

6. Release A Cross Subsystem Design Concepts

The design of the ECS applications is based on a number of common software architecture principles. They ensure that (1) ECS applications will be able to interoperate, (2) ECS applications are easier to maintain, support and operate; and (3) software components can be reused across subsystems, thus reducing the implementation effort and risk.

This section elaborates five of these architectural concepts. They are:

- Distributed Communications Architecture,
- Security Architecture,
- External Interface Architecture,
- Systems Management Architecture, and
- User Interface Architecture.

Within the ECS architecture, responsibility for design and development components is allocated in an unambiguous manner. Aspects of these designs, therefore, will appear in the various detailed design documents of the subsystems which have the responsibility for their implementation. However, the concepts are presented in this overview because they cross subsystem boundaries and their interconnections and rationales are lost when viewed solely within the context of a specific ECS subsystem.

6.1 Distributed Communications Architecture

6.1.1 Overview

The primary goal of the distributed communication architecture is to provide a software infrastructure for current and future ECS development. This infrastructure must meet the science and technology requirements of Release A, and provide a clear evolutionary path to the system capabilities, technologies, and capacities required for Release B and beyond. In order to achieve this goal, the Release A design was influenced by a number of design drivers. Foremost among them are the following:

- Extensability - During the lifetime of ECS the system must accommodate growth in the areas of value added providers, new data types and services, and interoperability with other information systems. The distributed object architecture provides a uniform mechanism for referencing and accessing ECS data, objects, and services.
- Evolvability - Due to the rapid growth in processing hardware, storage devices, network communications, and information services the ECS must support a continuous evolution of it's components.
- Scalability - The ECS design must scale easily to the expected increase in users, data products, and processing requirements of Release B and beyond.

The distributed communications architecture provides the framework for meeting the needs of the design drivers. Following the client/server architectural model, the distributed communications architecture allows client and server applications to cooperate without direct knowledge of each other. Service requests are made by invoking methods of a local "proxy" object within the client's address space. The proxy object, in turn, communicates with the server application to perform the service request. This communication may occur across address spaces and platforms, and is transparent to the application developer and the user.

This mechanism provides a layer of isolation between the client application and the server implementation such that modifications to the server implementation can proceed without changing the client implementation, so long as legacy methods provided by the server continue to be supported. This allows ECS to add new servers, upgrade processing or storage hardware, and to add new services without requiring modifications to clients of the server. The Release A distributed communications architecture is comprised of the following key components:

- Distributed Client/Server Communications — ECS will use DCE, via the Object-Oriented DCE (OODCE) COTS product for Release A. The architecture allows for a managed transition to other technologies (e.g., CORBA) in subsequent releases.
- Distributed Object Framework (DOF) — The DOF consists of a set of core services (interfaces and objects) that provide basic functionality to simplify the development of distributed applications. Operations provided by the DOF are expected to serve as building blocks for ECS client and server development. The DOF design is described in Section 6.1.2.
- Universal References (UR) — UR provide a system wide mechanism for referencing ECS data and service objects. The UR design encapsulates the details of locating and accessing a server or data object. This design insulates the client applications from knowledge of the location of the object and the communications protocols used to access the object. The UR design is described in Section 6.1.3.

6.1.2 Distributed Object Framework

Object-Oriented applications are assembled from a number of interrelated objects. Each object is characterized by a set of attributes and methods. Each object has a clear interface that identifies the methods a user can invoke. The object that requests information is called the requester and the object that provides a service is called the provider. Each provider object takes requests for operations that it has identified in the interface, performs the computations, and passes the results back to the requester. Object-oriented application development consists of defining and instantiating the objects and passing messages (invoking methods) between the objects to achieve its objective.

In single address space applications, all objects reside in the same address space. In the distributed object framework, objects are distributed in multiple address spaces, spanning heterogeneous platforms. Objects can reside anywhere in the network, but the basic contract between a provider object and requester object is the interface that the provider advertises for a requester to use. Objects can be spread across the network based on efficiency, availability of data, requirements for autonomy, etc. From the perspective of the requester of a service, the object location (location

independence) and invocation (invocation independence) should be the same no matter where the object physically resides. Invoking methods amounts to passing messages between objects. If the objects are in different address spaces, then the messaging is done via a network. The client/server paradigm supports this kind of communication. In this paradigm, one side of the session (client) is allowed to make requests, while the other side (server) may only make replies.

The distributed object framework consists of a set of core services with distinct functionality to make the development of the distributed applications easier. The core services are naming, security, threads, time, and rpc. DOF interacts with the Naming Service to save and retrieve service locations. Similarly it interacts with the Security Service to ensure security.

In order to aid the application programmer, another layer of abstraction is provided. This layer consists of four generic classes: Server, GenObject, Interface and InterfaceMgr. Application programmers implementing the client server application need to develop three parts: an application client part which invokes the service, an application server part that implements the service, and an application server main (driver) part which actually creates and runs the application server as a separate process and provides all the functionality needed at the server side.

The application client class inherits from the Interface class and the application server class inherits from the InterfaceMgr and the GenObject to use the default functionality provided in the parent classes. Alternately, they can modify the inherited methods to achieve the needed behavior. There is a global instance of the server class which the server driver users. These four generic classes, along with the Interface Definition Language (IDL), C and C++ (limited) language bindings to the IDL provide all the functionality for the application programmer to develop client/server applications.

The application program carries on all the interaction with the underlying core services like naming, security, threads, and time through the DOF for normal operations. Interfaces to these underlying core services are also provided.

6.1.3 Universal References (URs)

The ECS architecture contains a universal reference (UR) object which provides a means for identifying objects throughout ECS in a standardized manner. The primary purpose of the UR is to point to a data granule or to save a user session's context for actions such as subscriptions or queries. Since ECS is a distributed system of UNIX processes communicating using OODCE as the communication middleware, and since the persistent data tracked by a UR is located in any one of these processes, there is a strong correlation between UR's and the communication middleware's process identification mechanism. Therefore, the UR is a combination of network information and ECS application information.

In order to access the UNIX process (server object) and gain access to the persistent data associated with a UR, the UR must contain enough information to connect to that process. An entry into the address space is needed to create the OODCE connection. After connecting, a client of the ECS application data may create an instance of the actual object associated with the data. (e.g., in order to create an Earth Science Data Type (ESDT) object, one must use the UR to

connect to the correct Data Server object in that address space, and then perform whatever ESDT operations desired.).

In order to support the UR requirements, we derived the necessary capabilities for UR's in the ECS system:

- UR's identify the location of Data Granules and Service Providers.
- UR's can be used to connect to a Data Server.
- UR's have a finite/infinite lifetime.
- UR's map to a single data granule, service provider, subscription or query.
- The service provider can be derived from any UR.
- Working Collections can be created by inserting UR's.
- Data Granule UR's are created by and managed by the Data Server.
- UR's private ID mechanism is provided by CSS.
- The UR Model can be extended to Release B requirements (e.g., HTTP URL Relationships for Data and Documents).

These UR requirements result in two designs which mirror one another. There is an external ASCII based representation and an internal C++ based representation which allows the ECS system to easily manipulate UR's internally and allows clients the flexibility of viewing UR's outside the system, passing them to other clients in a form such as email, and re-introducing them into the ECS system to restore state.

The external UR is a set of ASCII strings in some well defined format. This format is defined/known by the internal ECS system. When a UR is extracted from the ECS system by a client, the UR is "externalized" into this string format and delivered to the client. An external UR has all the information necessary to pass back into the ECS system and restore all its state information.

The internal UR is a class hierarchy with various levels which specialize the UR to meet particular requirements for Data Granules, Service Providers and Session State storage. This is an extensible design which allows the UR to grow over time and be used to identify all flavors of persistent information. It leverages the capabilities of OODCE universal identifiers, since there is a great deal of similarity, but it does not couple the ECS architecture to OODCE. Future revisions of the ECS system may employ other universal identifier schemes to without damaging the design.

6.2 Security Architecture

6.2.1 Driving Requirements

ECS is part of an international science and research program accessible to users on a global basis. Built on an open-computing network architecture to facilitate access by a wide variety of users, the driving requirements of its security architecture are data integrity, availability, and confidentiality.

The project vision is open access to well-pedigreed data in which users may have absolute faith, while maintaining certain proprietary and administrative data in confidence.

Because of the scientific nature of the project, integrity of project information is paramount. Scientific formulae, streams of collected data, algorithms and modules used to process or measure, data files, archives, historical records control instructions and telemetry data all must be stored with absolute confidence in the integrity of the stored material. Users expect the system to guard against tampering with the source materials of the archive not only during storage, but during product generation, and distribution. Integrity threats are manifested through unauthorized access or use, leading to change, alteration or modification of information resources. The security architecture must provide adequate safeguards against these threats.

The ability to have assured use of project resources is vital to instill confidence in its users. Indeed, the users of ECS can be characterized as investors. Whether they are data providers (investing the fruits of their labor in the vision of an open, accessible archive) or data consumers (investing tax dollars, grant dollars, or corporate investment dollars), they have a right to expect availability. Availability is also important for more mundane, but just as vital reasons, for example, to satisfy timing and coordination requirements for scientific study and research experiments. Availability needs translate into two categories: fault tolerant features to preclude failure or operational disruption; and recovery actions, to enable timely resumption of operational activities and minimize the length of the disruption. Availability threats are manifested through denial of use actions, either physical in nature (e.g., explosion, flooding) or logical (e.g., computer virus or other intrusive software, or flooding, i.e., swamping a system component such as a gateway with such a large amount of messages or requests that other users find it difficult to get serviced). Because of the increasing complexity and interdependencies in large integrated systems, a failure in one component could have widespread physical or logical impact.

Confidentiality requirements exist because some of the information within ECS requires special protection. This includes operational or control information that is time-critical or used to control mission operations, specific user product requests and account information, and information of a private nature on individual users on the system. Information of this nature, if compromised, could result in damage or harm to ECS or to individuals. In addition, in some few instances archived data may be held as proprietary for a period of time, before it is made generally accessible. This may be for a short interval, while the data producer is exercising quality control testing over a new product, or of a longer duration, for example, provided to a data producer as an inducement to participate in the ECS system.

6.2.2 Information Security Strategy

Physical threats are countered by physical security measures that are, generally, implemented by physical barriers or operational procedures. The subject of this Security Architecture section of the design specification is logical security, i.e., measures taken by the computer system to protect itself against threats. Within the context of the International Standards Organization (ISO) Open Systems Interconnect (OSI) model, ECS security functions are applied at the Application Layer (layers 5 and above), supported by an infrastructure (layers 4 and below) providing network, transport, and interoperability services.

Security management services are provided enterprise-wide through the Management Services Subsystem (MSS). MSS applications provide services such as fault, performance, security, and accountability management. MSS manages both ECS's network resources, but also ECS's host and application resources. It also provides administrative support to the defensive infrastructure employed (e.g., DCE, application-layer gateways, network firewalls) by ECS to defend against a variety of logical threats. MSS allocates these management services to both the system-wide and local levels. Thus, with few exceptions, management services are fully decentralized, with no single point of failure that would preclude authorized user access, nor permit unauthorized user access.

