. Get metadata	Once per ECS science granule	Database: SDSRV (EcDsScienceData Server1)	Process: EcDIInsertUtility (NDPIU), M2XT (EcDIM2XTApp)	The NDPIU gets metadata about the ECS science granule from the SDSRV database.
R14. Write xml	Once per ECS science granule	Storage Device: temp directories in Data Pool file system	Process: EcDIInsertUtility (NDPIU)	The NDPIU writes the xml metadata file for the granule to the temp directory on the Data Pool file system.
R15. Register	Once per ECS science granule	Database: Data Pool	Process: EcDIInsertUtility (NDPIU)	The NDPIU populates basic tables in the Data Pool database with inventory information about the granule.
R16. Move science and xml to hidden directory	Once per science file and xml file per ECS science granule	Storage Device: temp and hidden directories in Data Pool file system	Process: EcDIInsertUtility (NDPIU)	The NDPIU moves the science file(s) and xml file for the granule from the temp directory in the Data Pool file system to the appropriate hidden directory (under /.orderdata).
[R17. Update status]	Once per insert request	Database: Data Pool	Process: EcDIActionDriver (DPAD)	The DPAD updates the insert request status in the DIInsertActionQueue.
[P1. Insert action]	Once per publication request	Database: Data Pool (DIInsertActionQueue)	Process: Data Pool Ingest Processing	The DPL Ingest Processing server places a request for granule publication in the DIInsertActionQueue.
[P2. Get actions]	Continuously, as long as there are actions in DIInsertActionQueue with status = null or status = RETRY. If no actions, once per configured time interval (IdleSleep in DIConfig)	Database: DataPool	Process: EcDIActionDriver (DPAD)	The DPAD gets batches of actions (with status = null or status = RETRY) from the DIInsertActionQueue.

Table 4.14-4. Data Pool ECS Insert CSCI Process Interface Events (5 of 5)

Event	Event Frequency	Interface	Initiated By	Event Description
P3. Invoke publication	Once per ECS granule to be published in the Data Pool	Process: EcDIInsertUtility (NDPIU)	Process: EcDIActionDriver (DPAD)	The DPAD invokes an instance of the NDPIU from a pool to perform the granule publication.
P4. Publish registered granule	Once per ECS granule to be published in the Data Pool	Database: Data Pool	Process: EcDIInsertUtility (NDPIU)	The NDPIU populates additional tables in the Data Pool database with inventory information needed to support web access to the granule.
P5. Move to public directory	Once per ECS granule to be published in the Data Pool	Storage Device: hidden and public directories in Data Pool file system	Process: EcDIInsertUtility (NDPIU)	The NDPIU moves the science file(s) and xml file for the granule from the hidden directory in the Data Pool file system to the appropriate public directory
P6. Update status	Once per ECS granule to be published in the Data Pool	Database: Data Pool	Process: EcDIActionDriver (DPAD)	The DPAD updates the insert request status in the DIInsertActionQueue to the final request state, and removes the insert request from the DIActiveInsertProcesses table..

4.14.4 WebAccess CSCI Functional Overview

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

Figure 4.14-5 is the WebAccess CSCI architecture diagram. The diagram shows the events sent to the WebAccess CSCI processes and the events the WebAccess CSCI processes send to other processes.

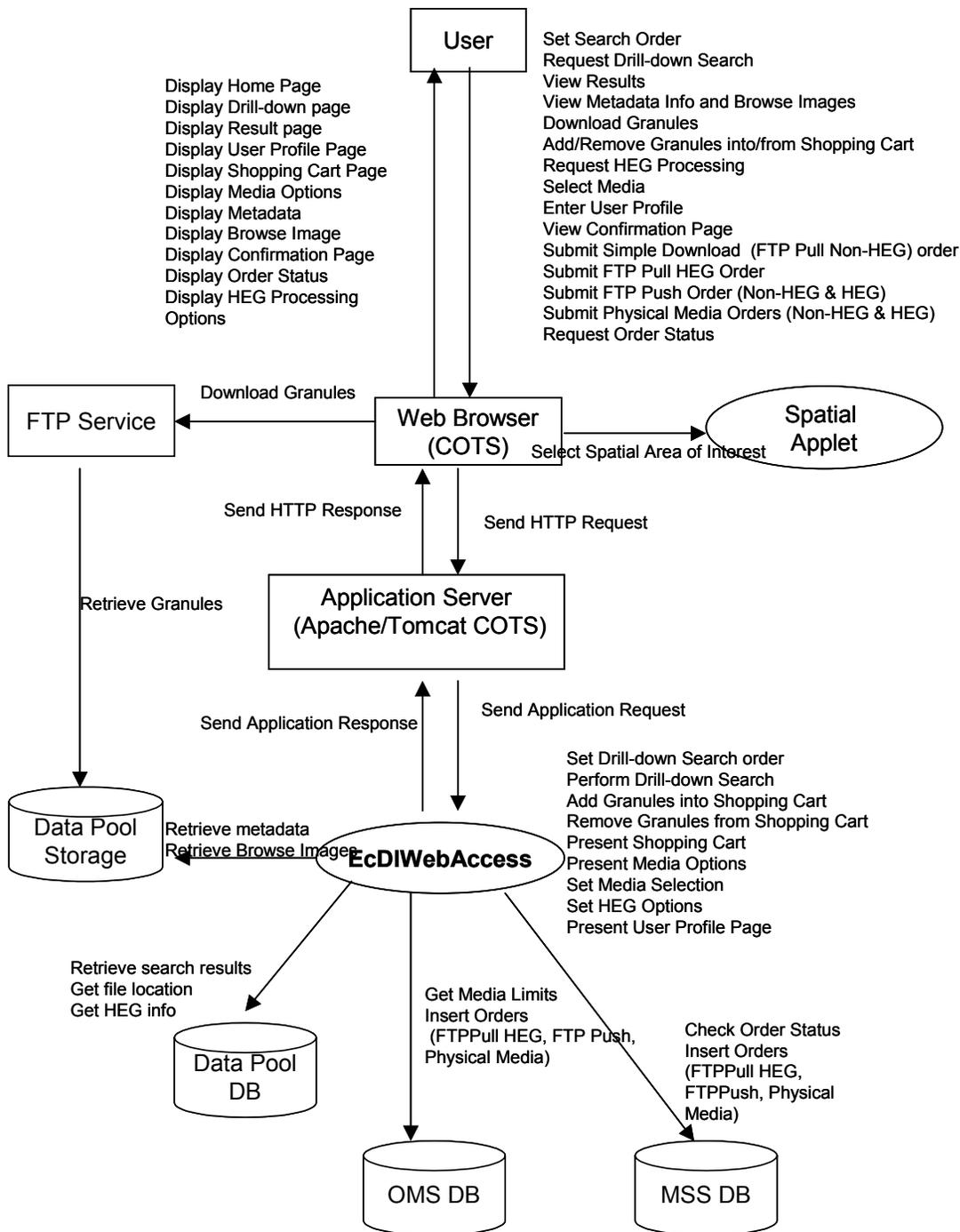


Figure 4.14-5. WebAccess CSCI Architecture Diagram

WebAccess Process Descriptions

Table 4.14-5 provides descriptions of the processes shown in the WebAccess architecture diagram.

Table 4.14-5. WebAccess CSCI Process Description

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

WebAccess Process Interface Descriptions

Table 4.14-6 describes the interface events among the WebAccess CSCI processes.

Table 4.14-6. WebAccess CSCI Process Interface Events (1 of 10)

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

Table 4.14-6. WebAccess CSCI Process Interface Events (2 of 10)

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

Table 4.14-6. WebAccess CSCI Process Interface Events (3 of 10)

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

Table 4.14-6. WebAccess CSCI Process Interface Events (4 of 10)

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

Table 4.14-6. WebAccess CSCI Process Interface Events (5 of 10)

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

Table 4.14-6. WebAccess CSCI Process Interface Events (6 of 10)

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

Table 4.14-6. WebAccess CSCI Process Interface Events (7 of 10)

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

Table 4.14-6. WebAccess CSCI Process Interface Events (8 of 10)

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

Table 4.14-6. WebAccess CSCI Process Interface Events (9 of 10)

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

Table 4.14-6. WebAccess CSCI Process Interface Events (10 of 10)

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

4.14.5 Data Stores

There are two data stores associated with the Data Pool subsystem. They are the Data Pool database (DPL DB) and the Order Manager database (OMS DB). Table 4.14-7 provides a description of these data stores.

Table 4.14-7. Data Pool Data Stores

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

4.15 Bulk Metadata Generation Tool Subsystem Overview

The EMD Bulk Metadata and Browse Export Capability were created to support the development of value-added providers (e.g., IIMS, ESIPs, RESACs, and InfoMarts). The Bulk Metadata Generation Tool (BMGT) provides interface to directly support the first part of the capability. Currently, EOS Clearinghouse (ECHO) is the primary consumer of this capability.

The BMGT will facilitate EMD sites generate and export an external representation of their metadata holdings. The format used for the external representation of the metadata is XML. BMGT and Bulk Browse Generator Tools are run as daily cron jobs at each site to populate these data collections. One metadata product is created per ESDT group per day. Each product will contain an external representation of the metadata for each new, updated, or deleted granule that is a member of the ESDT group. One bulk browse product is produced per day that contains references to all new, updated, or deleted browse granules. Value-added providers may use any of the standard EMD search, order, and subscription capabilities to find and order these bulk metadata and browse products.

EMD distributes bulk metadata and browse products to ECHO via SIPS interface.

The BMGT, in addition to above capability, provides mechanism to tag granules that were exported to ECHO with Data Pool FTP, OGC conformant Web Mapping and Web Coverage Service URLs. These URLs are published to ECHO through its update interface.

BMGT Subsystem Context

Figure 4.15-1 is the BMGT Subsystem context diagram. The diagram shows the events generated between BMGT and other subsystems.

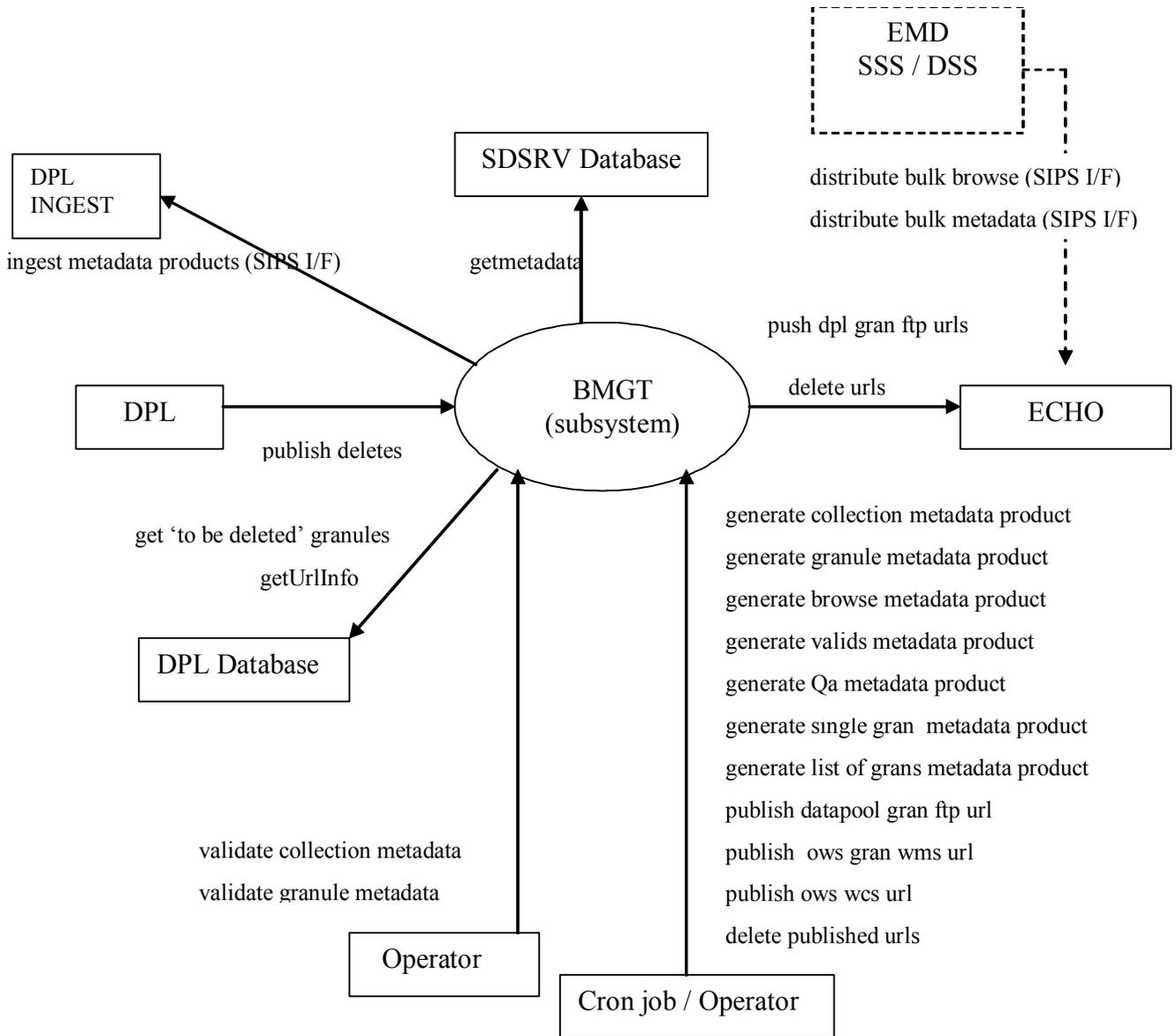


Figure 4.15-1. BMGT Subsystem Context Diagram

Table 4.15-1 provides descriptions of the interface events in the BMGT Subsystem context diagram.

Table 4.15-1. BMGT Subsystem Interface Events (1 of 2)

Event	Interface Event Description
generate collection metadata product	BMGT generates ECSMETC metadata product for archive and distribution purposes. This product contains XML representation of the collection level metadata and the packaging options that may be used when ordering products from each collection. BMGT reads target ESDTs metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes. BMGT includes in the product WmsUrl and WcsUrl additional attributes.
generate granule metadata product	BMGT generates ECSMETG metadata product for archive and distribution purposes. This product contains XML representation of the granule level metadata. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate browse metadata product	BMGT generates ECSBBR metadata product for archive and distribution purposes. This product contains XML representation of references to browse images. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate valids metadata product	BMGT generates ECSMETV metadata product for archive and distribution purposes. This product contains XML representation of EMD collection and granule valid values. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate QA metadata product	BMGT generates ECSMETU metadata product for archive and distribution purposes. This product contains XML representation of QA updates made to a granule. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate single gran metadata product	BMGT generates granule level metadata for the specified granule. A granule can be specified either by EMD dbID or Geoid. BMGT reads Metadata from SDSRV Database.
generate list of grans metadata product	BMGT generates granule level metadata for the specified granules. A list of EMD dbIDs or Geoids is made available to BMGT. BMGT reads Metadata from SDSRV Database.
publish Data Pool gran ftp url	BMGT/BulkURL generates a XML product containing Data Pool FTP URLs for the qualified granules. The product is delivered to ECHO via FTP push or pull. The product conforms to ECHO's update API. Information required to formulate FTP URL is read from Data Pool Database.

Table 4.15-1. BMGT Subsystem Interface Events (2 of 2)

Event	Interface Event Description
delete published urls	BMGT/BulkURL, upon invocation by Data Pool Cleanup utility, generates a XML product containing Data Pool granule URs that are targeted for removal from ECHO. The product is delivered to ECHO via FTP push or pull. ECHO will use this product to remove just the Data Pool related data like FTP, wms and wcs URLs. The product conforms to ECHO's update API.
validate collection metadata	BMGT validates its generated collection metadata (in xml format) with corresponding ODL Metadata file.
validate granule metadata	BMGT validates its generated granule metadata (in xml format) with corresponding ODL Metadata file.
GetUrlInfo	BMGT reads necessary info from Data Pool database to support 'publish' events mentioned above.
publish deletes	Data Pool Cleanup utility invokes BMGT/BulkURL to convey Data Pool granule deletes to ECHO. See 'delete published URLs' event above.
get 'to be deleted' granules	BMGT/BulkURL reads from Data Pool Database information pertaining to 'to be deleted granules'. Data Pool Cleanup utility persists this information before invoking BMGT/BulkURL.
ingest metadata products	BMGT generates ODL MET files and PDR files for all its Metadata products to facilitate 'Polling with Delivery Record Ingest'. This is part of SIPS interface.
Getmetadata	BMGT reads collection, granule, browse, valids, and QA related metadata from SDSRV Database.
distribute bulk browse	DSS distributes browse images if OMS is in SYN 3 mode, otherwise OMS distributes the browse images to ECHO via SIPS interface. This is not BMGT's responsibility. Included here to show how this event is triggered if an ECSBBR product is acquired.
distribute bulk metadata	EMD distributes BMGT Metadata products to ECHO via SIPS interface.

4.15.1 BMGT Architecture

Figure 4.15-2 displays the BMGT Architecture diagram.

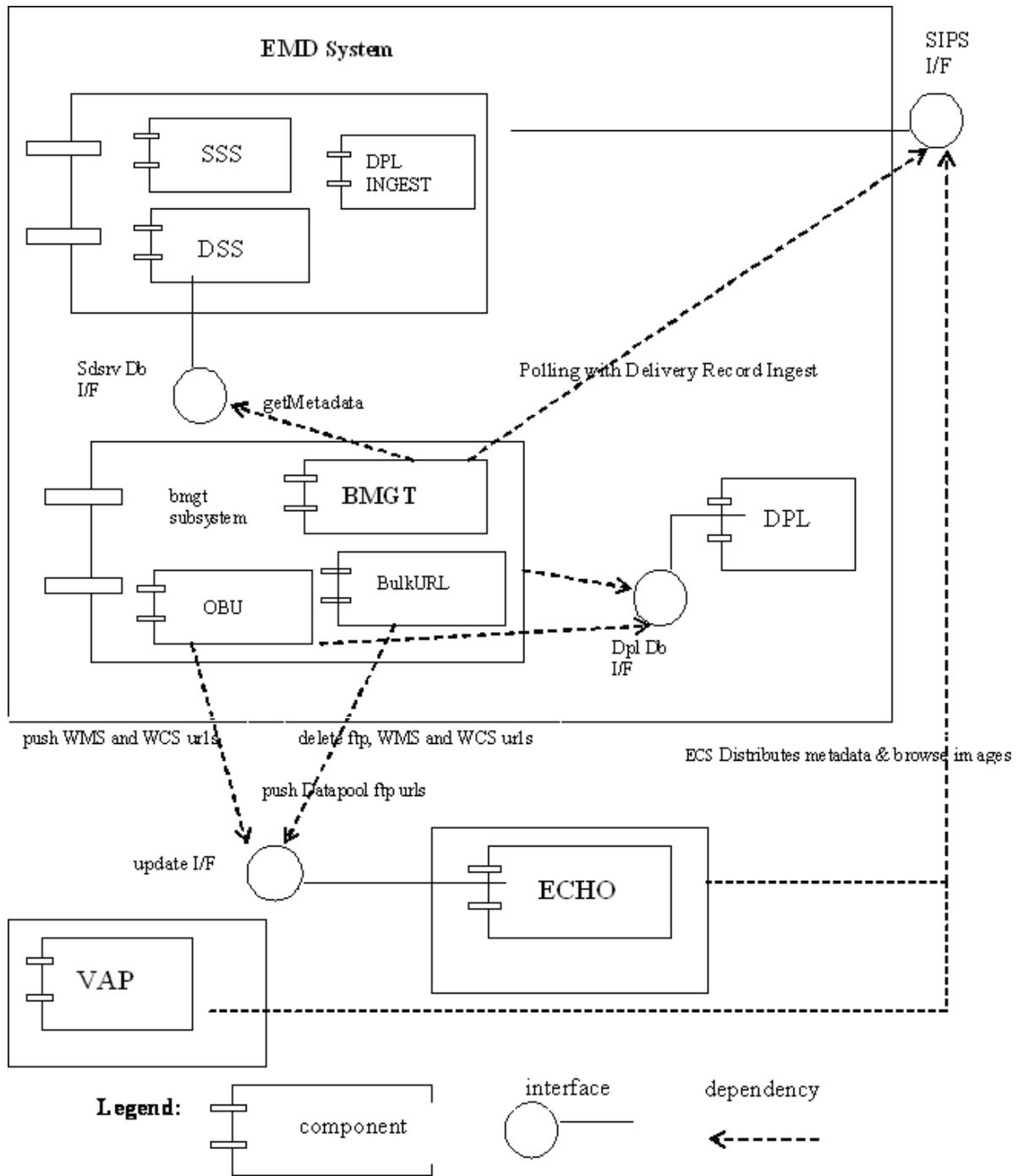


Figure 4.15-2. BMGT Architecture Diagram

Table 4.15-2 provides descriptions of processes shown in the architecture diagram.

Table 4.15-2. BMGT Processes

Process	Type	Hardware CI	COTS/ Developed	Functionality
EcBmBMGT ("BMGT")	Application	OMSHW	Developed	The BMGT when run either as a cron job or manually, will generate ECSMETC, ECSMETG, ECSBBR, ECSMETV and the ECSMETU type xml products per its configuration. The products contain metadata about inventory that was inserted, updated, and/or deleted from the SDSRV database. Also, it generates a MET and PDR files for ingest purposes.
EcBmBulkURL ("BulkURL")	Application	DPLHW	Developed	The BulkURL exports to ECHO Data Pool FTP URLs and delete information about granule FTP, WCS and WMS URLs for the qualified granules. Export is achieved via FTP push or pull mechanism.
EcBmBMGTValidateCollection	Script	OMSHW	Developed	A specified collection's generated metadata, in xml format, is validated. Script takes one ODL metadata file and one XML file as input parameters. The output of each script consists of a difference between the ODL metadata file and BMGT XML file.
EcBmBMGTValidateGranule	Script	OMSHW	Developed	A specified granule's generated metadata, in xml format, is validated. Script takes one ODL metadata file and one XML file as input parameters. The output of each script consists of a difference between the ODL metadata file and BMGT XML file.
Sybase ASE	Server	ACMHW	COTS	The Sybase ASE is where the SDSRV and Data Pool databases reside.

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

Use of COTS in the BMGT Subsystem

- JRE
The JRE constitutes Java virtual machine and the Java platform core libraries. It provides applications with Java platform. Included with it is JAXP (Java API for XML Processing).
- jConnect
The jConnect implements JDBC interface and provides application with drivers to access Sybase database SQL server.
- JDOM
JDOM libraries allow java applications to create and edit xml documents.
- Sybase Server
The BMGT accesses SDSRV database to read inventory metadata and update Data Pool collection level attribute.
- JAF / Javamail
Java Activation Framework (JAF) and Javamail provide EcBmBulkURL the capability to send email messages.
- JWSDP
The subsystem utilizes Java Architecture for XML Binding (JAXB) functionality of jwsdp package. JAXB is a Java technology that allows easy binding of XML schemas to Java objects. Thus it helps application edit and create xml documents conforming to schemas.
- Sun Java System Web Server
BMGT (Document Type Definitions) DTD schemas are hosted on the web server to provide access to consumers, like ECHO, who intend to validate the xml products.
- Perl
Validation tools – EcBmBMGTValidateCollection and EcBmBMGTValidateGranule need perl.

4.15.2 The BMGT Subsystem Software Description

As shown in Figure 4.15-2 architecture diagram, the BMGT Subsystem consists of three CSCIs: BMGT, BulkURL and the OBU. Essentially, these are Java applications

4.15.2.1 BMGT CSCI Functional Overview

BMGT reads SDSRV Database to detect collections, granules and browse granules etc that have been inserted, updated or deleted during the specified period. Upon detection it will generate XML representation of identified object's, say granule, metadata. The generated XML products enter into

EMD as new granules. Products are archived for distribution purposes. XML products belong to one of ECSMETC, ECSMETG, ECSBBR, ECSMETV or ECSMETU collections.

BMGT product generation rules in terms of what type of products to generate, what target ESDTs to include, time range etc are configurable.

For each of the archivable xml products, BMGT creates a MET file. Also, it creates PDR file for each type of product. BMGT places XML products, MET and PDR files in a polling directory for DPL Ingest pick up. DPL Ingest interface is per SIPS interface. After successful archiving, the products are pushed to ECHO via SIPS interface.

Each of the product types i.e., BMGT specific ESDTs are described below:

- ECSMETC – Stores products that contain an XML representation of EMD collection level metadata and the packaging options that may be used when ordering products from each collection;
- ECSMETG – Stores products that contain an XML representation of EMD granule level metadata;
- ECSBBR - Stores products that contain an XML representation of references to browse images.
- ECSMETV – Stores products that contain an XML representation of EMD collection and granule valid values.
- ECSMETU – Stores products that contain an XML representation of EMD granule level QA Updates.

The ECSMETC and ECSMETG data collections store products that contain metadata for multiple collections and multiple granules. The metadata will be grouped by instrument and mission except for metadata related to the MODIS instrument, which is grouped, by mission and major discipline (ocean, atmosphere, land, and snow & ice). Each product in these collections has a group identifier Product Specific Attribute (PSA) called GroupId. The mapping of specific ESDTs to groups is provided as a configuration file with the BMGT.

The ECSBBR collection stores products that contain browse product references. The ESDT has a custom acquire service that will convert the browse product references into actual browse products during distribution.

The ECSMETV collection stores products that contain the entire set of valids contained within the SDSRV database for a particular instance in time.

Each EMD site will run BMGT as a daily cron job. It can also be run manually for any specified duration or to recover from cron job failures.

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

For reference, DTD schemas are included in Section 4.15.2.13.

4.15.2.2 BMGT CSCI Context

Figure 4.15-3 is the BMGT CSCI context diagram.

Table 4.15-3 provides descriptions of the interface events shown in the BMGT CSCI context diagram.