The main threats to the ECS and their countermeasures are listed in Table 6.2-1. The table also indicates the ECS subsystem which is responsible for the implementation of the countermeasures. In general, system security measures are assigned to CSMS, since logical security attacks today usually exploit the network links that each system has with the outside. Moreover, by securing the communications within the system, it is often possible to isolate the remainder of the system from a successful attack on a particular system component.

However, communications components lack the application context that is sometimes needed to provide special protection. For example, CSMS can protect the access to a service which manages private data and thus relieve that service from having to implement extensive security measures itself. However, CSMS protection is incomplete, since CSMS will know only the destination but not the specific nature of an access request. In such cases, applications will provide additional access controls; these will typically rely on the user authentication performed by CSMS, but apply application-layer rules to determine authorization. They may also depend on CSMS ACL management as a repository for persistent, global, or remotely managed authorization rules.

6.2.3 OSF/DCE and OODCE Architecture

The core of information security within ECS will be based on the Open Software Foundation's Distributed Computing Environment (OSF/DCE). This technology is generally used in ECS to provide enterprise coordination among mixed computing platforms. Applications are layered atop a framework of DCE "middleware." In the present context the DCE middleware provides extra services in the area of data integrity, availability, and confidentiality. In particular, DCE provides user authentication, authorization down to the individual transaction level (for individuals or groups), and secure interoperability among applications.

Object-Oriented DCE (OODCE) provides a linkage between the object-oriented design approach selected by ECS and the DCE libraries provided by OSF. It may be viewed as an abstraction layer between the object-oriented application and the DCE middleware, making all of the DCE services (including naming, time, distributed file access, and threads, in addition to security) easily accessible to the application programmer.

Table 6.2-1. Threat/Countermeasures

Threat	Countermeasures	CSS	ISS	MSS	Other
Unauthorized use, privilege abuse	Authentication/ Authorization	X		X	
Unauthorized use of access controlled resources	Access Control	X		X	
Data tampering	Data Integrity	X	X		
Electronic 'eavesdropping'	Data Privacy	X			
Unauthorized use, cracker/hacker activity	Security Audits	X		X	
Unauthorized use, abuse, virus detection, denial of service, 'cracker' and hacker detection	Intrusion Detection	X		X	
Lack of secure environment due to non-compliance of security procedures	Compliance Management			X	
Crackers and hackers, viruses, broadcast storms, unauthorized use of resources	Routing Control Firewalls Address filtering		X		
Unauthorized access to private or proprietary data within a data server	Authentication/Authorization DBMS Security	X			DSS
Unauthorized alteration of processing schedules and plans	UNIX Access Controls Authentication/Authorization DBMS Security	X			PLS & DPS

DCE provides security features by employing a variation of the Kerberos model (version 5 or later) and POSIX 1003.6 Access Control lists (ACLs). Kerberos provides a strong protocol for authenticating users (proving a user is who he/she says he/she is), and thereafter provides trusted third-party services to applications, vouching to them for a previously authenticated user's identity. Secure authorization (verifying a user is authorized to perform the requested function, or access the selected data) is provided by authenticated remote procedure calls (RPCs), a form of function entitlement, and by application-embedded rules or application-invoked ACL managers, which verify the authorization of an authenticated user to any requested resource. The combination of authentication, authorization, and secure interoperability is the key security characteristic of the ECS DCE/OODCE architecture.

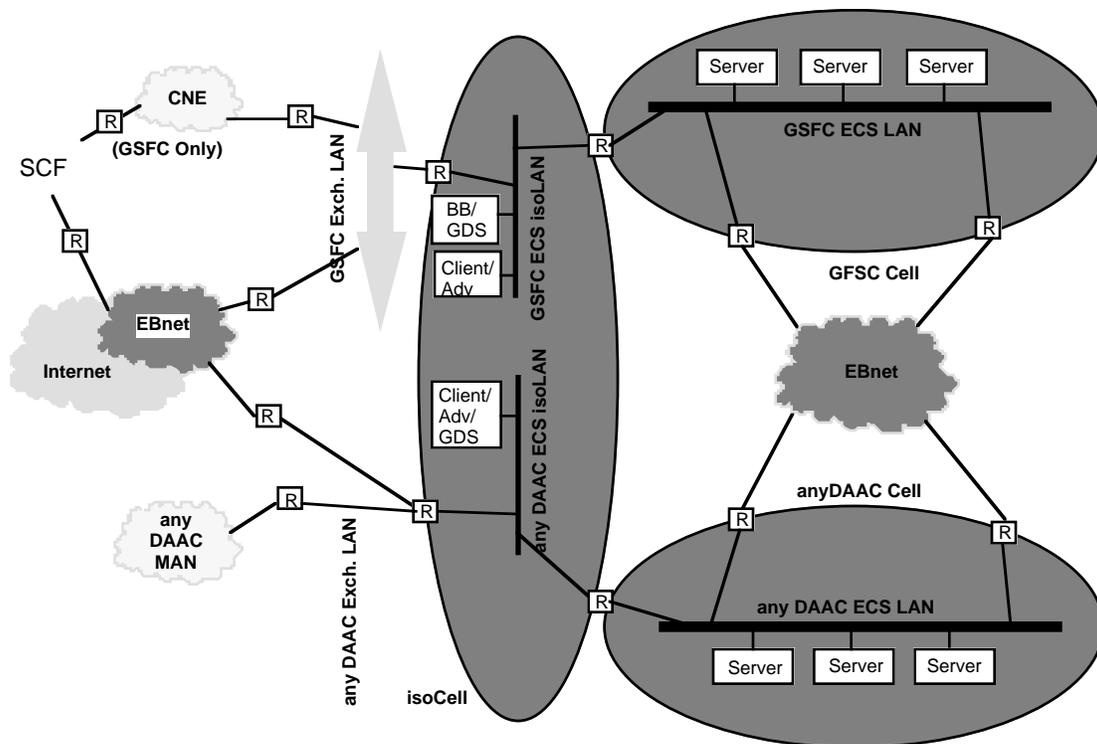


Figure 6.2-1. Conceptual DCE Cell-Based Security Architecture

Figure 6.2-1 illustrates the main concepts of the ECS security architecture. The DCE Cell Configuration trade (see 540-TP-001-001, Technical Paper: Communications and Systems Management Segment (CSMS) Preliminary Design Review Trade Studies for the ECS Project) identified multiple cell as a preferred design solution with regard to scalability and evolvability concerns. A cell around each DAAC set of resources is employed to prevent general public access into production environments. In addition, each DAAC requires at least one host with a low security, public access (a “gateway”) to encourage the use of ECS. One solution is the creation of an integration cell that 'connects' all public access points at each DAAC together and isolates the rest of the system from the security threats they pose.

Contrary to the expectation at PDR, the ECS Release A design consists of a single cell which encompasses all DAAC resources and crosses site boundaries. The reason for this compromise are certain multi-cell constraints inherent in the current DCE releases, which will disappear in the future. In particular, the generally available release of DCE (OSF 1.0.3) does not support cross-cell authentication. Thus, users (including applications) in one cell cannot securely access services in another cell. The single-cell security architecture planned for Release A represents the optimum middleware infrastructure and associated security, while living within the constraints of the currently available technology. It is depicted in Figure 6.2-2.

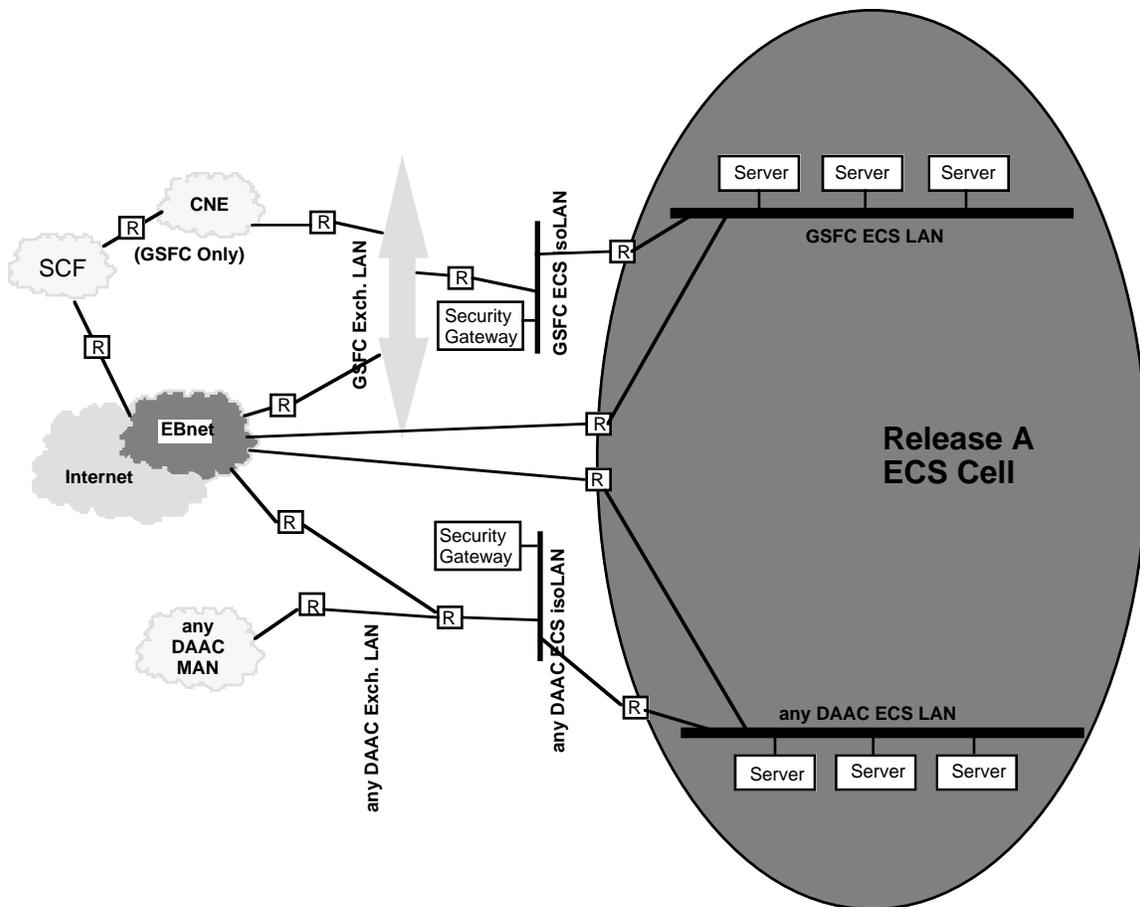


Figure 6.2-2. Release A Security Architecture

However, the single-cell approach should be viewed only as an intermediate step to the more desirable implementation. The maturing OSF/DCE product plans to support the necessary functions in the next major release. Thus, the Release B evolution path is toward multiple cells, one-per-DAAC, providing local autonomy and improved reliability and availability across the entire (multi-DAAC) system. The cell partitioning strategy does not prohibit direct high-speed access to ECS production LANs, and provides efficient location of system resources through distributed directory and security services in every cell. By delegating the assignment of user privileges independently to each DAAC cell, privileged user access may follow site policy, with little likelihood of direct intrusion or potential for security breach by the general public. By establishing a trust relationship among the various ECS cells, distributing the privilege assignments provides autonomy while enabling users to enter ECS anywhere and have the same view of the system. The principles of the architecture for future releases are further depicted in Figure 6.2-3.