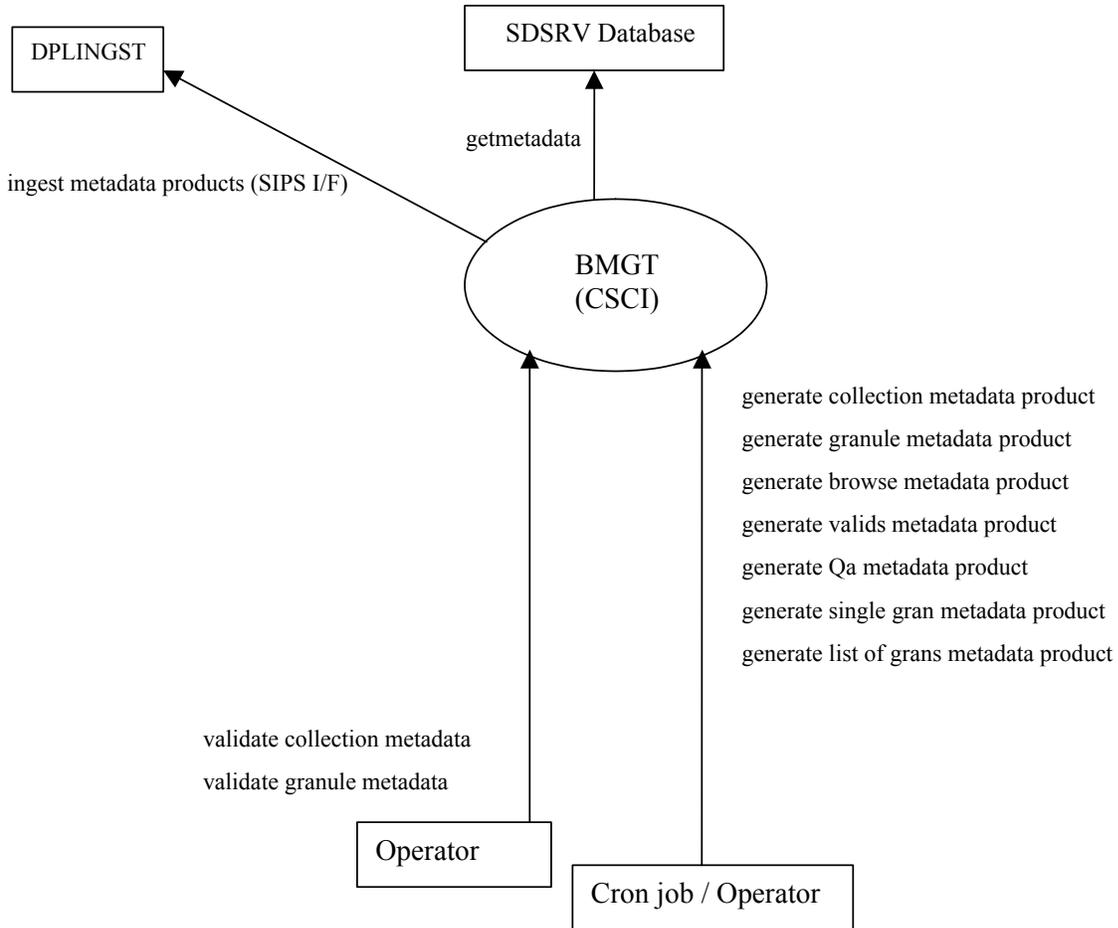


Figure 4.15-3. BMGT CSCI Context Diagram

Table 4.15-3. BMGT Subsystem Interface Events (1 of 2)

Event	Interface Event Description
generate collection metadata product	BMGT generates ECSMETC metadata product for archive and distribution purposes. This product contains XML representation of the collection level metadata and the packaging options that may be used when ordering products from each collection. BMGT reads target ESDTs metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes. BMGT includes in the product WmsUrl and WcsUrl additional attributes.
generate granule metadata product	BMGT generates ECSMETG metadata product for archive and distribution purposes. This product contains XML representation of the granule level metadata. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate browse metadata product	BMGT generates ECSBBR metadata product for archive and distribution purposes. This product contains XML representation of references to browse images. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate valids metadata product	BMGT generates ECSMETV metadata product for archive and distribution purposes. This product contains XML representation of EMD collection and granule valid values. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate QA metadata product	BMGT generates ECSMETU metadata product for archive and distribution purposes. This product contains XML representation of QA updates made to a granule. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate single gran metadata product	BMGT generates granule level metadata for the specified granule. A granule can be specified either by EMD dbID or Geoid. BMGT reads Metadata from SDSRV Database.
generate list of grans metadata product	BMGT generates granule level metadata for the specified granules. A list of EMD dbIDs or Geoids is made available to BMGT. BMGT reads Metadata from SDSRV Database.
publish Data Pool gran ftp url	BMGT/BulkURL generates a XML product containing Data Pool FTP URLs for the qualified granules. The product is delivered to ECHO via FTP push or pull. The product conforms to ECHO's update API. Information required to formulate FTP URL is read from Data Pool Database.

Table 4.15-3. BMGT Subsystem Interface Events (2 of 2)

Event	Interface Event Description
delete published urls	BMGT/BulkURL, upon invocation by Data Pool Cleanup utility, generates a XML product containing Data Pool granule URs that are targeted for removal from ECHO. The product is delivered to ECHO via FTP push or pull. ECHO will use this product to remove just the Data Pool related data like FTP, wms and wcs URLs. The product conforms to ECHO's update API.
validate collection metadata	BMGT validates its generated collection metadata (in xml format) with corresponding ODL Metadata file.
validate granule metadata	BMGT validates its generated granule metadata (in xml format) with corresponding ODL Metadata file.
GetUrlInfo	BMGT reads necessary info from Data Pool database to support 'publish' events mentioned above.
publish deletes	Data Pool Cleanup utility invokes BMGT/BulkURL to convey Data Pool granule deletes to ECHO. See 'delete published URLs' event above.
get 'to be deleted' granules	BMGT/BulkURL reads from Data Pool Database information pertaining to 'to be deleted granules'. Data Pool Cleanup utility persists this information before invoking BMGT/BulkURL.
ingest metadata products	BMGT generates ODL MET files and PDR files for all its Metadata products to facilitate 'Polling with Delivery Record Ingest'. This is part of SIPS interface.
Getmetadata	BMGT reads collection, granule, browse, valids and QArealted metadata from SDSRV Database.
distribute bulk browse	DSS distributes browse images if OMS is in SYN 3 mode, otherwise OMS distributes the browse images to ECHO via SIPS interface. This is not BMGT's responsibility. Included here to show how this event is triggered if an ECSBBR product is acquired.
distribute bulk metadata	EMD distributes BMGT Metadata products to ECHO via SIPS interface.

4.15.2.3 BMGT CSCI Process Interface Description

Table 4.15-4 provides descriptions of the interface events shown in the BMGT context diagram.

Table 4.15-4. BMGT CSCI Process Interface Events (1 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Generate collection metadata product	Once per service request	<i>Process:</i> EcBmBMGT <i>Collaborator:</i> ProcBmgtGetEsdCollMetadata	<i>Process:</i> Cron or operator	<p>Based on target ESDTs and period of interest specified in EcBmBMGTGroup.xml and EcBmBMGTUserParams.xml, BMGT generates ECSMETC xml product. Using a storedproc it reads metadata from SDSRV database. Inserts and updates are included in the product.</p> <p>The product conforms to BMGTCollectionMetadata.dtd schema. One logical product is generated per group, which is a set of ESDTs. In addition, a MET file and PDR is generated.</p> <p>BMGT includes in the product WmsUrl and WcsUrl additional attributes.</p>
Generate granule metadata product	Once per service request	<i>Process:</i> EcBmBMGT <i>Collaborator:</i> ProcBmgtGetEsdGranMetadata	<i>Process:</i> Cron or operator	<p>Based on target ESDTs and period of interest specified in EcBmBMGTGroup.xml and EcBmBMGTUserParams.xml, BMGT generates ECSMETG xml product. Using a storedproc it reads metadata from SDSRV database. Inserts, updates and deletes are included in the product.</p> <p>The product conforms to BMGTGranuleMetadata.dtd schema (Section 4.15.2.13). One logical product is generated per group, which is a set of ESDTs. In addition, a MET file and PDR is generated.</p>

Table 4.15-4. BMGT CSCI Process Interface Events (2 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Generate browse metadata product	Once per service request	<i>Process:</i> EcBmBMGT <i>Collaborator:</i> ProcBmgtGetEsdBrowMetadata	<i>Process:</i> Cron or operator	<p>For all browse images that were inserted or deleted within a specified time period, extract the browse identifiers and associated browse file names for each browse product and insert an XML file, called the Browse Reference File (BRF) file, as a product into the ECSBBR data collection.</p> <p>The product conforms to BMGTBrowseMetadata.dtd schema. In addition, a MET file (per configured set of browse cross-references) and PDR is generated.</p>
Generate valids metadata product	Once per service request	<i>Process:</i> EcBmBMGT <i>Collaborator:</i> ProcBmgtGetValdMetadata	<i>Process:</i> Cron or operator	<p>In a process similar to granules, ECSMETV product is generated. Specifically, if any collections were inserted, updated, and/or deleted during the period then an XML representation of the valids information is generated, and inserted as a product, into the ECSMETV data collection setting the starting date and ending date of the insert, update, and/or delete activity covered by this file.</p> <p>The product conforms to BMGTValidMetadata.dtd schema. In addition, a MET file and PDR is generated.</p>

Table 4.15-4. BMGT CSCI Process Interface Events (3 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Generate QA metadata product	Once per service request	<i>Process:</i> EcBmBMGT <i>Collaborator:</i> ProcBmgtGetEsdQaUpdates	<i>Process:</i> Cron or operator	BMGT determines granules for which only QA flag metadata updates have been made and generates ECSMETU xml product for the identified granules. These products are used for notifying ECHO about just the QA updates. QAMUT persists QA update changes in SDSRV tables for BMGT consumption. The product conforms to BMGTUpdateMetadata.dtd schema . In addition, a MET file and PDR is generated.
Validate collection metadata	One per service request	<i>Process:</i> EcBmBMGTValidateCollection	<i>Process:</i> operator	The script takes one ODL metadata file and one XML file as input parameters. The output of each script consists of a difference between the ODL metadata file and BMGT XML file. The BMGT XML collection file can contain multiple granules, but only the collection associated with the single ODL metadata file gets validated. All other collections are ignored. The same applies for granule XML files. In order to obtain the ODL metadata file, the associated granule must be ordered from EMD.
Validate granule metadata	One per service request	<i>Process:</i> EcBmBMGTValidateGranule	<i>Process:</i> operator	Similar to above collection validation.

4.15.2.4 Data Stores

BMGT uses SDSRV DB to generate its products. Table 4.15-5 describes the Data Stores.

Table 4.15-5. CSCI Data Stores

Data Store	Type	Description
SDSRV DB	Sybase	BMGT reads required metadata from SDSRV DB.

4.15.2.5 BulkURL CSCI Functional Overview

The BulkURL utility (EcBmBulkURL) exports to ECHO the metadata content of products in the Data Pool, including their FTP URLs, based on an assumption that the product is an ECS data type and that the information about the product in the EMD Science Data Server has already been exported to ECHO by the Bulk Metadata Generation Tool (BMGT).

The utility can be run either with ‘insert’ or ‘delete’ option. With the Insert option, the BulkURL utility is run on a daily cron job or from the command line. Data Pool Cleanup utility invokes BulkURL with ‘delete’ option.

Specifically, it will export an xml representation of the FTP URL information for science files, metadata files, and browse files associated with Data Pool granules.

It also exports Data Pool delete information to ECHO. Delete message sent to ECHO is a request for it to remove any FTP, wms and wcs URLs associated with the deleted Data Pool granules.

BulkURL generated products conform to BMGTUpdateMetadata.dtd schema. Export is effected via FTP push or pull.

For reference, DTD schemas are included in Section 4.15.2.13.

4.15.2.6 BulkURL CSCI Context

Figure 4.15-4 is the BulkURL CSCI context diagram and Table 4.15-6 describes the list of BMGT subsystem interface events.

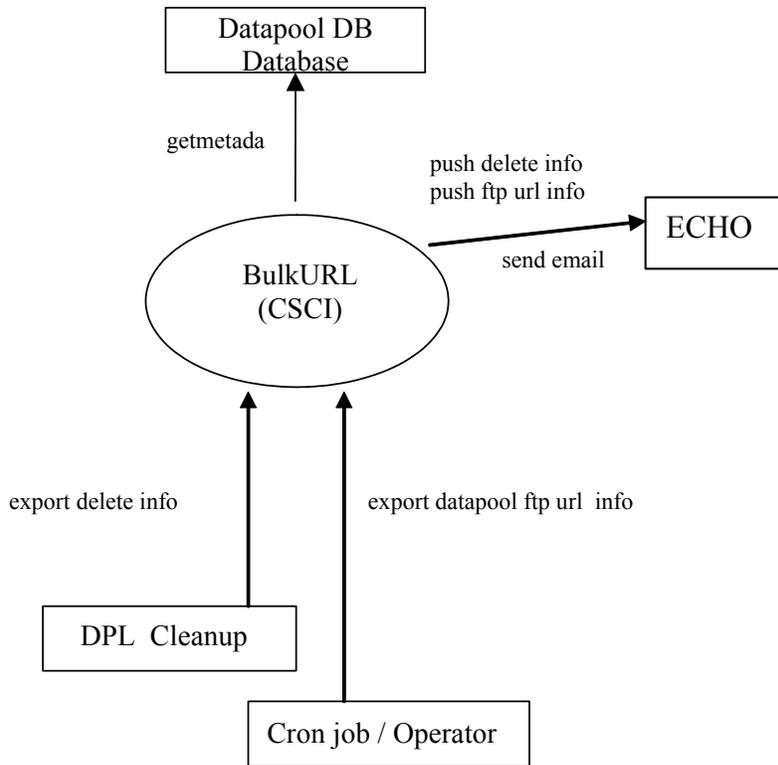


Figure 4.15-4. BulkURL CSCI Context Diagram

Table 4.15-6. BMGT Subsystem Interface Events (1 of 2)

Event	Interface Event Description
generate collection metadata product	BMGT generates ECSMETC metadata product for archive and distribution purposes. This product contains XML representation of the collection level metadata and the packaging options that may be used when ordering products from each collection. BMGT reads target ESDTs metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes. BMGT includes in the product WmsUrl and WcsUrl additional attributes.
generate granule metadata product	BMGT generates ECSMETG metadata product for archive and distribution purposes. This product contains XML representation of the granule level metadata. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate browse metadata product	BMGT generates ECSBBR metadata product for archive and distribution purposes. This product contains XML representation of references to browse images. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate valids metadata product	BMGT generates ECSMETV metadata product for archive and distribution purposes. This product contains XML representation of EMD collection and granule valid values. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.

Table 4.15-6. BMGT Subsystem Interface Events (2 of 2)

Event	Interface Event Description
generate QA metadata product	BMGT generates ECSMETU metadata product for archive and distribution purposes. This product contains XML representation of QA updates made to a granule. BMGT reads Metadata from SDSRV Database. MET files and a single PDR file are generated for EMD ingest purposes.
generate single gran metadata product	BMGT generates granule level metadata for the specified granule. A granule can be specified either by EMD dbID or Geoid. BMGT reads Metadata from SDSRV Database.
generate list of grans metadata product	BMGT generates granule level metadata for the specified granules. A list of EMD dbIDs or Geoids is made available to BMGT. BMGT reads Metadata from SDSRV Database.
publish Data Pool gran ftp url	BMGT/BulkURL generates a XML product containing Data Pool FTP URLs for the qualified granules. The product is delivered to ECHO via FTP push or pull. The product conforms to ECHO's update API. Information required to formulate FTP URL is read from Data Pool Database.
delete published urls	BMGT/BulkURL, upon invocation by Data Pool Cleanup utility, generates a XML product containing Data Pool granule URs that are targeted for removal from ECHO. The product is delivered to ECHO via FTP push or pull. ECHO will use this product to remove just the Data Pool related data like FTP, wms and wcs URLs. The product conforms to ECHO's update API.
validate collection metadata	BMGT validates its generated collection metadata (in xml format) with corresponding ODL Metadata file.
validate granule metadata	BMGT validates its generated granule metadata (in xml format) with corresponding ODL Metadata file.
GetUrlInfo	BMGT reads necessary info from Data Pool database to support 'publish' events mentioned above.
publish deletes	Data Pool Cleanup utility invokes BMGT/BulkURL to convey Data Pool granule deletes to ECHO. See 'delete published URLs' event above.
get 'to be deleted' granules	BMGT/BulkURL reads from Data Pool Database information pertaining to 'to be deleted granules'. Data Pool Cleanup utility persists this information before invoking BMGT/BulkURL.
ingest metadata products	BMGT generates ODL MET files and PDR files for all its Metadata products to facilitate 'Polling with Delivery Record Ingest'. This is part of SIPS interface.
Getmetadata	BMGT reads collection, granule, browse, valids, and QA related metadata from SDSRV Database.
distribute bulk browse	DSS distributes browse images if OMS is in SYN 3 mode, otherwise OMS distributes the browse images to ECHO via SIPS interface. This is not BMGT's responsibility. Included here to show how this event is triggered if an ECSBBR product is acquired.
distribute bulk metadata	EMD distributes BMGT Metadata products to ECHO via SIPS interface.

4.15.2.7 BulkURL CSCI Process Interface Description

Table 4.15-7 provides descriptions of the interface events shown in the BMGT context diagram.

Table 4.15-7. BulkURL CSCI Process Interface Events

Event	Event Frequency	Interface	Initiated By	Event Description
export Data Pool ftp url info	Once per service request	<i>Process:</i> EcBmBulkURL <i>Collaborator:</i> ProcOSGetDplURL	<i>Process:</i> Cron or operator	<p>The BulkURL utility executed with 'insert' option will export an xml representation of the FTP URL information for science files, metadata files, and browse files associated with Data Pool granules. After successful generation of xml product, the utility will push the product to ECHO via FTP. If it fails then the product will moved to a different folder and an email message is sent to ECHO for FTP pull.</p> <p>Only granules that meet a set of conditions are exported. The granule selection criteria is encapsulated in ProcOSGetDplURL stored proc.</p>
export delete info	Once per service request	<i>Process:</i> EcBmBulkURL <i>Collaborator:</i> ProcOSGetDplDeletedURL	<i>Process:</i> Cron or operator	<p>The Data Pool Cleanup Utility invokes BulkURL utility with 'delete' option. It will export an xml representation of granule UR and its FTP, wms and wcs URL information that needs to be removed from ECHO. After successful generation of xml product, the utility will push the product to ECHO via FTP. If it fails then the product will moved to a different folder and an email message is sent to ECHO for FTP pull.</p> <p>Only granules that meet a set of conditions are exported. The granule selection criteria is encapsulated in ProcOSGetDplDeletedURL stored procs.</p> <p>BulkURL writes Delete information to a flat file to recover from a failure.</p>

4.15.2.8 Data Stores

BulkURL uses Data Pool DB to generate its products. Table 4.15-8 provides the Data Store description.

Table 4.15-8. CSCI Data Stores

Data Store	Type	Description
Data Pool DB	Sybase	BulkURL reads required metadata from Data Pool DB.

4.15.2.9 BMGT DTD Schemas

4.15.2.9.1 BMGTCollectionMetadata.dtd

```
<!ELEMENT CollectionMetaDataFile (DTDVersion, DataCenterId, TemporalCoverage, DefaultPackage, CollectionMetaData*)>
```

```
<!-- Version identifier of the DTD used to generate the file -->
<!ELEMENT DTDVersion (#PCDATA)>
```

```
<!-- DataCenterId of the site that stores this metadata (e.g., EDC) -->
<!ELEMENT DataCenterId (#PCDATA)>
```

```
<!-- the start and end dates of this MetaDataFile (YYYY-MM-DD) -->
<!ELEMENT TemporalCoverage (StartDate, EndDate)>
<!ELEMENT StartDate (#PCDATA)>
<!ELEMENT EndDate (#PCDATA)>
```

```
<!-- Default Packaging Information will apply to every data collection unless over
written in the collection-level metadata -->
<!ELEMENT DefaultPackage (MediaTypes+, ProductionOptions, EstimatedCost?)>
<!ELEMENT EstimatedCost (#PCDATA)>
```

```
<!ELEMENT MediaTypes (MediaType, MediaFormats+)>
<!ELEMENT MediaType (#PCDATA)>
```

```
<!ELEMENT MediaFormats (MediaFormat, MediaParameters*)>
<!ELEMENT MediaFormat (#PCDATA)>
```

```
<!ELEMENT MediaParameters (ParameterName?, Specialized?, Obscured?, Type?,
Mandatory?, MaxLen?, Label?, MediaValid?)>
<!ELEMENT ParameterName (#PCDATA)>
<!ELEMENT Specialized (#PCDATA)>
<!ELEMENT Obscured (#PCDATA)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT Mandatory (#PCDATA)>
<!ELEMENT MaxLen (#PCDATA)>
<!ELEMENT Label (#PCDATA)>
```

```
<!ELEMENT MediaValid (MediaValid)+>
<!ELEMENT MediaValid (#PCDATA)>
```

```
<!ELEMENT ProductionOptions (ProductionHistoryOptionName?,
AncillaryDataOptionName?, NativeGranuleOptionName?)>
<!ELEMENT ProductionHistoryOptionName (#PCDATA)>
<!ELEMENT AncillaryDataOptionName (#PCDATA)>
<!ELEMENT NativeGranuleOptionName (#PCDATA)>
```

```

<!ELEMENT CollectionMetaData (ShortName, VersionID, InsertTime, LastUpdate?,
LongName, CollectionDescription, RevisionDate?, SuggestedUsagel?,
SuggestedUsage2?, ProcessingCenter?, ProcessingLevelId?,
ProcessingLevelDescription?, ArchiveCenter, VersionDescription,
CitationforExternalPublication?, CollectionState?, MaintenanceandUpdateFrequency?,
AccessConstraints?, CollectionPackage?, Spatial?, Temporal?, Contact*,
DisciplineTopicParameters*, Platform*, StorageMedium*, AdditionalAttributes*,
BrowseProduct?, SpatialKeyword*, TemporalKeyword*, CSDDescription*, Locality*,
CollReview*, Documents?, CollectionAssociation*, AnalysisSource*,
Campaign*,AssociatedDIFs?)>
<!ELEMENT ShortName (#PCDATA)>
<!ELEMENT VersionID (#PCDATA)>
<!ELEMENT InsertTime (#PCDATA)>
<!ELEMENT LastUpdate (#PCDATA)>
<!ELEMENT LongName (#PCDATA)>
<!ELEMENT CollectionDescription (#PCDATA)>
<!ELEMENT RevisionDate (#PCDATA)>
<!ELEMENT SuggestedUsagel (#PCDATA)>
<!ELEMENT SuggestedUsage2 (#PCDATA)>
<!ELEMENT ProcessingCenter (#PCDATA)>
<!ELEMENT ProcessingLevelId (#PCDATA)>
<!ELEMENT ProcessingLevelDescription (#PCDATA)>
<!ELEMENT ArchiveCenter (#PCDATA)>
<!ELEMENT VersionDescription (#PCDATA)>
<!ELEMENT CitationforExternalPublication (#PCDATA)>
<!ELEMENT CollectionState (#PCDATA)>
<!ELEMENT MaintenanceandUpdateFrequency (#PCDATA)>
<!ELEMENT AccessConstraints (#PCDATA)>
<!ELEMENT StorageMedium (#PCDATA)>
<!ELEMENT SpatialKeyword (#PCDATA)>
<!ELEMENT TemporalKeyword (#PCDATA)>

<!ELEMENT CollectionPackage (MediaTypes+, ProductionOptions, EstimatedCost?)>

<!ELEMENT Spatial (SpatialCoverageType, HorizontalSpatialDomain?,
VerticalSpatialDomain*, CoordinateSystemContainer?, OrbitParameters?,
GranuleSpatialRepresentation?)>
<!ELEMENT SpatialCoverageType (#PCDATA)>

<!ELEMENT HorizontalSpatialDomain ((ZoneIdentifier?,Geometry) | Global)>
<!ELEMENT ZoneIdentifier (#PCDATA)>
<!ELEMENT Geometry (CoordinateSystem?, (Point | Circle | BoundingRectangle |
GPolygon))>

<!ELEMENT CoordinateSystem (Geodetic | Cartesian)>
<!ELEMENT Geodetic EMPTY>
<!ELEMENT Cartesian EMPTY>
<!ELEMENT Global EMPTY>

<!ELEMENT Point (PointLongitude, PointLatitude)>
<!ELEMENT PointLongitude (#PCDATA)>
<!ELEMENT PointLatitude (#PCDATA)>

<!ELEMENT Circle (CenterLatitude, CenterLongitude, Radius, RadiusUnits)>
<!ELEMENT CenterLatitude (#PCDATA)>
<!ELEMENT CenterLongitude (#PCDATA)>
<!ELEMENT Radius (#PCDATA)>
<!ELEMENT RadiusUnits (#PCDATA)>

<!ELEMENT BoundingRectangle (WestBoundingCoordinate, NorthBoundingCoordinate,
EastBoundingCoordinate, SouthBoundingCoordinate)>

```

```

<!ELEMENT WestBoundingCoordinate (#PCDATA)>
<!ELEMENT NorthBoundingCoordinate (#PCDATA)>
<!ELEMENT EastBoundingCoordinate (#PCDATA)>
<!ELEMENT SouthBoundingCoordinate (#PCDATA)>

<!ELEMENT GPolygon (Boundary+)>
<!ELEMENT Boundary (Point, Point, Point, Point*)>

<!ELEMENT VerticalSpatialDomain (VerticalSpatialDomainType,
VerticalSpatialDomainValue)>
<!ELEMENT VerticalSpatialDomainType (#PCDATA)>
<!ELEMENT VerticalSpatialDomainValue (#PCDATA)>

<!ELEMENT CoordinateSystemContainer (VerticalCoordinateSystemContainer?,
HorizontalCoordinateSystemContainer?)>
<!ELEMENT VerticalCoordinateSystemContainer (AltitudeSystemDefinition?,
DepthSystemDefinition?)>

<!ELEMENT AltitudeSystemDefinition (AltitudeDatumName, AltitudeDistanceUnits,
AltitudeEncodingMethod, AltitudeResolution)>
<!ELEMENT AltitudeDatumName (#PCDATA)>
<!ELEMENT AltitudeDistanceUnits (#PCDATA)>
<!ELEMENT AltitudeEncodingMethod (#PCDATA)>
<!ELEMENT AltitudeResolution (#PCDATA)>

<!ELEMENT DepthSystemDefinition (DepthDatumName, DepthDistanceUnits,
DepthEncodingMethod, DepthResolution)>
<!ELEMENT DepthDatumName (#PCDATA)>
<!ELEMENT DepthDistanceUnits (#PCDATA)>
<!ELEMENT DepthEncodingMethod (#PCDATA)>
<!ELEMENT DepthResolution (#PCDATA)>

<!ELEMENT HorizontalCoordinateSystemContainer (GeodeticModel?,
(GeographicCoordinateSystem | PlanarCoordinateSystems | LocalCoordinateSystem))>

<!ELEMENT GeodeticModel (HorizontalDatumName?, EllipsoidName, SemiMajorAxis,
DenominatorofFlatteningRatio)>
<!ELEMENT HorizontalDatumName (#PCDATA)>
<!ELEMENT EllipsoidName (#PCDATA)>
<!ELEMENT SemiMajorAxis (#PCDATA)>
<!ELEMENT DenominatorofFlatteningRatio (#PCDATA)>

<!ELEMENT GeographicCoordinateSystem (LatitudeResolution, LongitudeResolution,
GeographicCoordinateUnits)>
<!ELEMENT LatitudeResolution (#PCDATA)>
<!ELEMENT LongitudeResolution (#PCDATA)>
<!ELEMENT GeographicCoordinateUnits (#PCDATA)>

<!ELEMENT PlanarCoordinateSystems (PlanarCoordinateSystem)>

<!ELEMENT PlanarCoordinateSystem (PlanarCoordinateSystemContainer+)>

<!ELEMENT PlanarCoordinateSystemContainer (PlanarCoordinateInformation,
(MapProjection | LocalPlanarCoordinateSystem | GridCoordinateSystem))>

<!ELEMENT PlanarCoordinateInformation (PlanarDistanceUnits,
PlanarCoordinateEncodingMethod, (DistanceandBearingRepresentation |
CoordinateRepresentation))>
<!ELEMENT PlanarDistanceUnits (#PCDATA)>
<!ELEMENT PlanarCoordinateEncodingMethod (#PCDATA)>