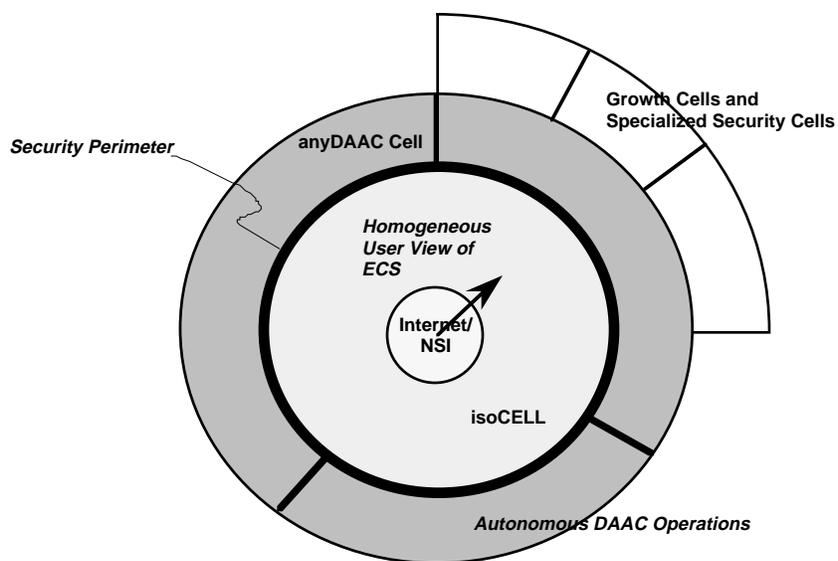


Figure 6.2-3. DCE Cell Partitioning Strategy

6.2.4 Security Implementation within ECS

Figure 6.2-4 illustrates how DCE security is used within ECS to enable a secure distributed computing environment. After the administrator has created an account for a user, the user can participate in a secure DCE system. Typically a user logs in at the beginning of a session through the CSS login facility server. The login facility server sends a request for authentication credentials to Authentication Server. The Authentication Server sends back the authentication credentials, called a Ticket. The *Authentication Server's* replay is encrypted using the user's password, so if the user can supply the right password, the replay can be decrypted and the *Ticket* can be accessed. Tickets are used by clients to authenticate themselves to servers; that is, to prove that clients are really who they say they are.

Next, the *Login Facility* sends the Ticket to the Privilege Server. The *Privilege Server* returns authentication credentials, called a PAC (Privilege Attribute Certificate). The PAC contains authentication information specific to the user, such as which groups the user belongs to. PACs are used to authorize users; that is, to help a server decide whether users should be granted access to resources that the server manages. When the Login Facility has finished running, the user has a security environment and can communicate in a secure way with application servers. Major aspects of security include data privacy and integrity, authentication, and authorization.

Data integrity ensures that data in transit or in storage is not modified. DCE provides an option (specified by the application programmer when designing the respective interface) to add encrypted checksums to the data. DCE can also ensure that data in transit cannot be read any unauthorized parties by encrypting the data stream itself. The DBMS, UNIX file system, and access to either (protected by DCE authentication/authorization) are responsible for the integrity of data in storage.

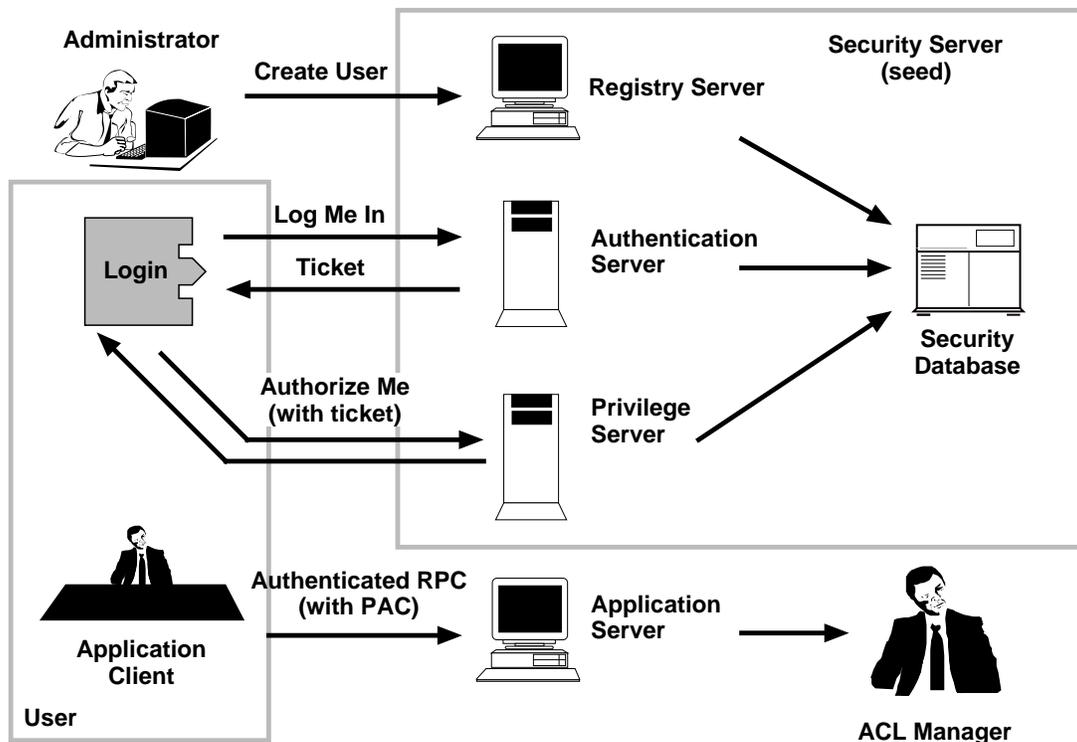


Figure 6.2-4. Use of DCE within ECS

The protection level for data privacy or confidentiality is, like data integrity, again partially determined by the application designer. Full encryption, used to guarantee the privacy of data either in transit or in storage, is expensive in terms of system performance and should be used sparingly. The same controls providing for data integrity (DBMS, file system, authentication, authorization) also are used to provide for data privacy for data in storage. For Release A, no requirements for data privacy have yet been identified, though there is discussion of some data being temporarily proprietary, and therefore subject to read access controls.

DCE also provides strong authentication, being able to verify the identity of a principal without passing his or her password in the clear. With DCE, authentication is done through a trusted third party. Authentication can be performed manually through the principal, or automatically within normal client/server authentication processes. A principal authentication, done by a user during login, is an interactive process in which the user password is supplied on the users command line, but not passed across the network to the server. A non-interactive principal authentication occurs when a host is booted and a server needs to acquire a different identity other than the default (root).

The process of checking the privileges of a principal (i.e., whether he or she is allowed to access a system resource on the network) is called Authorization. Authorization can be based on the name of the principal (this is called name-based authorization), or on the group to which the principal belongs (this is called PAC based authorization). The specific authorization type is specified upon initial session login by the selection of the appropriate authorization protocol. Group based authorization is useful when many users share similar privileges; by assigning their privileges to a group, the administration of the privileges is simplified, but it is still possible to maintain the user's

identity within the system. Name based security, on the other hand, is needed when individual users have specific privileges not shared by others (as may be the case, for example, with DAAC operations personnel).

DCE manages authorizations using Access Control Lists (ACL). A component of DCE, the ACL manager, provides functionality needed for the authorization process. It defines access control permissions, creates and associates ACLs to objects, creates and manages ACL databases, and supports standard interfaces for external system. CSS provides an API via which ECS applications can inquire whether a user is authorized to obtain the service he or she requested. This interface hides the nature and the mechanics of the authorization process from the application - the application need not even “know” the user’s login name or group account.

6.2.5 Non-DCE Based Security

Within ECS Release A, not all communications take place via DCE or OODCE. For example, bulk data transfers regularly take place using file transfer (ftp). ECS secures these interfaces by making them an integral part of a client-server sessions which is initially established using DCE (via the OODCE layer). For example, in order to initiate a file transfer, a client such as the processing subsystem needs to issue an authorized data staging request to the data server subsystem. That is, the initiating event will have had to pass DCE authentication. In addition, depending on the sensitivity of the data transfer, a Kerberized version of ftp will be supported. By issuing one-time userid/passwords for specified bulk data transfers, additional protection may thus be provided.

The situation is more complex at the external interfaces which ECS has with other systems or legacy software which is reused for Release A (e.g., the Version 0 IMS). This is further discussed in the next section of this overview, as part of the External Interface Architecture.

6.3 External Interface Architecture

6.3.1 Overview

ECS provides interfaces with external systems (e.g., TSDIS) or legacy software (e.g., the Version 0 IMS). These interfaces pose challenges in several distinct areas:

- The external systems or legacy software components submit requests which are not formatted in accordance with ECS rules. For example, TSDIS uses SFDU messages which follow CCSDS standards; the Version 0 IMS formats its requests in an Object Description Language (ODL).
- In addition, data exchanged with external systems often need to be reformatted during import into ECS, or export to the external system. For example, the Version 0 IMS expects result sets (i.e., lists of found data granules) to be formatted in ODL.
- ECS uses a distributed object infrastructure in which service requests are exchanged as distributed “object method” invocations using OODCE (and ECS plans to migrate to CORBA eventually). TSDIS and the Version 0 IMS, on the other hand, uses tcp/ip sockets to communicate with ECS.

- ECS uses DCE-based security. All requests are verified using DCE Access Control Lists (ACL) as described in the Security Section of this overview. Users of ECS need to login to their password-secured ECS account before they can obtain any services from ECS. None of the Release A external systems nor the Version 0 IMS currently use DCE authentication and security.

The ECS interfaces with external systems are called gateways. For Release A, the responsibility for data access gateways (from the Version 0 IMS and TSDIS) belongs to the Data Management Subsystem, whereas the responsibility for data ingest gateways belongs to the Ingest Subsystem. However, all gateways pose similar design issues which can be grouped according to the above areas:

- Application level issues. The incoming request must be parsed and interpreted, data references must be translated from the vocabulary of the external system (e.g., Version 0) into that of the ECS data model, and the ECS objects and methods must be called which are needed to service the original request.
- Communications level issues. There needs to be support to transition from the external communications method (such as sockets) to the ECS internal communications infrastructure (OODCE based distributed objects).
- Security level issues. An ECS login must be performed before the request can be passed on to ECS. Security policies which may be specific to each external interface determine how the gateway will verify the identity of the external users, what ECS account it will use to submit the external request, and what kind of other protection mechanisms it will employ to safeguard the external communication, as well as the ECS system.

ECS has decided to base all gateway designs on a generic blueprint called the “Gateway”. Elements of that gateway will be first implemented in Ir-1, and will be refined by Release A. The generic blueprint pursues the following goals:

- Clearly separate applications, security and communications aspect of the gateway, such that they each can be re-used independently, and also to simplify implementation.
- In particular, standardize the interfaces for the security gateway, such that if the security policy for an external interface changes, the changes to the security gateway software are transparent to the communications and applications gateway components. Such changes in security policy are likely as ECS migrates through its various releases (e.g., due to a change in the security sensitivity levels assigned to ECS by NASA).

For example, a large portion of the Version 0 Gateway is applicable to the TSDIS gateway: TSDIS inventory search and data ordering capabilities are similar to (actually subsets of) the Version 0 capabilities, but they do not use Version 0 data names and valids, and the requests are not formatted in ODL. The security aspects of the two interfaces, however, are likely to be different (e.g., TSDIS may use Kerberos in its ECS interface). Reuse of the Version 0 gateway software for the TSDIS interface, therefore, will benefit from a clean separation of the various gateway functions.

The following sections first provide an overview of the generic gateway architecture, followed by an example in which the external communication uses Kerberos (this example may be applicable to the TSDIS interface).

6.3.2 Gateway Architecture

Figure 6.3-1 shows the general gateway architecture. In the figure, the gateway provides an interface between an external application (not shown) and some ECS Service.

The architecture shows the following six interfaces and components:

1. The communications gateway interfaces with the external system or software (Interface 1). From an application perspective, this interface implements a standard polling or non-polling mechanism to obtain service requests (for examples and details see the design of the Ingest subsystem), but its details are otherwise irrelevant. When a send or receive is permitted, and what is sent or received (e.g., DANs, Ingest Requests, TSDIS Status Requests, etc.), and message formats and layouts are part of the applications protocol and are of no interest to the communications gateway.

The communications gateway is configurable (e.g., communications method, communications addresses, directory locations, mailbox addresses, etc.). The communications gateway may remove information from incoming messages which is only relevant for communications purposes.

2. The security gateway might process all incoming and outgoing messages, only incoming ones, or only the first message (via Interfaces 2 and 3). For performance reasons, the security gateway may remove itself from the event path, in which case communications gateway and applications gateway may communicate directly (via Interface 4). The security gateway may remove information from incoming messages which is only relevant for security purposes.

The gateway will establish a DCE identity for the subsequent request and data exchange, in general after verifying the identity and authority of the external user. The specifics of this depend on the security policy implemented by the gateway. For example, the Version 0 IMS gateway will verify the user name and password, and then use it to log the external user into DCE.

3. The applications gateway decodes an incoming message, determines the nature of the request to be issued to ECS, creates the appropriate ECS client objects, formats and issues the appropriate client methods in accordance with the API of the particular ECS service which is the target of the request. The Application Gateway should call the CSS Security Services to verify that the user (now identified as an ECS account) is authorized to issue the gateway request. Interfaces 5 and 6 are normal ECS internal interfaces.
4. The applications gateway receives outgoing notices or responses from the ECS service via the corresponding client objects, reformats these in accordance with the applications protocol governing that external interface and sends them to the target recipient via the Communications Gateway (via Interface 4), or perhaps via the security gateway (via Interfaces 3 and 2) depending on security policy as controlled by the security gateway.

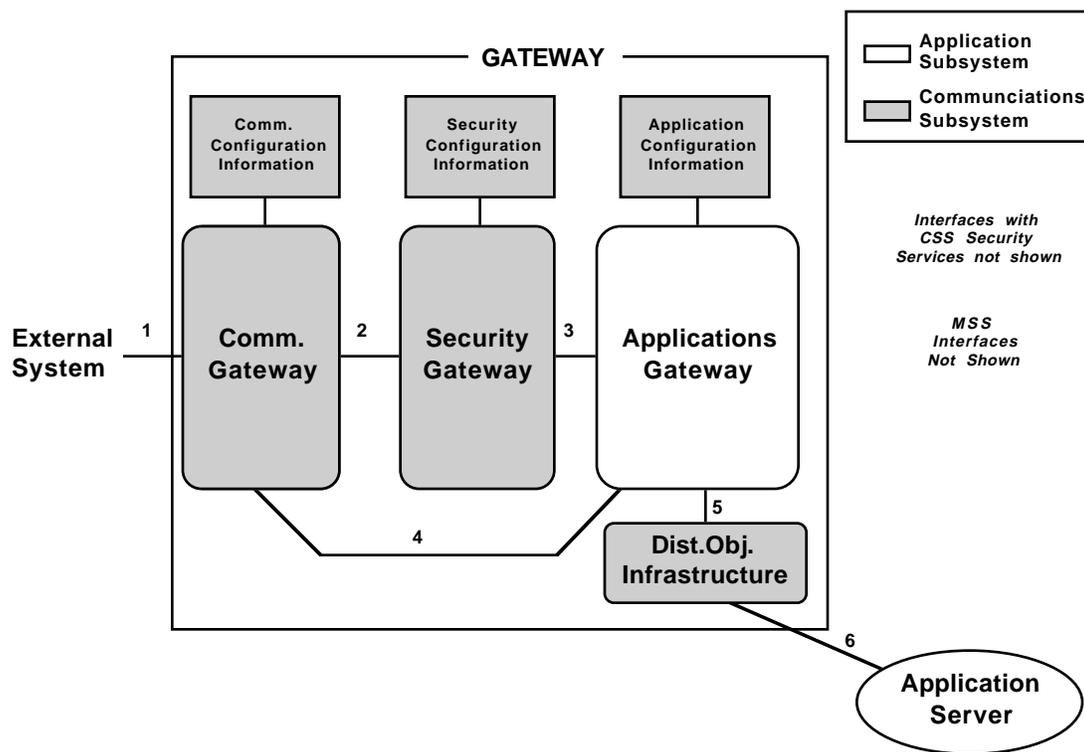


Figure 6.3-1. Gateway Architecture

The presence of the security filter must be transparent to the application. That is, the interface between the communications gateway and the applications gateway (Interface 4) will be identical to that between the security gateway and the applications gateway (Interface 3).

Within ECS, the responsibility for the communications and security gateway is allocated to the communications services subsystem (CSS), whereas the applications gateway responsibility falls into the area of the Ingest Subsystem (INS) for ingest interfaces, and the data management subsystem (DMS) for data access interfaces.

6.3.3 Gateway Example

This example focuses on the security gateway to illustrate how different “plug-in” security gateways can implement different security policies. In the example, the external system has Kerberos installed. ECS will maintain user accounts for the users from the external system in both DCE and Kerberos Security databases. The functions of the security gateway then are to:

- authenticate the external users (validate the credentials of the principal coming into the gateway),

- authorize the external users (OPTIONAL, so only users with proper privileges can make the requests to certain ECS applications)
- provide data integrity (OPTIONAL, make sure that the data coming from the external entity is not modified in transit)
- provide data privacy (OPTIONAL, to protect the contents of the data coming from external entity from eavesdropping).

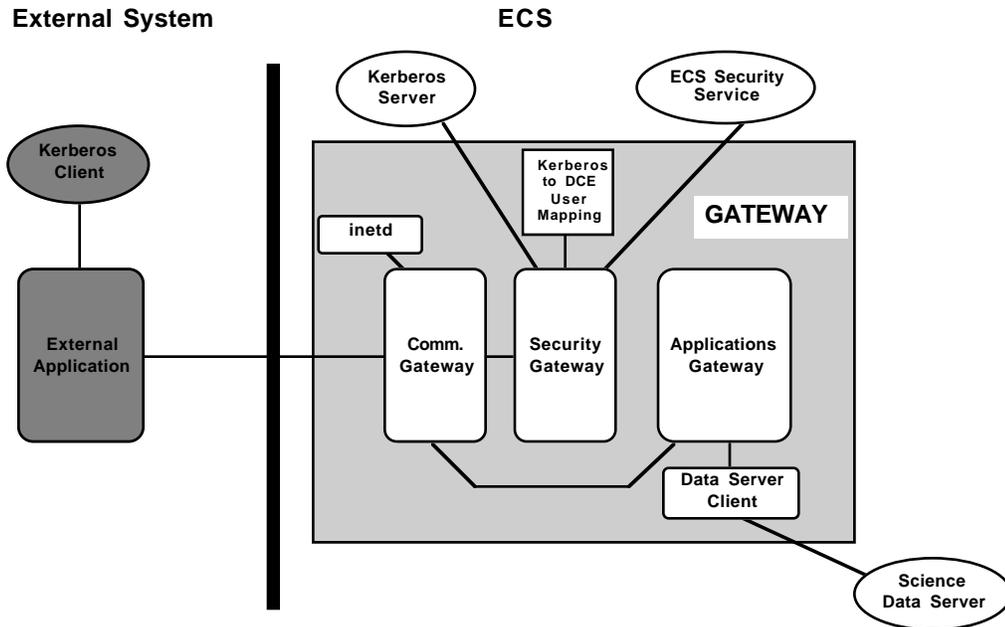


Figure 6.3-2. Kerberos Security Gateway Example

The resulting gateway implementation is shown in Figure 6.3-2. This design refines the general architecture as follows:

- The external application interfaces with the local Kerberos client, which in turn interfaces with an ECS operated Kerberos server to obtain a Ticket Granting Ticket (TGT) and subsequently, a Kerberos ticket to access the ECS gateway.
- The ticket is included in a message sent to the ECS resident gateway via a tcp/ip socket. The message is accepted by the Communications Gateway and passed on to the Security Gateway (note that the first message will be handled by the inetd, which will assign a port to this communications session).

- The Security Gateway verifies the ticket and establishes the external user's identity. It uses a mapping policy to determine the user's ECS identity - the policy might be to use the exact same identity, or there might be a substitution rule or table for all or some accounts. The security gateway then logs the account into DCE. The subsequent ECS requests which will result from this particular communications session will be submitted under this ECS identity.
- The first message exchange was dedicated to the authentication process. The application gateway is unaware that it took place. The security gateway now removes itself from the interchange. Subsequent messages will pass directly from the communications gateway to the application gateway (but only, if the initial authentication was successful).

It is easy to see how the security gateway design could be modified to require authentication for each incoming and outgoing message or to add/remove cryptographic checksums in order to ensure the integrity of the transferred information without affecting the application gateway.

6.4 System Management Architecture

ECS is a large, distributed, heterogeneous system consisting of many off the shelf (OTS) and developed components. The management and operation (M&O) of such a system poses numerous challenges. This section reviews the architectural concepts which are the foundation of the ECS system management support and which are applied across ECS subsystems. The section is organized as follows:

- Sections 6.4.1 and 6.4.2 review the various levels of system management and the geographic distribution of management responsibilities
- Section 6.4.3 lists and briefly explains the M&O positions which have been defined for the ECS system
- Section 6.4.4 explains how the system management responsibilities are divided up between the ECS subsystems
- The most important subsystem in this context is the Management Services Subsystem (MSS) within the CSMS segment. Section 6.4.5 provides an overview of its functions and its relationship to the application subsystems.
- Section 6.4.6 discusses an important aspect of system management and MSS, namely the handling and management of errors and faults, and the roles which the applications subsystems and MSS play in it.
- Finally, Section 6.4.7 describes how these concepts combine to support management and operations reporting.