```

```

<!ELEMENT DistanceandBearingRepresentation (DistanceResolution, BearingResolution,
BearingUnits, BearingReferenceDirection, BearingReferenceMeridian)>
<!ELEMENT DistanceResolution (#PCDATA)>
<!ELEMENT BearingResolution (#PCDATA)>
<!ELEMENT BearingUnits (#PCDATA)>
<!ELEMENT BearingReferenceDirection (#PCDATA)>
<!ELEMENT BearingReferenceMeridian (#PCDATA)>

<!ELEMENT CoordinateRepresentation (AbscissaResolution, OrdinateResolution)>
<!ELEMENT AbscissaResolution (#PCDATA)>
<!ELEMENT OrdinateResolution (#PCDATA)>

<!ELEMENT MapProjection (MapProjectionName, MapProjectionPointer?)>
<!ELEMENT MapProjectionName (#PCDATA)>
<!ELEMENT MapProjectionPointer (#PCDATA)>

<!ELEMENT LocalPlanarCoordinateSystem (LocalPlanarCoordinateSystemDescription,
LocalPlanarGeoreferenceInformation)>
<!ELEMENT LocalPlanarCoordinateSystemDescription (#PCDATA)>
<!ELEMENT LocalPlanarGeoreferenceInformation (#PCDATA)>

<!ELEMENT GridCoordinateSystem (GridCoordinateSystemName)>
<!ELEMENT GridCoordinateSystemName (#PCDATA)>

<!ELEMENT LocalCoordinateSystem (LocalCoordinateSystemDescription,
LocalGeoreferenceInformation)>
<!ELEMENT LocalCoordinateSystemDescription (#PCDATA)>
<!ELEMENT LocalGeoreferenceInformation (#PCDATA)>

<!ELEMENT OrbitParameters (SwathWidth, Period, InclinationAngle)>
<!ELEMENT SwathWidth (#PCDATA)>
<!ELEMENT Period (#PCDATA)>
<!ELEMENT InclinationAngle (#PCDATA)>

<!ELEMENT GranuleSpatialRepresentation (Cartesian | Geodetic | Orbit | NoSpatial)>
<!ELEMENT Orbit EMPTY>
<!ELEMENT NoSpatial EMPTY>

<!ELEMENT Temporal (TimeType, DateType, TemporalRangeType, PrecisionofSeconds,
EndsatPresentFlag, (RangeDateTime | SingleDateTime+))>
<!ELEMENT TimeType (#PCDATA)>
<!ELEMENT DateType (#PCDATA)>
<!ELEMENT TemporalRangeType (#PCDATA)>
<!ELEMENT PrecisionofSeconds (#PCDATA)>
<!ELEMENT EndsatPresentFlag (#PCDATA)>

<!ELEMENT RangeDateTime (RangeBeginningDate, RangeBeginningTime, RangeEndingDate?,
RangeEndingTime?)>
<!ELEMENT RangeBeginningDate (#PCDATA)>
<!ELEMENT RangeBeginningTime (#PCDATA)>
<!ELEMENT RangeEndingDate (#PCDATA)>
<!ELEMENT RangeEndingTime (#PCDATA)>

<!ELEMENT SingleDateTime (CalendarDate, TimeOfDay)>
<!ELEMENT CalendarDate (#PCDATA)>
<!ELEMENT TimeOfDay (#PCDATA)>

<!ELEMENT Contact (ContactRole, HoursOfService?, ContactInstructions?,
(Organization | ContactPerson), Address?, Email?, Telephone*, Fax*, ContactURL?)>
<!ELEMENT ContactRole (#PCDATA)>
<!ELEMENT HoursOfService (#PCDATA)>

```

```

<!ELEMENT ContactInstructions (#PCDATA)>

<!ELEMENT Organization (OrganizationName)>
<!ELEMENT OrganizationName (#PCDATA)>

<!ELEMENT ContactPerson (FirstName, MiddleName?, LastName, JobPosition?)>
<!ELEMENT FirstName (#PCDATA)>
<!ELEMENT MiddleName (#PCDATA)>
<!ELEMENT LastName (#PCDATA)>
<!ELEMENT JobPosition (#PCDATA)>

<!ELEMENT Address (StreetAddress, City, StateProvince, PostalCode, Country)>
<!ELEMENT StreetAddress (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT StateProvince (#PCDATA)>
<!ELEMENT PostalCode (#PCDATA)>
<!ELEMENT Country (#PCDATA)>

<!ELEMENT Email (#PCDATA)>
<!ELEMENT Telephone (#PCDATA)>
<!ELEMENT Fax (#PCDATA)>
<!ELEMENT ContactURL (#PCDATA)>

<!ELEMENT DisciplineTopicParameters (DisciplineKeyword, TopicKeyword, TermKeyword,
VariableKeyword?, ECSParameterKeyword*)>
<!ELEMENT DisciplineKeyword (#PCDATA)>
<!ELEMENT TopicKeyword (#PCDATA)>
<!ELEMENT TermKeyword (#PCDATA)>
<!ELEMENT VariableKeyword (#PCDATA)>
<!ELEMENT ECSParameterKeyword (#PCDATA)>

<!ELEMENT Platform (PlatformShortName, PlatformLongName, PlatformType,
PlatformCharacteristic*, Instrument*)>
<!ELEMENT PlatformShortName (#PCDATA)>
<!ELEMENT PlatformLongName (#PCDATA)>
<!ELEMENT PlatformType (#PCDATA)>

<!ELEMENT PlatformCharacteristic (PlatformCharacteristicName,
PlatformCharacteristicDescription, PlatformCharacteristicDataType,
PlatformCharacteristicUnit?, PlatformCharacteristicValue)>
<!ELEMENT PlatformCharacteristicName (#PCDATA)>
<!ELEMENT PlatformCharacteristicDescription (#PCDATA)>
<!ELEMENT PlatformCharacteristicDataType (#PCDATA)>
<!ELEMENT PlatformCharacteristicUnit (#PCDATA)>
<!ELEMENT PlatformCharacteristicValue (#PCDATA)>

<!ELEMENT Instrument (InstrumentShortName, InstrumentLongName?,
InstrumentTechnique?, NumberOfSensors?, InstrumentCharacteristic*, Sensor*,
OperationMode*)>
<!ELEMENT InstrumentShortName (#PCDATA)>
<!ELEMENT InstrumentLongName (#PCDATA)>
<!ELEMENT InstrumentTechnique (#PCDATA)>
<!ELEMENT NumberOfSensors (#PCDATA)>

<!ELEMENT InstrumentCharacteristic (InstrumentCharacteristicName,
InstrumentCharacteristicDescription, InstrumentCharacteristicDataType,
InstrumentCharacteristicUnit?, InstrumentCharacteristicValue)>
<!ELEMENT InstrumentCharacteristicName (#PCDATA)>
<!ELEMENT InstrumentCharacteristicDescription (#PCDATA)>
<!ELEMENT InstrumentCharacteristicDataType (#PCDATA)>
<!ELEMENT InstrumentCharacteristicUnit (#PCDATA)>

```

```

<!ELEMENT InstrumentCharacteristicValue (#PCDATA)>

<!ELEMENT Sensor (SensorShortName, SensorLongName?, SensorTechnique?,
SensorCharacteristic*)>
<!ELEMENT SensorShortName (#PCDATA)>
<!ELEMENT SensorLongName (#PCDATA)>
<!ELEMENT SensorTechnique (#PCDATA)>

<!ELEMENT SensorCharacteristic (SensorCharacteristicName,
SensorCharacteristicDescription, SensorCharacteristicDataType,
SensorCharacteristicUnit?, SensorCharacteristicValue)>
<!ELEMENT SensorCharacteristicName (#PCDATA)>
<!ELEMENT SensorCharacteristicDescription (#PCDATA)>
<!ELEMENT SensorCharacteristicDataType (#PCDATA)>
<!ELEMENT SensorCharacteristicUnit (#PCDATA)>
<!ELEMENT SensorCharacteristicValue (#PCDATA)>
<!ELEMENT OperationMode (#PCDATA)>

<!ELEMENT AdditionalAttributes (AdditionalAttributeDataType,
AdditionalAttributeDescription, AdditionalAttributeName, MeasurementResolution?,
ParameterRangeBegin?, ParameterRangeEnd?, ParameterUnitsOfMeasure?,
ParameterValueAccuracy?, ValueAccuracyExplanation?, ParameterValue*)>
<!ELEMENT AdditionalAttributeDataType (#PCDATA)>
<!ELEMENT AdditionalAttributeDescription (#PCDATA)>
<!ELEMENT AdditionalAttributeName (#PCDATA)>
<!ELEMENT MeasurementResolution (#PCDATA)>
<!ELEMENT ParameterRangeBegin (#PCDATA)>
<!ELEMENT ParameterRangeEnd (#PCDATA)>
<!ELEMENT ParameterUnitsOfMeasure (#PCDATA)>
<!ELEMENT ParameterValueAccuracy (#PCDATA)>
<!ELEMENT ValueAccuracyExplanation (#PCDATA)>
<!ELEMENT ParameterValue (#PCDATA)>

<!-- List of browse granules that are related to this collection -->
<!ELEMENT BrowseProduct (BrowseGranuleId*)>
<!ELEMENT BrowseGranuleId (#PCDATA)>

<!ELEMENT CSDTDescription (PrimaryCSDT, Implementation?, CSDTComments?,
IndirectReference?)>
<!ELEMENT PrimaryCSDT (#PCDATA)>
<!ELEMENT Implementation (#PCDATA)>
<!ELEMENT CSDTComments (#PCDATA)>
<!ELEMENT IndirectReference (#PCDATA)>

<!ELEMENT Locality (LocalityType, LocalityDescription?)>
<!ELEMENT LocalityType (#PCDATA)>
<!ELEMENT LocalityDescription (#PCDATA)>

<!ELEMENT CollReview (ScienceReviewDate, ScienceReviewStatus, FutureReviewDate?)>
<!ELEMENT ScienceReviewDate (#PCDATA)>
<!ELEMENT ScienceReviewStatus (#PCDATA)>
<!ELEMENT FutureReviewDate (#PCDATA)>

<!ELEMENT Documents (Document+)>
<!ELEMENT Document (DocumentType?, DocumentURL?, DocumentURLComment?)>
<!ELEMENT DocumentType (#PCDATA)>
<!ELEMENT DocumentURL (#PCDATA)>
<!ELEMENT DocumentURLComment (#PCDATA)>

<!ELEMENT CollectionAssociation (AssociatedShortName, AssociatedVersionId,
CollectionType, CollectionUse1?, CollectionUse2?)>
<!ELEMENT AssociatedShortName (#PCDATA)>

```

```

<!ELEMENT AssociatedVersionId (#PCDATA)>
<!ELEMENT CollectionType (#PCDATA)>
<!ELEMENT CollectionUse1 (#PCDATA)>
<!ELEMENT CollectionUse2 (#PCDATA)>

<!ELEMENT AnalysisSource (AnalysisType, AnalysisShortName, AnalysisLongName?,
AnalysisTechnique?)>
<!ELEMENT AnalysisType (#PCDATA)>
<!ELEMENT AnalysisShortName (#PCDATA)>
<!ELEMENT AnalysisLongName (#PCDATA)>
<!ELEMENT AnalysisTechnique (#PCDATA)>

<!ELEMENT Campaign (CampaignShortName, CampaignLongName?, CampaignStartDate?,
CampaignEndDate?)>
<!ELEMENT CampaignShortName (#PCDATA)>
<!ELEMENT CampaignLongName (#PCDATA)>
<!ELEMENT CampaignStartDate (#PCDATA)>
<!ELEMENT CampaignEndDate (#PCDATA)>

<!ELEMENT AssociatedDIFs (DIF+)>
<!ELEMENT DIF (EntryID)>
<!ELEMENT EntryID (#PCDATA)>

```

4.15.2.9.2 **BMGTGranuleMetadata.dtd**

```

<!ELEMENT GranuleMetaDataFile (DTDVersion, DataCenterId, TemporalCoverage,
GranuleURMetaData*)>

<!-- Version identifier of the DTD used to generate the file -->
<!ELEMENT DTDVersion (#PCDATA)>

<!-- DataCenterId of the site that stores this metadata (e.g., EDC) -->
<!ELEMENT DataCenterId (#PCDATA)>

<!-- the start and end dates of this MetaDataFile (YYYY-MM-DD) -->
<!ELEMENT TemporalCoverage (StartDate, EndDate)>
<!ELEMENT StartDate (#PCDATA)>
<!ELEMENT EndDate (#PCDATA)>

<!ELEMENT GranuleURMetaData
(GranuleUR, DbID?, InsertTime?, LastUpdate?, DeleteTime?, CollectionMetaData?,
ECSDataGranule?, PGEVersionClass?, (RangeDateTime | SingleDateTime)?,
SpatialDomainContainer?, OrbitCalculatedSpatialDomain?, MeasuredParameter?,
ProcessingQA?, StorageMediumClass?, Review?, Platform*, AnalysisSource*,
Campaign*, PSAs?, InputGranule?, BrowseProduct?, PHProduct?, QAProduct?,
AlgorithmPackage*, AncillaryInputGranules?)>
<!ELEMENT GranuleUR (#PCDATA)>
<!ELEMENT DbID (#PCDATA)>
<!ELEMENT InsertTime (#PCDATA)>
<!ELEMENT LastUpdate (#PCDATA)>
<!ELEMENT DeleteTime (#PCDATA)>

<!ELEMENT CollectionMetaData (ShortName, VersionID)>
<!ELEMENT ShortName (#PCDATA)>
<!ELEMENT VersionID (#PCDATA)>

<!ELEMENT ECSDataGranule
(SizeMBECSDataGranule, ReprocessingPlanned?, ReprocessingActual?, LocalGranuleID?,
DayNightFlag?, ProductionDateTime, LocalVersionID?)>
<!ELEMENT SizeMBECSDataGranule (#PCDATA)>
<!ELEMENT ReprocessingPlanned (#PCDATA)>