6.4.1 System Management Levels

ECS provides system management at several levels, namely at the:

- Individual Service Level - certain types of services require specialized operator knowledge. For example, a DBMS administrator is expected to know database technology in general

and the specific product (Sybase) being used by the system; the manager of the science processing environment needs experience with UNIX batch production environments as well as some familiarity with various groups of science algorithms and their interdependencies

- Subsystem Level - while the smooth operation of the various services which make up a subsystem is important, it is the subsystem as a whole which provides an essential and integral set of ECS capabilities. For example, the Ingest Subsystem is needed to load and archive level 0 data. Individual component failures are of interest insofar as they impede the ability of the Ingest Subsystem to perform that function.
- Site Level - many M&O functions, on the other hand, are best executed on a site-wide basis, for example, the health of computing platforms, their operating software and applications, peripherals, and networks and networking devices. This reduces staffing requirements, and the maintenance of site-wide awareness of overall system performance and status makes for better planning and better management decisions in emergency situations.
- ECS Level - monitoring and management of the ECS mission, its budgets, and resources, on the other hand, requires an overall system view. This is also referred to as enterprise level management.

6.4.2 Distribution of Management Functions

Geographically, system management occurs at the individual DAAC, and at the SMC at GSFC, which performs Enterprise Monitoring and Coordination (EMC) functions. As a rule, service, subsystem, and site level management are performed at each DAAC; enterprise management occurs at the SMC. Table 6.4-1 lists the various types of management services, and shows where they are performed or how management responsibilities are allocated between DAACs and SMC.

Table 6.4-1. Management Service Distribution (1 of 3)

Management Service	Enterprise Monitor and Coordination (EMC)	Local System Management (LSM)	Comments
Policies and Procedures	Provide Policy Decisions, Coordinate Policy, Policy Compliance Monitoring	Coordinate Policy, Site Level Policy Decisions, Policy Compliance Monitoring and reporting	Policy Management in Release A will be through the use of Office Automation Tools.
Fault Management	Receive Summary Reports from Sites, Monitor System Wide Resources (WANs), Perform Trend Analysis	Monitor, detect, isolate, diagnose, and recover from faults within domain	Largely COTS capabilities (HPOV), EMC maintains system-wide view from Site updates and monitoring

Table 6.4-1. Management Service Distribution (2 of 3)

Management Service	Enterprise Monitor and Coordination (EMC)	Local System Management (LSM)	Comments
Performance	Trend analysis and system-wide view provided from Site updates	Collect server, hardware, and network performance data, analyze performance data, tune and report to SDPS/FOS/EMC	Site performance is cooperative effort between LSM and SDPS/FOS Trends are through roll up of site reports
Trouble Ticketing	Summary Reports, View selected Site problems, support resolution	Document problem reports, track actions and closure. User and resource summaries	Remedy selected as TT package.
Physical Configuration Management	Same as DAAC	Maintain Physical location and configuration information	Commercial package to locate and record resources, detects changes to approved configuration
Security	Policy flowdown, system-wide monitoring and analysis DCE Cell Management	Authentication, authorization, intrusion detection, DCE Cell Management	Largely public domain and COTS , HAL for cell management Policy flowdown and administration is through OA tools
Inventory	System-wide inventory creation and management	Site inventory data maintenance and management	Inventory Management in Release A is through OA tools
Logistics	System-wide monitoring of spares and consumables	Site-level monitoring of spares and consumables including replenishment	Logistics Management in Release A is through OA tools
Maintenance	System-wide maintenance analysis	Establish and maintain PM schedules, monitor and coordinate off-site maintenance	Sites maintain schedules and records through OA tools
Configuration Management (CM) (Software Change Manager)	Software CM of ECS baseline	Software CM for Site Baseline	Clearcase selected for Software CM
Baseline Manager	Consolidated baseline for system wide configuration and dependencies	Maintain site baseline for operational system configuration	COTS being evaluated, Selection expected before CDR

Table 6.4-1. Management Service Distribution (3 of 3)

Management Service	Enterprise Monitor and Coordination (EMC)	Local System Management (LSM)	Comments
Change Request Manager	Maintain system wide status of change requests	Maintain record of configuration change requests, tracks status,	COTS, DDTS has been selected based on evaluation and project experience with product.
Training	Coordinate training schedules, curricula, user feedback, and develop materials	Provide input on training schedules, curricula, local course development, and evaluation	Training Management in Release A is through OA tools used at Sites and SMC
Planning	System-wide schedule policy, priorities, performance assessment System wide ground event coordination	Schedule own resources based on system-wide priorities and policies, plan ground events and interface with FOS and PDPS	Release A provides for DAAC Resource planning with limited "Rollup capability for SMC
Directory	Content, format, and update procedures	Maintain, replicate, distribute	Limited to user directory information through Release A
Reports	System-wide reporting based on "roll-up" of Site level data	Site-level reporting on performance, security, fault, and configuration information	Ad-hoc reporting from DBMS, other reports directly from COTS products

6.4.3 M&O Positions

Table 6.4-2 lists the ECS M&O positions and their responsibilities. These positions are defined in a functional sense - they represent a collection of related operator, support, or management functions. Several positions could be filled by a single person, provided he or she has the required qualifications and training.

Table 6.4-2 Ops Positions and Responsibilities (1 of 3)

Operator	Responsibilities
H/W Maintenance Technician	Trouble shoot and isolate problems to and perform maintenance at the LRU level, and/or coordinate with vendor for said maintenance of COTS hardware .
System Administrator/ Computer Operator	Operate the host processors, support restarts/reboots, configure / re-configure resources as directed, monitor system status, respond to console messages and do initial program loads for all system upgrades. Perform minor housekeeping maintenance, system back-up and recovery, operator level preventive maintenance and problem diagnosis and recovery. Administer the networked workstations supporting the M&O staff.
Operations Supervisor	Provide first line supervision of DAAC ECS Operations including conflict resolution, policy enforcement, productivity monitoring and staff supervision. Serve as backup resource to fill in for operations personnel when required due to illness, vacations, etc.
Resource Planner / Performance Analyst	Maintain and modify hardware characteristics database; generate monthly, weekly and daily hardware activity schedules; monitor system anomaly tracking and analysis. Monitor, analyze, trend and report local system performance; recommend and track implementation of changes to system control parameters to improve system performance; participate, with the SMC Performance Analyst, in monitoring overall ECS system-wide performance. Alert DAAC to potential performance issues and problems and SMC, ECS M&O Office and/or Project management to circumstances that may require coordination between DAACs with Project (ESDIS) participation.
Resource Manager	On-line management of operational resources. Configure operational resources in accordance with approved resource schedule and operational resource baseline. Re-configure as required in response to utilization requirements, changes and anomalies. Monitor, manage and control the local area network and system via the LSM software suite, take corrective action to respond to anomalies, coordinate with SMC for wide area network problems, assist in diagnosis and isolation of hardware and software problems, and maintain system configuration tables. Monitor and report on the network performance to management
Sustaining Engineer	Analyze and support problem resolution and engineering change activities, including all ECS interfaces; analyze and identify ways to accommodate needed improvements, new technologies and new concepts; plan and manage system upgrades and evolution; control and maintain ECS updates and perform the activities necessary to assure ECS reliability, maintainability, and availability. As TAG member, evaluate user inputs and monitor system performance to tune the system for optimum response and support.
Configuration Management Administrator	Provide configuration and problem management system administration. Maintain control of all DAAC configured hardware, software, science software and specified DAAC documents. Ensure that changes to the hardware, software, and procedures are properly documented and coordinated. Coordinate usage of approved configuration management procedures with elements and external interface configuration management. Assist in the development and administration of the library with respect to configuration management procedures. Provide recording tasks for DAAC CCB, generate configuration and problem status reports and prepare agendas for and schedule CCB meetings.

Table 6.4-2 Ops Positions and Responsibilities (2 of 3)

Operator	Responsibilities
Property Manager	Provide control of Contractor and Government ECS property and continuous audit trail from receipt of item until transfer of accountability. Property management responsibility for ECS equipment until accepted by CO/COTR and for equipment for which the contractor has M&O responsibility. Interface with ECS ILS function at GSFC in coordination of delivery of COTS hardware or software, handle ECS site hardware shipping and receiving, act as local ILS representative for ECS. Control and record consumables and inventory. Receive, mark, report, store, stage, control, pack and ship ECS spares, repair parts, consumables, and HW/SW items received at or shipped from each DAAC.
S/W Maintainer / Programmer	Provide / support site problem resolution, integration and test of system changes and develop DAAC unique extensions to ECS software. Produce, deliver and document the corrections, modifications, and enhancements made to ECS software (including COTS), and/or adapt or incorporate COTS software for ECS use.
Sr. Science Coordinator	DAAC Sustaining Engineering team liaison / interface to the DAAC Scientist and the DAAC Science support staff and through them, the DAAC's user community. Contractor science lead for change review, science software I&T and science production planning and operations.
System Tester	Responsible for all system and acceptance testing of software and hardware modifications and upgrades. Maintain and update test procedures and data bases.
Science S/W Integrator (ECS)	After transition, provide on-site Science Tool Kit expertise and support to DAAC Scientist / Science Support Teams in the test and integration of science S/W (both updates and new S/W) into the ECS system. (Note: Prior to transition, this expertise is provided initially by the ECS Science Office and then by the ECS Algorithm (Science S/W) Development Support personnel at the DAAC.)
DAAC Assistant	Provide technical support to DAAC, including its user services personnel. This includes the provision of any necessary training to the DAAC staff to enable them to continue performing operations for each subsequent release or system upgrade.
Production Planner / Scheduler	Develop and maintain Production Planning Data Base. Develop, coordinate and monitor Data Availability Schedules with external providers. Generate production resource requirements. Develop and maintain production plans and schedules.
Production Monitor / QA	Manage processing queues, monitor Data Processing Request execution status, manage / optimize production resource utilization. Monitor quality and completeness of input and output, using science software provided QA tools. Provide production reports.
Ingest - Distribution Technician	Receive, log, ingest and disposition all incoming non-electronic media. Monitor electronic and manual ingest of data. Coordinate with sender to resolve any ingest problems. Monitor electronic and perform physical distribution of data. Load / unload media from write devices, assemble "data packages", package, label, and ship output to science users. Follows up and trace undelivered output. Receive, open, and route incoming mail to appropriate action department.
Archive (Data Server) Manager	Manage input, storage and output of science data. Manage ingest data server(s) and archive, ensuring data is made available for use in science production operations, to fill user requests and that data is successfully archived. Oversee ingest and distribution functions. Ensure the successful backup of science data. Perform periodic media sampling and refresh as required and periodic preventive maintenance of archive media. Establish and maintain data server and ingest-archive-retrieval subsystem configurations. Status and report on ingest-archive-retrieval subsystem performance.

Table 6.4-2 Ops Positions and Responsibilities (3 of 3)

Operator	Responsibilities
Data Base Administrator	Maintain the data bases and structure management for the integrated SDPS and the LSM. Perform the data base administration utilities, such as data base backup and recovery, performance monitoring and tuning. Administer data base access control, and daily data base synchronization.
Algorithm Development Support	Provide support to DAAC Scientist, Science Support Teams and the Instrument Team scientists in the development and integration of algorithms (science S/W) for both updates and new algorithms into the ECS system. In accordance with SCF policy, provide science production QA and problem resolution with the SCF. Support USO in responding to highly technical user assistance requests. (Note: prior to M&O staffing at each DAAC, these functions are performed by the ECS Science Office.)
Data Specialist	Possess an intimate knowledge of the DAAC's data and metadata sets and are expert users of the ECS suite of data access software tools. Answer detailed questions concerning the discipline data stored at their DAAC. Support initialization and maintenance of data server and production planning data base, advertisement of new data sets and services and responding to technical user requests. Work with Database Administrator in structuring data base, data sets and metadata.
User Assistance	Provide the user support interface (phone; electronic and hard copy mail) to help users locate and order data, to request processing and to report problems. Answer general user questions about the ECS services, tools and data. Assist new users in registration and generation of user profiles.

6.4.4 Division of Responsibilities

The ECS subsystems have distinct responsibilities for the management of various resources within ECS. In this context, the term “managed resource” refers to the following types of objects (they are also called “Managed Objects”):

- networks and network devices
- computing platforms and their peripherals and operating systems
- other special devices with their operating software (e.g., archives)
- off-the-shelf and custom developed application software services

As a general rule, each individual subsystem has the responsibility to provide system management functions at the service and subsystem level for its custom software services, any off-the-shelf software services, and any special devices employed by that subsystem. Subsystems are also responsible for supporting site and enterprise level system management through appropriate interfaces with the ECS system management infrastructure, unless such interfaces are provided by a vendor as off-the-shelf items.

Examples of service and subsystem level managed resources include databases and their database management systems, ECS application services such as data server search and access services, data distribution services, and ingest services. It is the responsibility of the respective subsystem to support the operator positions which are specifically assigned to support that subsystem, such as

an Ingest or Distribution Technician or an Archive Manager. Some off-the-shelf products are used across (or by) several subsystems. Support for the corresponding operator positions is generally provided by the OTS product vendor, however, any additional support functions which must be tailored to the specific needs of a subsystem are the responsibility of that subsystem.

MSS, on the other hand, has the responsibility for providing site and enterprise system management functions. This includes the management of networks, network devices, hardware platforms and their peripherals and operating software, and any off-the-shelf components for which the vendor provides interfaces into the ECS system management infrastructure.

Examples of system resources that MSS manages include routers, hubs, communications links (FDDI, Ethernet, and WAN links), hosts, applications, processes, operating systems, logical devices, software libraries, file systems, and peripheral devices. In managing these resources, MSS is responsible for accountability management (maintaining user profiles, maintaining an audit trail of user actions on each managed resource, and maintaining a data audit trail of actions performed on a data item); configuration management (maintaining the resource baseline, managing software changes to managed resources, and tracking change requests for managed resources); fault management (identifying, isolating, and resolving faults detected on a managed resource); performance management (monitoring both real-time and long term resource performance), and security management (protecting managed resources from security intrusions and controlling access to managed resources).

In the management of resources, the subsystems may also need to deal with security issues such as the authorization rights of a client to access a requested service or resource, the detection, reporting and possible recovery from errors (such as errors in data, or errors with media in dedicated hardware such as the archive server). Application-level performance metrics monitored may include items such as the number of data products that have been ordered, the number of data sets that have been ingested, and the number of data products that have been processed. In order to assist the subsystems in gathering this management data where necessary, MSS provides the subsystems with the capability to log the information through the use of its management agent services. This information can then be used directly by the other subsystem or imported into the management database for reporting via the report generation application. Figure 6.4-1 shows, at a conceptual level, the flow of data and commands between application subsystems and MSS.

Each of the application services will interface with MSS through an MSS provided interface. Applications provide information about themselves (e.g., current state and performance statistics), report noteworthy events (e.g., errors and faults), accept management called “instrumentation” (e.g., shut down commands), and receive notifications of events to which they need to react. Application platforms (and devices) also contain system management “agents” which are responsible for providing similar functions for the platform and its operating software (or for the device).

Events are logged at three levels. They are logged locally initially (i.e., on the application platform). These logs may be managed by the application (e.g., an ingest log), or they may be managed entirely by MSS (if the application itself has no direct interest in the event log). On a regular basis (or as needed), the event reports are consolidated into a site event history database

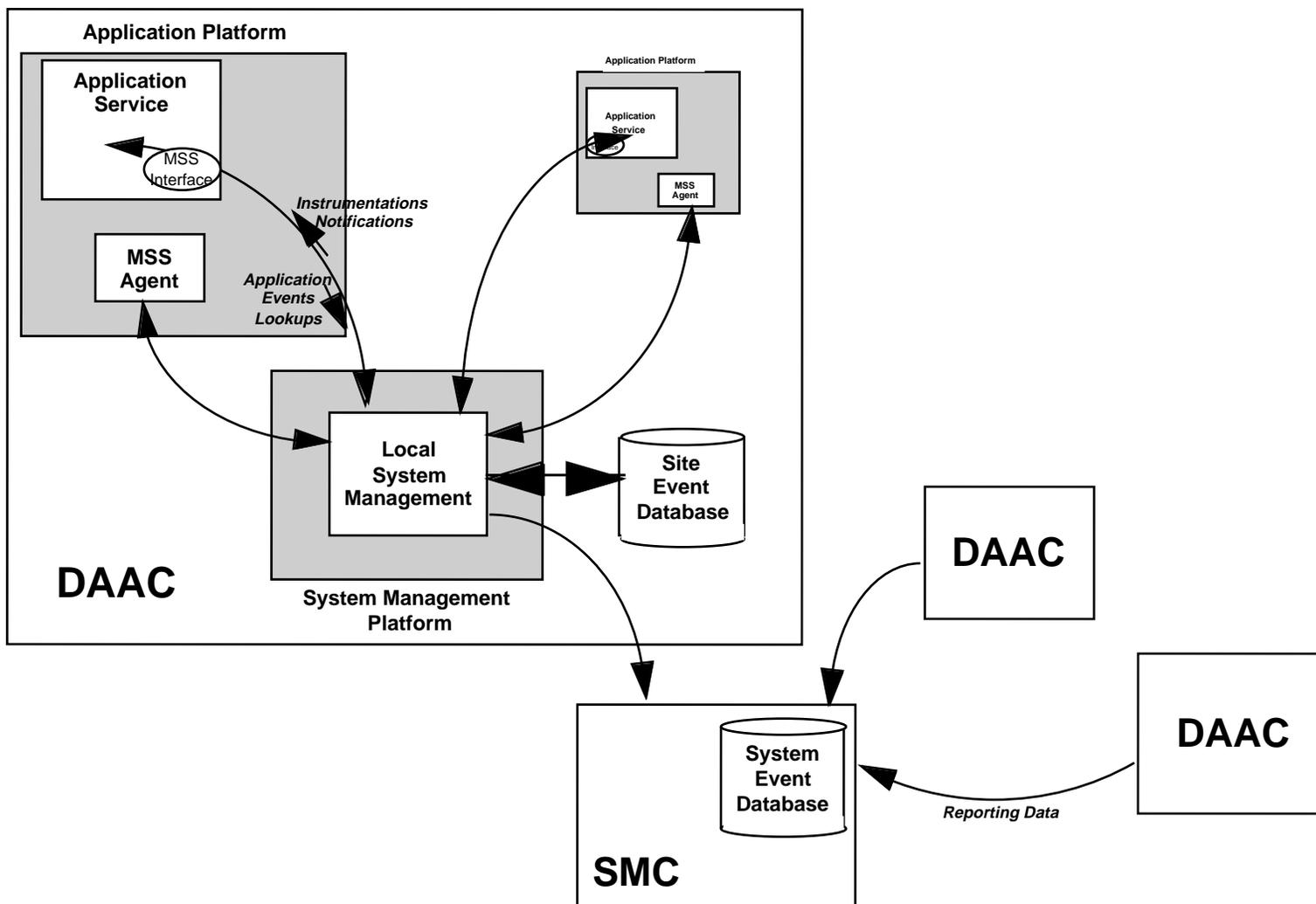


Figure 6.4-1. System Management Data and Command Flows

(managed by Sybase), which can be used for ad-hoc and regular site level management reporting (see Section 6.4.6). Finally, extracts and summaries of the site event data will be forwarded to the SMC for consolidation and integration across the ECS. Logging is discussed in more detail in Section 6.4.6 on reporting, because it forms the basis for the management reporting functions.

SNMP has been chosen as the management protocol since it is the defacto and Internet standard protocol for network management in TCP/IP environments. Applications, on the other hand, interface with MSS via their MSS interfaces using OODCE.

6.4.5 MSS Overview

The most important component in the overall system management architecture is the MSS subsystem of the CSMS segment. It has the responsibility for coordinating system management at the site and system level. The following were key drivers in its design:

- there will be no single point of failure
- DAACs will be able to manage their own resources
- it will be possible to perform system-wide monitoring and coordination
- the architecture is neutral with respect to system management policy, for example, how enterprise management functions are geographically partitioned
- within an implementation, it will be possible to distribute authority based on policy
- system management will not interfere with the accomplishment of operational DAAC functions

Avoiding any single point of failure is paramount to good system engineering practice. All system functions must be implemented in a manner that provides high availability and failure protection. The concept of a central monitoring and coordination function (versus central management and control) allows the SMC to monitor activities between all DAACs and within DAACs to identify problem areas and coordinate solutions as required. Performance analysis of the total system would become a predominant EMC function. Flows outside of ECS would be monitored at the SMC.

Local management at the DAACs of their unique resources, with visibility when required across the system is synonymous with the concept of federation in advanced enterprise management solutions. The federated approach to systems management allows the DAACs to work problems between themselves without having to have GSFC involved unless appropriate. This represents an expedient approach to problem resolution, by allowing the involved parties to directly resolve a problem.

A policy neutral architecture defers policy decisions out to the implementation (design) stage, and allows flexibility for system reconfiguration to reflect policy changes over time. Distribution of authority provides flexibility for monitoring and coordination backup, both at the DAACs and at the SMC. Finally, the concept of systems management being unobtrusive to in-line operations acknowledges that systems management is a *service* to users.

Figure 6.4-2 expands on Figure 6.4-1. It illustrates the MSS internal management data flows and the flows to ECS applications. While individual subsystems are independently responsible for process and task management-related aspects of their own subsystem, MSS is responsible for the management (monitor and control) of ECS resources, including networks, systems, and applications. Health and status of ECS resources are monitored in addition to functional areas of performance, fault, accountability, and security. MSS controls the real-time configuration of networks, systems and server applications, including system startup, shutdown, suspend, and resume operations.

The figure provides more detail on the four general forms of intercommunication between MSS and the other ECS subsystems - event logging, notification, lookups, and commands (called instrumentation). Event logging is a one way exchange from the ECS subsystem to MSS to log key subsystem level events into the MSS for monitoring purposes. Examples of application use of the MSS event logger include the start and stop of product generation, and data delivery completion. Notification is a one way exchange from MSS to ECS subsystem applications that are used to notify ECS applications of system health and status concerns that may affect the applications operation. An example of a notification is the message delivery of production string unavailability due to hardware error. Lookup is a bi-directional interface, called by the ECS subsystem applications, to obtain system information from MSS. An example of a lookup is an application request for user profile information for a shipping address to satisfy a product order. Instrumentation is a bi-directional exchange initiated by MSS to the ECS subsystem applications to control applications processes. Generic instrumentation requests include application suspend and resume operations. Application specific instrumentation requests include requests such number of orders processed, number of browses, number of searches, or number of products delivered.