```

```

<!ELEMENT ReprocessingActual (#PCDATA)>
<!ELEMENT LocalGranuleID (#PCDATA)>
<!ELEMENT DayNightFlag (#PCDATA)>
<!ELEMENT ProductionDateTime (#PCDATA)>
<!ELEMENT LocalVersionID (#PCDATA)>

<!ELEMENT PGEVersionClass (PGEVersion)>
<!ELEMENT PGEVersion (#PCDATA)>

<!ELEMENT RangeDateTime
(RangeEndingTime, RangeEndingDate, RangeBeginningTime, RangeBeginningDate)>
<!ELEMENT RangeEndingTime (#PCDATA)>
<!ELEMENT RangeEndingDate (#PCDATA)>
<!ELEMENT RangeBeginningTime (#PCDATA)>
<!ELEMENT RangeBeginningDate (#PCDATA)>

<!ELEMENT SingleDateTime (TimeOfDay, CalendarDate)>
<!ELEMENT TimeOfDay (#PCDATA)>
<!ELEMENT CalendarDate (#PCDATA)>

<!ELEMENT SpatialDomainContainer
(GranuleLocality*, VerticalSpatialDomain*, HorizontalSpatialDomainContainer?)>

<!ELEMENT GranuleLocality (LocalityValue)>
<!ELEMENT LocalityValue (#PCDATA)>

<!ELEMENT VerticalSpatialDomain (VerticalSpatialDomainContainer)>

<!ELEMENT VerticalSpatialDomainContainer
(VerticalSpatialDomainType, VerticalSpatialDomainValue)>
<!ELEMENT VerticalSpatialDomainType (#PCDATA)>
<!ELEMENT VerticalSpatialDomainValue (#PCDATA)>

<!ELEMENT HorizontalSpatialDomainContainer
(ZoneIdentifierClass?, (Point | Circle | BoundingBoxRectangle | GPolygon | Global))>

<!ELEMENT ZoneIdentifierClass (ZoneIdentifier)>
<!ELEMENT ZoneIdentifier (#PCDATA)>

<!ELEMENT Point (PointLongitude, PointLatitude)>
<!ELEMENT PointLongitude (#PCDATA)>
<!ELEMENT PointLatitude (#PCDATA)>

<!ELEMENT Circle (CenterLatitude, CenterLongitude, Radius, RadiusUnits)>
<!ELEMENT CenterLatitude (#PCDATA)>
<!ELEMENT CenterLongitude (#PCDATA)>
<!ELEMENT Radius (#PCDATA)>
<!ELEMENT RadiusUnits (#PCDATA)>

<!ELEMENT BoundingBoxRectangle
(WestBoundingCoordinate, NorthBoundingCoordinate, EastBoundingCoordinate,
SouthBoundingCoordinate)>
<!ELEMENT WestBoundingCoordinate (#PCDATA)>
<!ELEMENT NorthBoundingCoordinate (#PCDATA)>
<!ELEMENT EastBoundingCoordinate (#PCDATA)>
<!ELEMENT SouthBoundingCoordinate (#PCDATA)>

<!ELEMENT GPolygon (Boundary)+>
<!ELEMENT Boundary (Point, Point, Point, Point*)>

<!ELEMENT Global EMPTY>

```

```

<!ELEMENT OrbitCalculatedSpatialDomain (OrbitCalculatedSpatialDomainContainer)+>

<!ELEMENT OrbitCalculatedSpatialDomainContainer
(OrbitalModelName?, OrbitNumber?, OrbitRange?, EquatorCrossingLongitude,
EquatorCrossingDate, EquatorCrossingTime)>

<!ELEMENT OrbitalModelName (#PCDATA)>
<!ELEMENT OrbitNumber (#PCDATA)>

<!ELEMENT OrbitRange (StartOrbitNumber, StopOrbitNumber)>
<!ELEMENT StartOrbitNumber (#PCDATA)>
<!ELEMENT StopOrbitNumber (#PCDATA)>

<!ELEMENT EquatorCrossingLongitude (#PCDATA)>
<!ELEMENT EquatorCrossingDate (#PCDATA)>
<!ELEMENT EquatorCrossingTime (#PCDATA)>

<!ELEMENT MeasuredParameter (MeasuredParameterContainer)+>
<!ELEMENT MeasuredParameterContainer (ParameterName, QAStats?, QAFlags?)>
<!ELEMENT ParameterName (#PCDATA)>

<!ELEMENT QAStats
(QAPercentMissingData, QAPercentOutOfBoundsData?, QAPercentInterpolatedData?,
QAPercentCloudCover?)>
<!ELEMENT QAPercentMissingData (#PCDATA)>
<!ELEMENT QAPercentOutOfBoundsData (#PCDATA)>
<!ELEMENT QAPercentInterpolatedData (#PCDATA)>
<!ELEMENT QAPercentCloudCover (#PCDATA)>

<!ELEMENT QAFlags
(AutomaticQualityFlag?, AutomaticQualityFlagExplanation?, OperationalQualityFlag?,
OperationalQualityFlagExplanation?, ScienceQualityFlag?,
ScienceQualityFlagExplanation?)>
<!ELEMENT AutomaticQualityFlag (#PCDATA)>
<!ELEMENT AutomaticQualityFlagExplanation (#PCDATA)>
<!ELEMENT OperationalQualityFlag (#PCDATA)>
<!ELEMENT OperationalQualityFlagExplanation (#PCDATA)>
<!ELEMENT ScienceQualityFlag (#PCDATA)>
<!ELEMENT ScienceQualityFlagExplanation (#PCDATA)>

<!ELEMENT ProcessingQA (ProcessingQAContainer)+>

<!ELEMENT ProcessingQAContainer (ProcessingQADescription, ProcessingQAAttribute)>
<!ELEMENT ProcessingQADescription (#PCDATA)>
<!ELEMENT ProcessingQAAttribute (#PCDATA)>

<!ELEMENT StorageMediumClass (StorageMedium)+>
<!ELEMENT StorageMedium (#PCDATA)>

<!ELEMENT Review (ReviewContainer)+>
<!ELEMENT ReviewContainer (ScienceReviewStatus, ScienceReviewDate,
FutureReviewDate?)>
<!ELEMENT ScienceReviewStatus (#PCDATA)>
<!ELEMENT ScienceReviewDate (#PCDATA)>
<!ELEMENT FutureReviewDate (#PCDATA)>

<!ELEMENT Platform (PlatformShortName, Instrument*)>
<!ELEMENT PlatformShortName (#PCDATA)>

<!ELEMENT Instrument (InstrumentShortName, Sensor*, OperationMode*)>
<!ELEMENT InstrumentShortName (#PCDATA)>
<!ELEMENT OperationMode (#PCDATA)>

```

```

<!ELEMENT Sensor (SensorShortName, SensorCharacteristic*)>
<!ELEMENT SensorShortName (#PCDATA)>

<!ELEMENT SensorCharacteristic (SensorCharacteristicName,
SensorCharacteristicValue)>
<!ELEMENT SensorCharacteristicName (#PCDATA)>
<!ELEMENT SensorCharacteristicValue (#PCDATA)>

<!ELEMENT AnalysisSource (AnalysisShortName)>
<!ELEMENT AnalysisShortName (#PCDATA)>

<!ELEMENT Campaign (CampaignShortName)>
<!ELEMENT CampaignShortName (#PCDATA)>

<!ELEMENT PSAs (PSA+)>
<!ELEMENT PSA (PSAName, PSAValue+)>
<!ELEMENT PSAName (#PCDATA)>
<!ELEMENT PSAValue (#PCDATA)>

<!ELEMENT InputGranule (InputPointer+)>
<!ELEMENT InputPointer (#PCDATA)>

<!-- List of browse granules that are related to this granule -->
<!ELEMENT BrowseProduct (BrowseGranuleId+)>
<!ELEMENT BrowseGranuleId (#PCDATA)>

<!-- List of production history granules that are related to this granule -->
<!ELEMENT PHProduct (PHGranuleId+)>
<!ELEMENT PHGranuleId (#PCDATA)>

<!-- List of QA granules that are related to this granule -->
<!ELEMENT QAProduct (QAGranuleId+)>
<!ELEMENT QAGranuleId (#PCDATA)>

<!ELEMENT AlgorithmPackage (AlgorithmPackageName, AlgorithmPackageVersion,
AlgorithmPackageMaturityCode, AlgorithmPackageAcceptDate, DeliveryPurpose,
PGName, PGEVersion, PGEIdentifier, PGEFunction, PGEDateLastModified, SWVersion,
SWDateLastModified, SSAPComponent*)>
<!ELEMENT AlgorithmPackageName (#PCDATA)>
<!ELEMENT AlgorithmPackageVersion (#PCDATA)>
<!ELEMENT AlgorithmPackageMaturityCode (#PCDATA)>
<!ELEMENT AlgorithmPackageAcceptDate (#PCDATA)>
<!ELEMENT DeliveryPurpose (#PCDATA)>
<!ELEMENT PGName (#PCDATA)>
<!ELEMENT PGEIdentifier (#PCDATA)>
<!ELEMENT PGEFunction (#PCDATA)>
<!ELEMENT PGEDateLastModified (#PCDATA)>
<!ELEMENT SWVersion (#PCDATA)>
<!ELEMENT SWDateLastModified (#PCDATA)>

<!ELEMENT SSAPComponent (ComponentType, ComponentName, SSAPAlgorithmPackageName,
SSAPInsertDate)>
<!ELEMENT ComponentType (#PCDATA)>
<!ELEMENT ComponentName (#PCDATA)>
<!ELEMENT SSAPAlgorithmPackageName (#PCDATA)>
<!ELEMENT SSAPInsertDate (#PCDATA)>

<!ELEMENT AncillaryInputGranules (AncillaryInputGranule+)>
<!ELEMENT AncillaryInputGranule (AncillaryInputType, AncillaryInputPointer)>
<!ELEMENT AncillaryInputType (#PCDATA)>

```

```
<!ELEMENT AncillaryInputPointer (#PCDATA)>
```

4.15.2.9.3 BMGTBrowseMetadata.dtd

```
<!ELEMENT BrowseReferenceFile (DTDVersion, DataCenterId, TemporalCoverage, BrowseCrossReference*)>
```

```
<!-- Version identifier of the DTD used to generate the file -->  
<!ELEMENT DTDVersion (#PCDATA)>
```

```
<!-- DataCenterId of the site that stores this metadata (e.g., LP DAAC-EMD) -->  
<!ELEMENT DataCenterId (#PCDATA)>
```

```
<!-- the start and end dates of this MetaDataFile (YYYYDDD) -->  
<!ELEMENT TemporalCoverage (StartDate, EndDate)>  
<!ELEMENT StartDate (#PCDATA)>  
<!ELEMENT EndDate (#PCDATA)>
```

```
<!ELEMENT BrowseCrossReference (GranuleUR, BrowseGranuleId?, InsertTime?, LastUpdate?, DeleteTime?, InternalFileName, BrowseDescription?, BrowseSize?)>  
<!ELEMENT GranuleUR (#PCDATA)>  
<!ELEMENT BrowseGranuleId (#PCDATA)>  
<!ELEMENT InsertTime (#PCDATA)>  
<!ELEMENT LastUpdate (#PCDATA)>  
<!ELEMENT DeleteTime (#PCDATA)>  
<!ELEMENT InternalFileName (#PCDATA)>  
<!ELEMENT BrowseDescription (#PCDATA)>  
<!ELEMENT BrowseSize (#PCDATA)>
```

4.15.2.9.4 BMGTValidMetadata.dtd

```
<!ELEMENT ValidFile (DTDVersion, DataCenterId, TemporalCoverage, DictionaryAttribute+, KeywordValid+)>
```

```
<!-- Version identifier of the DTD used to generate the file -->  
<!ELEMENT DTDVersion (#PCDATA)>
```

```
<!-- DataCenterId of the site that stores the metadata (e.g. LP DAAC-EMD) -->  
<!ELEMENT DataCenterId (#PCDATA)>
```

```
<!-- The start and end dates of this MetaDataFile (YYYYDDD) -->  
<!ELEMENT TemporalCoverage (StartDate, EndDate)>  
<!ELEMENT StartDate (#PCDATA)>  
<!ELEMENT EndDate (#PCDATA)>
```

```
<!-- Attributes and their Data Types -->  
<!ELEMENT DictionaryAttribute (QualifiedAttrName, Type, Length, RuleText*)>  
<!ELEMENT QualifiedAttrName (#PCDATA)>  
<!ELEMENT Type (#PCDATA)>  
<!ELEMENT Length (#PCDATA)>  
<!ELEMENT RuleText (#PCDATA)>
```

```
<!-- Keyword Attributes and their Domain Values -->  
<!ELEMENT KeywordValid (DisciplineKeyword, TopicKeyword, TermKeyword, VariableKeyword?, ParameterKeyword?)>  
<!ELEMENT DisciplineKeyword (#PCDATA)>  
<!ELEMENT TopicKeyword (#PCDATA)>  
<!ELEMENT TermKeyword (#PCDATA)>  
<!ELEMENT VariableKeyword (#PCDATA)>  
<!ELEMENT ParameterKeyword (#PCDATA)>
```

4.15.2.9.5 BMGTUpdateMetadata.dtd

```
<!ELEMENT ProviderAccountService (UpdateMetadata)>
<!--UpdateMetadata can update a single collection, multiple collections, a single
granule, or multiple granules in one transaction. Each update allows the addition
of new metadata-->
<!ELEMENT UpdateMetadata (Collection*, Granule*)>
<!ELEMENT Collection (Target+, (Add | Update | Delete)+)>
<!ELEMENT Granule (Target+, (Add | Update | Delete)+)>
<!-- Target+ allows the same change to be made to several different granules or
collections simultaneously. This is especially useful for bulk deletions of
OnlineURLs. -->
<!ELEMENT Target (ID, ProviderLastUpdateDateTime, SaveDateTimeFlag?)>
<!-- SaveDateTimeFlag is the flag that allows echo to update the last update date
time for Target. The default is SAVE -->
<!ELEMENT Add (QualifiedTag, MetadataValue)>
<!ELEMENT Update (QualifiedTag, MetadataValue)>
<!ELEMENT Delete (QualifiedTag+)>
<!ELEMENT QualifiedTag (#PCDATA)>
<!ELEMENT MetadataValue (#PCDATA)>
<!ELEMENT ProviderLastUpdateDateTime (#PCDATA)>
<!ELEMENT SaveDateTimeFlag (SAVE | DONTSAVE)>
<!ELEMENT SAVE EMPTY>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT DONTSAVE EMPTY>
```

4.16 OGC-ECHO Adaptor (OEA) Subsystem Overview

The OGC-ECHO Adaptor (OEA) provides a mechanism to allow the Earth Science Gateway Portal and other OGC clients to perform OGC catalogue searches against ECHO holdings. This is presented in an OGC-compliant fashion, which is accomplished via servicing Z39.50 GEO profile catalog requests from the Earth Science Gateway Portal.