Monitoring of ECS subsystem applications is performed one of two ways. The first method is to collect information about managed objects. This is done through either UNIX commands or other utilities described later in this section. A second method of monitoring is to collect information from managed objects through instrumentation of the managed object. In this case, applications must be developed to provide information on events that generate management data and respond to MSS instrumentation requests.

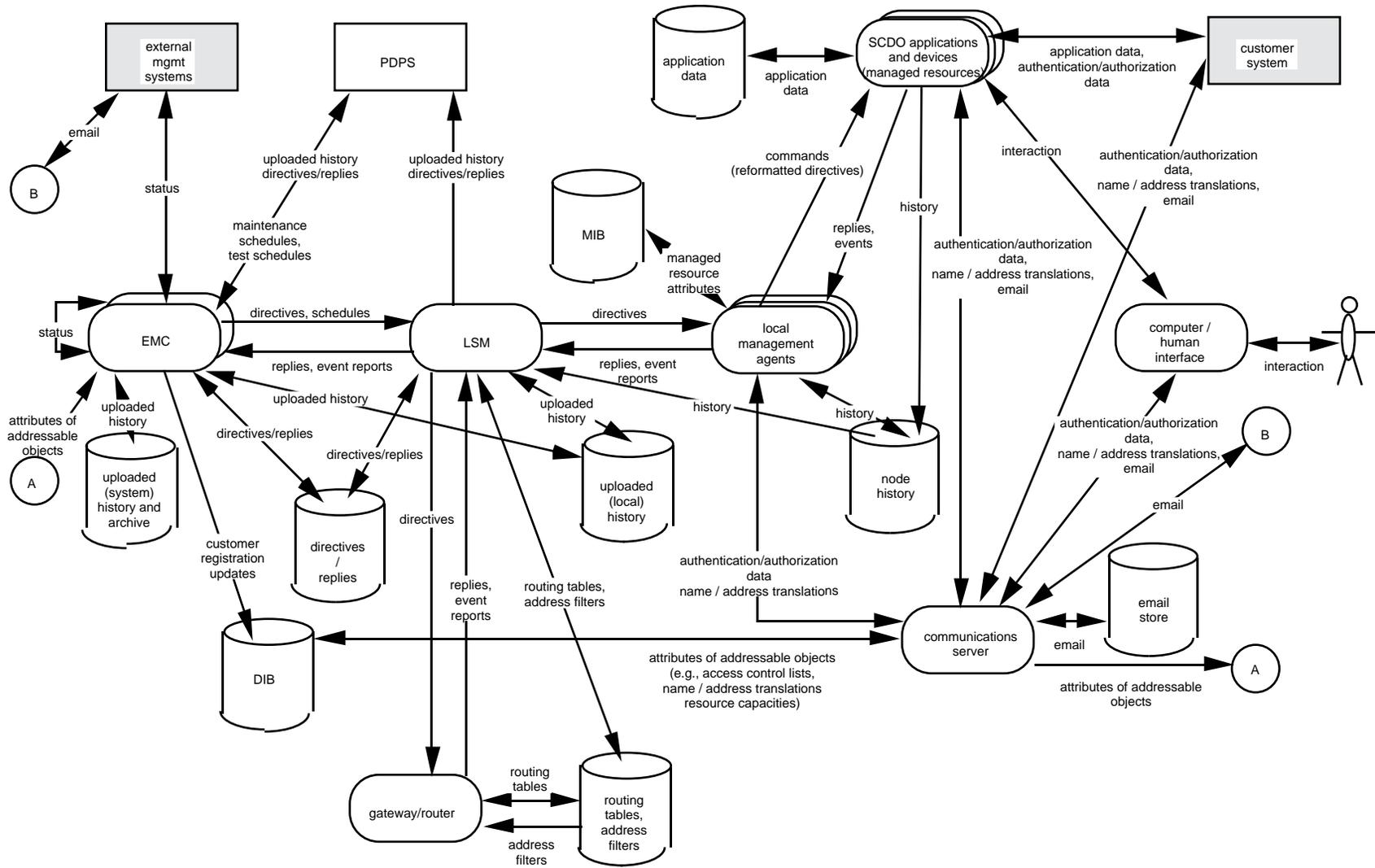


Figure 6.4-2 MSS Management Flows

MSS control of ECS subsystem applications is only through instrumentation requests. In order for instrumentation to function, the ECS subsystem applications must be instrumented to respond to MSS control requests.

The problem of the management of widely distributed resources is solved by using a manager-agent architecture, as depicted in Figure 6.4-3. The manager-agent architecture makes a formal split between two types of functions: managers and agents. Manager functions consume management information and control/initiate management actions with a managed object (resource) as the target of the operation. Agent functions, which are co-resident with the managed objects (resources) produce management data and take action on behalf of managers. That is, the manager makes management requests of the agent, and the agent emits responses to those requests. Agents act as an intermediary between the management applications and the managed objects. Further, agents are also capable of emitting event notifications to the manager. The distinction here between responses and events is that responses from agents always match up to a manager request, while events have no corresponding manager request.

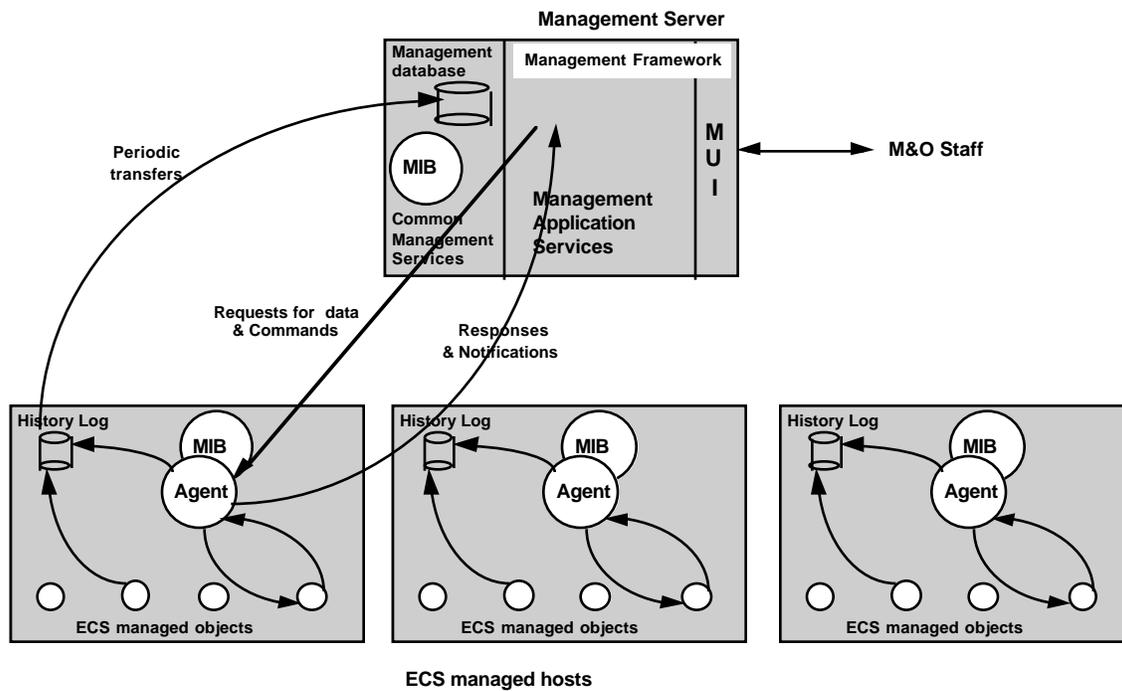


Figure 6.4-3. Manager/Agent Architecture

The MSS Management Agents provides the following functions:

- Enable the management applications to retrieve and to set managed object values.
- Perform local polling on remote hosts to monitor the state of managed resources.
- Handle event logging and notifications.

- Provide instrumentation API to application developers to enable the manageability of ECS applications.
- Define the managed object model to represent the management characteristics of ECS applications.

MSS agents are based on the use of a Management Information Base (MIB). MSS requires that on each managed host, standard SNMP MIB II, Host Resource MIB, and the MIBs of network devices are supported by vendor agents. In addition, a managed object model is defined by MSS for ECS applications in SNMP MIB format. The information contained in the MIB is composed of different types of attributes: configuration, performance, fault, dynamic, static, and traps.

Fully distributed client-server system management application services for distributed enterprises such as ECS are not commercially available today. For Release A, HP OpenView has been chosen as a framework for integration of multi-vendor network and system management products to support migration to a fully integrated management solution as such products become commercially available.

6.4.6 Event, Exception, Error and Fault Handling

One of the main areas in which applications subsystems and MSS will work together is the handling of events, and in particular of errors and faults. Section 6.4.5 described how applications and MSS in principle interface, and how their responsibilities are partitioned.

This section provides a general overview of Exception, Error and Fault handling across subsystems within ECS:

- Section 6.4.6.1 presents a framework for the classification of error and fault related events.
- Section 6.4.6.2 describes rules for error and fault handling. In the ECS design, the two combine to define a set of common software objects for reuse across subsystems which provide error detection and reporting, standardize status information returned from object invocations, and place some application level service on top of the MSS provided infrastructure.
- Section 6.4.6.3 lists and provides brief descriptions of the various classes of faults and error conditions within ECS.

6.4.6.1 Classification Framework

The following provides the definitions for a set of terms and their related software objects in the ECS design.

6.4.6.1.1 Exceptions

An exception is an abnormal condition arising from the environment in which a program executes. An exception may be raised because of an illegal execution of a particular instruction, or it may be triggered by the application itself to report an error. For example, dividing a number by zero would raise an exception.

An exception must be handled by an exception handler. In other words, an exception will propagate upwards the calling chain until an exception handler is found which will deal with the exception. It may or may not be possible to recover from the exception within the program, i.e., the application may revert to a normal state after dealing with the exception, or the program may terminate abnormally.

6.4.6.1.2 Faults

The term “Fault” is defined in the context of the MSS concept of “Managed Objects”. A Managed Object is an system or application resource which is monitored and managed by MSS (see Section 6.4.4). A fault is an unacceptable change in the state of a managed object that leads to a change in the real time configuration of the system. A fault is typically triggered by some software or hardware failure which is detected by or reported to the managed object (e.g., in the form of an error return status on some system call); concurrently and independently, the fault may be also detected by an MSS agent monitoring the resource which failed. As a result of the fault, the system may stop to operate, or generate errors and then continue to run in a mode that is not considered normal (downgraded mode). There is usually a symptom before a fault occurs.

- Example 1: A parity error can kill a process (the process being killed in response to a parity error is a fault) which might have been caused by the failure of a microscopic capacitor to hold a charge, which is reflected in a bit being in the wrong state ("off" rather than "on") (symptom).
- Example 2: When one component (object) requests service from another component, service may not be provided because client has made a valid request to the server, but the server is unable to fulfill the request (due to unavailability of resource - a hardware fault)

It is possible to create fault hierarchies based on a number of categorization factors. Faults can be classified based on

- their nature (e.g. hardware fault, software fault, etc.), or
- from an operations perspective.