The OEA complies with version 2 of the OGC Catalog Services Specification (http://portal.opengis.org/files/?artifact_id=5929&version=1) using the Z39.50 binding. This specification defines the following mandatory requests.

Housekeeping Requests:

- Initiate Session
- Terminate Session
- Status
- Cancel

Catalog requests will require interaction with the ECHO Application Programming Interface. These will be rendered into ECHO client API requests by the OEA. The results of those ECHO API requests will be rendered into the counterpart ESG Portal catalog results. Those requests being,

- Search
- Present

Consequently, the OEA will comply with version 6.0 of the ECHO API (<http://api.echo.eos.nasa.gov/echo/About.jsp>)

OGC-ECHO Adaptor (OEA) Context

Figure 4.16-1 describes the interaction between the ESG Portal, OEA and ECHO API.

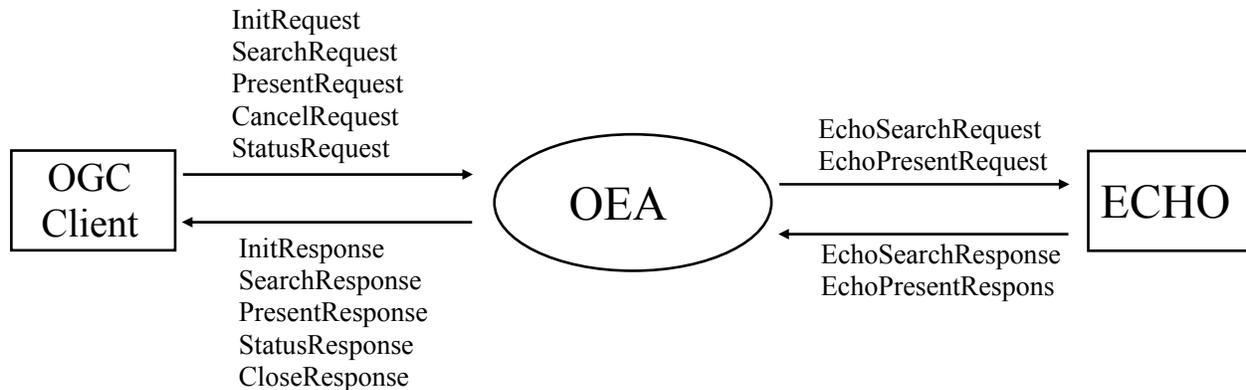


Figure 4.16-1. OEA Context Diagram

Note that the OEA will be subject to interrogation of two distinct types,

- On demand searches from individual users. Individual search and present operations for distinct inventory.
- Harvesting operations from NSDI (National Spatial Data Infrastructure) clients. Broad search and present operations with the intent of replicating all or part of the NDSI node’s data holdings. Harvesting operations are subject to repetition in order to keep the client’s inventory copy up to date.

The ESG, for example, can operate in both modes.

Table 4.16-1 provides descriptions of the interface events shown in the OEA context diagram.

Table 4.16-1. OEA Interface Events (1 of 2)

Event	Interface Event Description
InitRequest	OEA receives Init Request from Client to create a new session.
InitResponse	OEA creates a new session and sends back a valid init response to notify client that a session has been generated.
SearchRequest	OEA receives a Search Request from Client; it could involve granule, collection, granule and collection, temporal and so on.
EchosearchRequest	OEA translates the Search to XML format and submits it to ECHO on behave of a user.
CloseRequest	OEA receives Close Request from Client to terminate a connection.
CloseResponse	OEA cleans up a session, send back a Close Response and terminates the connection.
PresentRequest	OEA receives a Present Request from Client with some resultSetName.

Table 4.16-1. OEA Interface Events (2 of 2)

Event	Interface Event Description
EchoPresentRequest	OEA translates the Present to XML format and submits it to ECHO on behalf of a user.
StatusRequest	OEA receives a Status Request during a Search/Present.
StatusResponse	OEA sends back a Status Response only if the Search/Present request is not in a terminal state (Canceled, Failed, Done).
CancelRequest	OEA receives Cancel Request during a Search/Present to terminate the request.
CancelResponse	OEA stops the Search/Present and sends back a Cancel response only if the Search/Present is not in a terminal state (Canceled, Failed, Done).

4.16.1 OGC-ECHO Adaptor Architecture

The OEA is functionally divided into three packages -- *z3950serverfacade package*, *translation package*, and *Echoclientfacade package*. The reason for this decoupling of functionality allows reuse of each element in possible future adaptor development. Certainly, Z39.50 GEO is not the only protocol compliant with the OGC Catalogue Service Specification (http and xml may become prevalent in the future).

- **z3950serverfacade package**

The z3950serverfacade is responsible for session management between OEA and OGC clients. It also validates and parses all incoming z3950 requests (Init, Search, Present, Status, Cancel and Close), calls the translation package to translate all the Search/Present requests into XML formats, and sends them to the echoclientfacade package for processing. After receiving responses from ECHO, it calls the translation package again to translate the responses into z3950 Search/Present responses and eventually sends them back to the client. For any other type of requests (Init, Status, Cancel, Close), the z3950serverfacade also generates responses and sends them back to the client.

- **translation package**

All requests that are submitted to OEA from ESG are in z3950 format, however, ECHO handles all its requests and responses in XML format. This is when the translation package comes in. Equipped with style sheets for both Search and Present, the translation package handles the crucial translation of requests and responses between z3950 and XML formats. It ensures no loss of information between the requests that come in from ESG and the requests that are sent to ECHO and vice versa.

- **echoclientfacade package**

The Echoclientfacade handles session management between OEA and ECHO by creating a session with a valid username and password per ECHO request. It also manages all Echo request and response objects, sends and receives requests to and from ECHO.

Figure 4.16-2 describes the OEA processes to handle a request and Table 4.16-2 provides descriptions of the events in Figure 4.16-2.

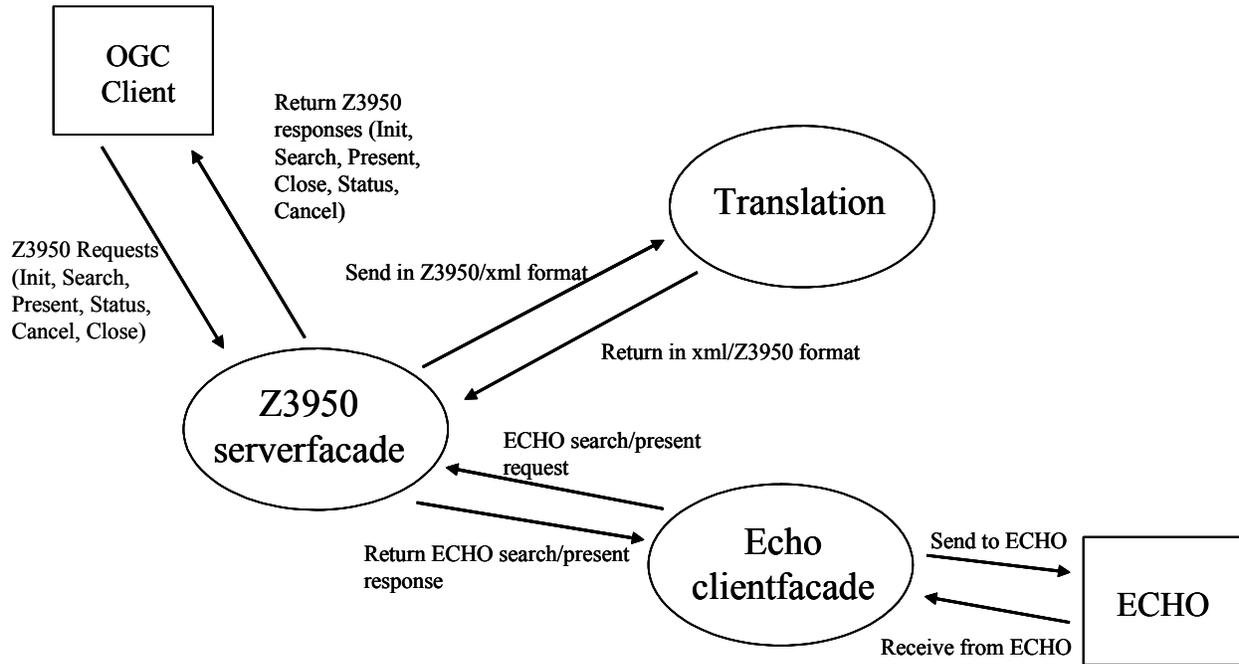


Figure 4.16-2. OEA Architecture

4.16.1.1 OEA Process Interface Descriptions

Table 4.16-2 provides descriptions of the interface processes shown in the OEA architecture.

Table 4.16-2. OEA Processes Interface Events (1 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
z3950Requests (Init, Search, Present, Status, Cancel, Close)	One per request	Process: EcOwOea Library: Z3950serverfacade Class: Server Session InitService CloseService TRCService SearchService PresentService	Process: OGCClient	The OEA receives requests from Clients to do one of the following: Initiate a Session Search Echo Inventory Present a resultSet Request status of a particular Search/Present Cancel a Search/Present Close a Session OEA then validates and parses the request

Table 4.16-2. OEA Processes Interface Events (2 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Send in z3950/xml format	One call to translator per echo request	<i>Process:</i> EcOwOea <i>Library:</i> translation <i>Class:</i> ServiceTranslator SearchTranslator PresentTranslator XMLGenerator	<i>Process:</i> EcOwOea <i>Library:</i> Z3950serverfacade <i>Class:</i> SearchService PresentService	For any Search/Present request, OEA calls the translator to translate z3950 requests into XML format, so they later can be sent to ECHO. When OEA receives request from ECHO, it will also call the translator to translate the responses back to z3950 responses.
Return in xml/z3950format	One z3950response per request	<i>Process:</i> EcOwOea <i>Library:</i> Z3950Serverfacade <i>Class:</i> SearchService PresentService	<i>Process:</i> EcOwOea <i>Library:</i> translation <i>Class:</i> SearchTranslator PresentTranslator	After applying style sheets, OEA passes the newly formatted request/response back to z3950serverfacade package to get ready for ECHO
Echo Search/present Request	One per z3950 request	<i>Process:</i> EcOwOea <i>Library:</i> echoclientfacade <i>Class:</i> ECHOSearchRequest ECHOPresentRequest Session SearchService PresentService	<i>Process:</i> EcOwOea <i>Library:</i> Z3950serverfacade <i>Class:</i> SearchService PresentService	After translation, OEA creates the ECHO Search/Present object, and establishes a session to ECHO per request
Send to ECHO	One connection per request	<i>Process:</i> ECHO	<i>Process:</i> EcOwOea <i>Library:</i> echoclientfacade <i>Class:</i> Session SessionPool	OEA sends the ECHO Search/Present request to ECHO and awaits responses.

Table 4.16-2. OEA Processes Interface Events (3 of 3)

Event	Event Frequency	Interface	Initiated By	Event Description
Receive from ECHO	One ECHO response per request	<i>Process:</i> EcOwOea <i>Library:</i> echoclientfacade <i>Class:</i> Session SessionPool	<i>Process:</i> ECHO	ECHO returns the Search/Present result
Return Echo Search/Present Response	One per Echo response	<i>Process:</i> EcOwOea <i>Library:</i> Z3950serverfacade <i>Class:</i> SearchService PresentService	<i>Process:</i> EcOwOea <i>Library:</i> echoclientfacade <i>Class:</i> ECHOSearchRequest ECHOPresentRequest	Echoclientfacade returns the ECHO Search/Present responses back to z3950serverfacade .Z3950serverfacade then calls translation to translate the responses back into z3950 responses.
Z3950Response (Init, Search, Present, Status, Cancel, Close)	One per Z3950 request	<i>Process:</i> OGC Clients	<i>Process:</i> EcOwOea <i>Library:</i> Z3950serverfacade <i>Class:</i> Server Session InitService CloseService TRCService SearchService PresentService	The OEA finally send z3950 responses back to the Client (Note for the requests not going thru ECHO: Init, Close, Status and Cancel, z3950serverfacade will construct the response accordingly right after validation and parsing of the incoming request)

4.16.1.2 Use of COTS in the OEA

- **Java 1.4.2**

The Java 1.4.2 class libraries are used by the OEA to provide basic functions and objects such as strings and collections. The Java libraries must be installed with the OEA software for any of the OEA processes to run.

- **Jzkit**

The Jzkit class libraries are a pure java toolset for building advanced search and retrieve applications. It assists in implementing Z3950 standard in pure java environment. The Jzkit libraries must be installed with OEA software for any interactions between OGC and OEA.

- **Jdom**

The Jdom class libraries enables the use of XSL style sheets to translate any Z3950 requests to XML formatted ECHO requests and vice versa. It has to be installed to allow proper translation between OEA and ECHO.

- **Echo Client Toolkit**

Allows OEA to query ECHO's metadata clearinghouse for data sets and granules by using a set of Java classes that interact with the ECHO system. It has to be installed to allow proper communication between OEA and ECHO.

- **Apache Soap**

Apache SOAP ("Simple Object Access Protocol") is an implementation of the SOAP submission to W3C. It is based on, and supersedes, the IBM SOAP4J implementation. All requests are submitted to ECHO through a SOAP mechanism. It has to be installed to allow proper communication between OEA and ECHO.

4.17 Open Geospatial Consortium (OGC) Web Services (OWS) Functional Overview (REMOVED)

5. Limitations of Current Implementation

5.1 Data Server Subsystem

Science Data Server (SDSRV) CSCI

- **Operator GUI:** There is no support for re-installation of ESDTs from the GUI. Reinstallation of ESDTs is supported through a command line Unix Shell script interface. Corresponding data must be manually deleted from the SDSRV, Spatial Subscription Server and Data Dictionary.
- Metadata update services only support QA metadata.
- There is no persistence of Client requests, except for asynchronous acquire requests. Requests must be re-submitted.
- There is no support for Access Control List checking.

5.2 Planning Subsystem (REMOVED)

5.3 Infrastructure Subsystem

Machine-to-Machine Gateway CSC

- The machine-to-machine gateway relies on ssh for the authentication of a connection.
- Each request must be associated with a MSS User ID. The operator has the choice of configuring a default MSS User ID for each MTMGW server. If there is no default for a server and the user does not provide a User ID in the request, the server rejects requests that do not specify a User ID.

5.4 MSS Subsystem

MCI CSCI – Security CSC

- If an unauthorized user gains access to a host despite security measures, it is not detected until the next detection interval expires, since this is only checked periodically. (If the detection intervals were decreased, the system uses too much of its processing power for monitoring itself.)
- The security implementation requires the operator to perform security tasks such as running Crack manually.
- The configuration files for TCP Wrappers can become difficult to manage if multiple versions exist. If administrators setup a “back door,” the system security can be more easily compromised.

MCI CSCI – Accountability CSC

- There is no retry in place for Database updates or inserts and errors are logged as low priority.

MCI CSCI – Trouble Ticketing CSC

- Trouble Tickets forwarded from a DAAC to the SMC are set to a forwarded state in the DAAC. A manual process is necessary to receive notification that the Trouble Ticket has been closed at the SMC and is now closed at the DAAC.
- For an overall status of Trouble Tickets in the EMD system, reports must be run at each DAAC and forwarded via e-mail where they can be consolidated.

5.5 Product Distribution System (PDS) Subsystem (REMOVED)

5.6 Spatial Subscription Server (SSS) Subsystem

- Subscriptions can be entered only for future granules. It is not possible to order existing granules via subscription.
- Non-spatial qualification of a subscription is limited to five numeric/date qualifiers and fifty string qualifiers.
- Spatial qualification is limited to the specification of the coordinates of an overlapping latitude-longitude box.
- It is not possible to request Data Pool insertion for a granule when entering a subscription via the command line interface. The subscription must be entered via the GUI if Data Pool insertion is desired.

5.7 Open Geospatial Consortium (OGC) Web Services (OWS) Subsystem (REMOVED)

5.8 Order Manager Server (OMS) Subsystem

No limitations.

5.9 Data Management (DMS) Subsystem

No limitations.

5.10 Data Pool Ingest (DPLINGEST) Subsystem

- PAN / PDRD notifications will be retried indefinitely if they encounter a failure. There could be a scenario where a notification will not succeed. The operator will need to find a way to remedy the situation. Most likely the issue is that the PAN / PDRD file has been deleted and so the operator may need to regenerate the PAN / PDRD file manually.

Abbreviations and Acronyms

A

ABC++	Document Generator used to provide class level detail
ACMHW	Access and Control Management Hardware (Configuration Item)
AD	Advertisement
ADC	Affiliated Data Center (National Oceanic and Atmospheric Administration only)
AGS	ASTER Ground System
AIT	Algorithm Integration and Test
AITHW	Algorithm Integration and Test Hardware (Configuration Item)
AI&T	Algorithm Integration and Test
AITTL	Algorithm Integration and Test Tools (Computer Software Configuration Item)
AM-1	See TERRA (spacecraft)
AOI	Area of Interest
AOS	ASTER Operations Segment
AP	Algorithm Package
APC	Access/Process Coordinators
API	Application Program Interface
AQA	Algorithm Quality Assurance
AQUA	PM-1 Satellite (AIRS, AMSR-E, AMSU, CERES, HSB, MODIS)
AR	Action Request
AS	Administration Stations
ASCII	American Standard Code for Information Interchange
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
ATM	Asynchronous Transfer Mode
AURA	NASA mission to study the earth's ozone, air quality and climate (formerly the CHEM mission)

B

BCP	Bulk Copy Program
	Bulk Copy Procedure
BDS	Bulk Data Server
BLM	Baseline Manager

C

CAD	Computer Aided Design
CCB	Change Control Board (Raytheon Convention)
	Configuration Control Board (NASA Convention)
CCDI	ClearCase DDTS Integration
CCR	Configuration Change Request
CDE	Common Desktop Environment
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CD-ROM	Compact Disk - Read Only Memory
CDS	Cell Directory Service
CFG	Configuration File
CGI	Common Gateway Interface
CHUI	Character-based User Interface
CI	Configuration Item
CLS	Client Subsystem
CM	Configuration Management
CMI	Cryptographic Management Interface
CMP	Configuration Management Plan
CN	Change Notice
CO	Contracting Officer
COTS	Commercial Off the Shelf (Software or Hardware)
CPF	Calibration Parameter File
CPU	Central Processing Unit
CRM	Change Request Manager

CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSMS	Communications and Systems Management Segment (ECS)
CSS	Communications Subsystem
<u>D</u>	
DAAC	Distributed Active Archive Center
DADS	Data Archive and Distribution System
DAO	Data Assimilation Office
DAP	Delivered Algorithm Package
DAS	Dual Attached Station
DB	Database
DBMS	Database Management System
DCCI	Distributed Computing Configuration Item
DCN	Document Change Notice
DDICT	Data Dictionary (Computer Software Configuration Item)
DDR	Detailed Design Review
	Data Delivery Record (same as a Product Delivery Record)
DDT	DAAC Distribution Technician
DDTS	Distributed Defect Tracking System (COTS)
DEM	Digital Elevation Model
DESKT	Desktop (Computer Software Configuration Item)
DEV	Custom Developed Code
DFS	Distributed File System
DID	Data Item Description
DIPHW	Distribution and Ingest Peripheral Hardware Configuration Item
DLL	Dynamic Link Library
DLT	Digital Linear Tape
DM	Data Management
DMGHW	Data Management Hardware (Configuration Item)
DMS	Data Management Subsystem

DNS	Domain Name Service
DOF	Distributed Object Framework
DORRAN	Distributed Ordering, Researching, Reporting, and Accounting Network (At EDC)
DP	Data Provider
DPL	Data Pool Subsystem
DPR	Data Processing Request
DPRID	Data Processing Request Identifier
DPREP	Data Pre-Processing
DR	Data Repository
DRPHW	Data Repository Hardware (Configuration Item)
DSC	Development Solution for the C programming language
DSS	Data Server Subsystem
DTF	Sony DTF Tape cartridge system (replacement for the D3 tape cartridge system)
DTS	Distributed Time Service

E

ECHO	ECS Clearing House
ECN	Engineering Change Notice
ECS	Earth Observing System Data and Information Core System
EDC	Earth Resource Observation System (EROS) Data Center
EDF	ECS Development Facility
EDG	EOS Data Gateway
EDHS	ECS Data Handling System
EDN	Expedited Data Set Notification
EDOS	Earth Observing System Data and Operations System
EDR	Expedited Data Set Request
EDS	Expedited Data Set
EC	Error conditions (in tickets)
EGS	EOS Ground System
EISA	Enhanced Industry Standard Architecture

E-mail	Electronic Mail (also Email, e-mail, and email)
EMOS	ECS Mission Operations Segment (formerly FOS)
EMSn	EOSDIS Mission Support network
EOC	Earth Observing System Operations Center
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
EROS	Earth Resource Observation System
ESDIS	Earth Science Data and Information System (GSFC Code 505)
ESDT	Earth Science Data Type
ESRI	Environmental Systems Research Institute
ETM+	Enhanced Thematic Mapper Plus (Landsat 7)
<u>F</u>	
FC	Functional components (capabilities in tickets)
FCAPS	Fault, Configuration, Accountability, Performance, and Security services
FDS	Flight Dynamics System
FH	Fault Handling
FLDB	Fileset Location Database
F&PRS	Functional and Performance Requirements Specification
FSMS	File and Storage Management System
FTP	File Transfer Protocol
FTPD	File Transfer Protocol Daemon
<u>G</u>	
GB	gigabyte (10^9)
Gb	gigabit (10^9)
GCDIS	Global Change Data and Information System
GCMD	Global Change Master Directory (not developed by ECS)
GFE	Government Furnished Equipment
GLAS	Geoscience Laser Altimeter System
GSFC	GODDARD Space Flight Center (NASA facility and DAAC)
GSMS	Ground System Management Subsystem (ASTER)

GTWAY (Version 0 Interoperability/ASTER) Gateway (Computer Software Configuration Item)

GUI Graphical User Interface

H

HDF Hierarchical Data Format

HDF-EOS an EOS proposed standard for a specialized HDF data format

HMI Human Machine Interface

HSB Humidity Sounder for Brazil

HTML HyperText Markup Language

HTTP HyperText Transport Protocol

HWCI Hardware Configuration Item

I

IAS Image Assessment System

ICESat Ice, Cloud and Land Elevation Satellite

ICD Interface Control Document

ID User Identification (or Identifier)

IDG Infrastructure Development Group

IDL Interactive Data Language

I/F Interface

IGS International Ground Station (Landsat 7)

IHCI Internetworking hardware configuration item

ILG Infrastructure Library Group

ILM Inventory, Logistics, Maintenance (ILM) Manager

IMS Information Management System (ECS element name)

INHCI Internetworking HWCI

I/O Input/Output

IP Internet Protocol
International Partner

IRD Interface Requirements Document

IRR Incremental Release Review

ISIPS ICESat Science Investigator-Led Processing System

ISO International Standards Organization

ISS Internetworking Subsystem

I&T Integration and Test

J

JESS Java Earth Science Server

JEST Java Earth Science Tool

JPL Jet Propulsion Laboratory (DAAC)

K

KFTP Kerberos File Transfer Protocol

L

L0 - L4 Level-0 through Level-4 data (ECS)

L0R Landsat Reformatted Data

LAMS Landsat 7 Archive Management System

LAN Local Area Network

LaRC Langley Research Center (DAAC)

LFS Local File System

LZ77 Lampel-Ziv coding

M

M&O Maintenance and Operations

MB Megabyte (10^6)

Mbps Megabits Per Second

MCF Metadata Configuration File

MCI Management Software Configuration Item (Computer Software Configuration Item)

MSSHW (System) Management (Subsystem) Hardware Configuration Item

MISR Multi-Imaging SpectroRadiometer

MLCI Management Logistics Configuration Item (Computer Software Configuration Item)

MM	Mode Management
MMO	Mission Management Office
MMS	Mode Management Service
M&O	Maintenance and Operations (Staff)
MOC	Mission Operations Center
MOPITT	Measurements of Pollution in the Troposphere
MP	Message Passing
MS	Mass Storage
MSCD	Mirror Scan Correction Data (file)
MSS	System Management Subsystem
MTA	LAMS Metadata File
MTMGW	Machine-to-Machine Gateway
MTP	Distribution Product Metadata File Extension (<filename>.MTP)
MTPE	Mission to Planet Earth
<u>N</u>	
NASA	National Aeronautics and Space Administration
NCEP	National Centers for Environmental Predictions
NCR	Non-conformance Report
NESDIS	National Environmental Satellite, Data, and Information Service (NOAA)
Netscape	Browser (for user registration and search engine) [Netscape Communicator] Mail component for e-mail transfers
NFS	Network File System
NIS	Network Information Service
NMC	National Meteorological Center (located at National Oceanic and Atmospheric Administration - NOAA)
NNTP	Network News Transfer Protocol
NOAA	National Oceanic and Atmospheric Administration
NSI	National Aeronautics and Space Administration Science Internet
NSIDC	National Snow and Ice Data Center (DAAC)

Q

ODL	Object Description Language
OEA	OGC-ECHO Adaptor
OEM	Original Equipment Manufacturer
OMS	Order Manager Subsystem
OMSRV	Order Manager Server
OMGUI	Order Manager Graphical User Interface
OOA	Object oriented analysis
OOD	Object oriented design
OODCE	Object Oriented distributed computing environment
OPS CON	Operations Concept
OS	Operating System
OSI	Open Systems Interconnect

P

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

Q

QA	Quality Assurance
QDS	Quick Look Data Set (Same as Expedited Data Set)

R

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

S

SAGE III	Stratospheric Aerosol and Gas Experiment III
SAS	Single Attached Station
SATAN	Security Administrator Tool for Analyzing Networks
SCF	Science Computing Facility
SCLI	SDSRV Command Line Interface
SCSI	Small Computer System Interface
SDE	Software Development Environment
SDF	Software Development Folder
SDP	Science Data Processing
SDPTK	Science Data Processing Toolkit Science Data Processing Toolkit (Computer Software Configuration Item)
SDSRV	Science Data SeRVer (Computer Software Configuration Item)
SGI	Silicon Graphics, Inc.
SIM	Spectral Irradiance Monitor

SIPS	Science Investigator-Led Processing Systems
SMC	System Management Center System Monitoring and Coordination Center
SMP	Symmetric Multi-processor
SMTP	Simple Mail Transport Protocol
SOLSTICE	Solar Stellar Irradiance Comparison Experiment
SORCE	Solar Radiation and Climate Experiment
SPARC	Single Processor Architecture
SPRHW	Science Processing Hardware (Configuration Item)
SQL	Structured Query Language
SQS	Spatial Query Server
SRF	Server Request Framework
SSAP	Science Software Archive Package
SSIT	Science Software Integration and Test
SSS	Spatial Subscription Server Subsystem
STK	StorageTek
Sybase	(ECS) COTS database management product (ASE)
SYSLOG	System Log

T

TAR	Tape Archive
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TELNET	Telecommunications Network
TERRA	EOS AM Mission spacecraft 1, morning equator crossing spacecraft series -- ASTER, MISR, MODIS and MOPITT instruments; Mission to study the land, oceans and the earth's radiation budget
TIM	Total Irradiance Monitor
TM	Thematic Mapper (Landsat)
TT	Trouble Ticket
TTPro	TestTrack Pro

U

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

V

V0	Version Zero (Version 0)
V0 GTWAY	Version Zero Gateway (V0 GTWAY CSCI)
V0 ODL	Version 0 Object Description Language
VT	Virtual terminal

W

WAN	Wide Area Network
WKBCH	WorKBenCH (Computer Software Configuration Item)
WKSHW	Working Storage Hardware Configuration Item
WRS	Worldwide Reference System
WS	Working Storage
WWW	World Wide Web

X

xAR	x Acquisition Request (where x is any kind of or generic acquisition request)
XBDS	Bulk Data Service Protocol
XDR	External Data Representation
XFS	Extended File System