From a system management perspective, the latter classifications are more useful and has been adopted by ECS. For example, operational categorization will include the duration of the outage; the level of service that will be available until the fault is repaired and the system has returned to its normal operating state; the nature of the recovery actions which are needed (e.g., automatic vs. manual); and the impact on the rest of the system (e.g., localized, affecting the whole subsystem, affecting a site, or affecting ECS as a whole).

6.4.6.1.3 Errors

An error is an unexpected problem internal to the system which usually manifests itself, eventually in the system's external behavior. It is a deviation from the correct result -- an indication of the occurrence of a fault and may be reported by an exception. The error is usually returned in a status variable. An error may be detected by hardware (e.g., illegal instruction executed, arithmetic overflow), by a run-time support system (e.g., array bounds error), or by the application itself (e.g., checksum error). In other words,

- Example: When one component (object) requests service from another component, service may not be provided because the client has made a request that is outside of the server's specification. In this case, the server is unwilling to provide the service, and the client is the party in error.

Errors are associated with an error explanation (which may be reported to the operator when the error is displayed on a console), and require some recovery procedure in the software which receives the error notification. For example, if an unexpected input occurs, the common recovery procedure would be to skip the record and process the next record. Similarly, the recovery for a duplicate message is to ignore it, while the recovery for a missed message could be a request to resend.

Within ECS, errors are categorized based on the commonality of their error messages and recovery procedure. Each class of error is represented in the design as software object class which encapsulates the desired behavior and information.

6.4.6.1.4 Events

An event is an individual stimulus from one object to another or from the environment to an object. Events include error conditions as well as normal occurrences. Events are atomic and asynchronous. A function call is an event, and the return from that call is a second event. Error is also an event. Only our interpretation makes it an error. Other examples include reception of a command or a message, starting or stopping an activity, creation or deletion of an object, causing a change in the relationship, etc.

6.4.6.2 Error, Fault and Event Handling

The following describes the general framework for handling events within ECS. It includes the detection of events, their identification, reporting, and resultant actions (including entering into or recovering from a degraded mode).

6.4.6.2.1 Event Identification and Attributes

The following are general rules which define how events are identified and what attributes are associated with them, at a minimum:

- Every event will have a unique identifier for the purposes of identification and log maintenance.
- Events will identify the subsystem and subsystem component with which they are associated.
- Every event will have associated with it the identifiers and names of the calling object and the called object. This provides a mechanism for linking events as a chain for traceability.
- Every event will have information about its disposition in the form of a status (error, fault)
- Every event will record its various classification attributes (e.g., severity).
- Every event will have a holder for a message string describing the event

The design provides a common base class for uniform handling of events. It will contain all the attributes and operations necessary for the communication between two objects. The common base class will log certain events automatically via the MSS event reporting interface, and provide capabilities that permit the use of other logs besides the MSS log, e.g., to support debugging/tracing of errors, analyzing performance, generating security audits trails, determining and analyzing resource utilization.

6.4.6.2.2 Exception Handling

The C++ language provides a mechanism for raising and handling exceptions that arise due to anomalies that include user, logic or system errors. If the detecting function cannot deal with the anomaly, it raises, or throws, an exception. A function that handles that kind of exception, catches it. This function is also known as an exception handler.

In C++, whenever an exception is thrown, it cannot be ignored - there must be some kind of notification or termination of the program. If no user-provided exception handler is present, the compiler provides a default mechanism to terminate the program.

The significant benefit of using exceptions is that it significantly reduces the size and complexity of program code and eliminates the need to explicitly test for specific program anomalies. Because of its many advantages, and the fact that ECS software will be developed using C++, it is project directive to use the C++ provided exception handling mechanism as one alternative for handling errors.

The following common framework will be provided during the implementation phase for uniform handling of exceptions:

- Naming convention. All exceptions will be named according to a common convention that is tied to the underlying error being reported through the exception.
- Event / Message Catalog. All events will be collected in an ECS wide event catalog, receive a unique identification, and will have explanatory text (suitable for display to operations personnel) associated with them..
- Linking events. Links must be provided to trace the exception to the underlying error that raised the exception and the corresponding event that produced the error.
- Event hierarchy. Because exceptions are a mechanism for reporting errors, the exception hierarchy will be identical to the error class hierarchy.

6.4.6.2.2 Catch-all Exception Handler

Each application service must have a catch-all exception handler which reports other-wise uncaught exceptions. A catch-all exception handler is a default exception handler that terminates any exception which is propagated up to the top of the calling chain.

6.4.6.2.3 Fault Handling

The ECS Fault handling is performed by MSS which provides the following services:

- MSS Objects for notification, lookup, event logging and instrumentation (a common interface for handling faults through change of state of managed objects and MSS Event log)
- Fault Management including notification to dependents that a fault has occurred and launch of a recovery procedure to recover from the fault

All faults are reported, and the reported information includes a fault classification attribute. The following classifications are used by ECS:

- Minor degradation. The component (object) can continue to provide all the services that it supports, although there may be reduced performance. For example, a storage device may fail, but the service can continue using the remaining storage devices. Likewise if a couple of processors fail in a Symmetric Multiprocessor in the Processing Subsystem, other processors can be used to make up the loss.
- Major degradation. The component (object) is unable to provide some services but can still support other services. For example an instance of a Client is unable to send or receive messages due to a network problem, but can continue to provide local services.
- Loss of services. The component (object) is unable to provide any useful services. For example, if a critical COTS component fails, a loss of service may result.

A loss of service may need a common recovery procedure like restart, or remove failed component and integrate new component irrespective of whether the failed component is a disk, processor or network. Also, such a classification scheme could introduce a common display procedure that colors an icon in the M&O GUI a very dark shade of red for loss of service and a lighter shade of red for minor degradation. In summary, faults will be classified based on operations concepts and M&O perspective.

6.4.6.2.4 Fault Tolerance and Recovery.

Fault Tolerance is the ability of the system to continue to operate in the presence of faults. For example, if an archive drive fails, but the archive has two or more drives, the archive will continue to be operable, albeit at a reduced rate of throughput. When the system suffers a fault which reduces some operational system capability, the system is said to operate in “degraded mode”. When the original failure has been repaired and the system is returns to its normal operating capability, the system is said to have “recovered.

There are three main types of fault response:

- Continue to process the event despite the failure. The system may be changed to a downgraded mode until the cause of the fault is removed and the system can return to normal operation
- Abandon the event, perform any cleanup and regain control for handling new events
- Cause failure (Fail-stop - the application will either operate according to its specification or it will stop). No more events are processed until a recovery procedure (automatic or manual) is completed

ECS subsystems will be designed to continue to operate in degraded mode (rather than terminating their operation completely), whenever possible. It is also an ECS design goal to have subsystems recover automatically to their normal operational capability when the original fault is repaired. Where the application cannot be able to recover on its own, the MSS Fault Manager will launch the designated recovery procedure.

6.4.6.2.5 Error Handling

All errors will be reported to MSS via the event reporting interfaces. The reported information will include the error identification, an error message, and any addition attributes which are needed to describe the detailed nature of the error (e.g., for a file open error, the type and name of the file which could not be opened

6.4.6.2.4.6 Error Detection

An important aspect of errors is the method of their detection. Within ECS, methods of detection are categorized into Detector Classes. Detector classes are derived by examining each input to the application and enumerating the types of error detection which are possible for those inputs (some error detectors may not be dependent on any specific input). The following is a basic list of detector classes:

- Response time exceeded - many device access services monitor the health of devices by using timers. To avoid redundant error detection code, individual service implementations should be examined to see if this check is done below the application level.
- Capacity exceeded - applies to all storage including datasets, queues, heaps, etc.
- Missed message - for point-to-point messages, the lower level communication protocols will check sequence numbers, guaranteeing that the messages are delivered in order to the application. This is not the case for broadcast messages.
- Faulty/Unexpected input - the input can be checked for syntax and semantic errors without reference to state data
- Duplicate message - this happens when the original component sent a message to another component and failed after sending the message, the backup component has resent the same message. The receiving component will give this error.
- Input/state inconsistency - the input, when checked against state data is found to be semantically incorrect
- Faulty state or logic - for example: directory corruption; or previously committed state data now looks bad, often due to inconsistencies among data sets, an unplanned execution flow.
- Negative response - a peer or lower level service returns an "unwilling" or "unable" response
- Heartbeat - A solicited or unsolicited periodic message was not received. Generally this occurs with external interfaces and attachments
- Stop (relayed from part of the environment) - PGE failure

A stop can mean one of two things: a) the address space stops, or b) the processing of the particular event stops. When this behavior is desired, the failure of a particular event should not be allowed to affect other events (for e.g., partial database updates should be rolled back, or the particular database should be marked as unusable)

- Unknown ("unknown" should never be explicitly reported; it exists to account for bugs in error handling logic that leave this field unspecified)

6.4.6.3 List of Errors and Faults

Table 6.4.6-1 illustrates major errors in subsystems and categorization according to the scheme outlined above. The detector class is used to frame a message string to describe the error message.

6.4.6.11 Rules for Error, Exception and Fault Handling

The rules are classified as general, application domain specific error handling (where MSS is not involved other than just logging the event), and MSS Fault Management through change of state of Managed Objects. The following is a strawman list of rules for subsystem designers and developers:

General rules

- The definitions provided here will form the basis for Event, Error, Exception and Fault handling
- Every exception and fault must have a recovery procedure or a handler to recover
- The public class GLEvent must be used for all event handling
- Exceptions must be handled locally using an exception handler. Fault notification and logging must be handled through interface class provided by MSS Fault Management services (MsEvent)
- Any recovery that cannot be handled locally must be handled through interface class provided by MSS Fault Management services
- All objects within a failed component must recover to a defined state
- When recovering from a fault, the recovery procedure must not produce the same fault
- When recovering from an exception, only if the exception is handled partially, the same exception may be raised to be caught at the next higher level by another exception handler
- A catch-all exception handler will be provided to catch any exception. However, the catch-all exception handler will not provide information on the type of exception. The catch-all handler will be used to confine exceptions from propagating beyond a certain domain.
- Faults should be classified from an M&O perspective: a) minor degradation, b) major degradation, c) loss of services

Table 6.4.6-1 Strawman List of Errors/Faults for SDSRV

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low,med)	CI to be reported (including, MSS and M&O)	Detector class
Initialization File/ Environment Corrupt	Error	This would be seen during a system startup process and would result in one or more executables not starting	SDSRV	SDSRV	M&O staff will perform recovery procedures	Medium	MSS and M&O	Faulty state or logic
DB Transaction Log Full	Error	This condition would result in the inability to insert additional persistent data into the database engine(s) used with the Data Server	SDSRV	SDSRV	Operations DBA would need to dump the transaction log and restore prior operation of the affected database	High	MSS and M&O	Capacity exceeded
DB Connection Dropped	Fault	This could be a serious failure with the database or a short-lived problem with the connection	SDSRV	SDSRV	Operations DBA would need to evaluate the problem, possibly restarting the database and data server processes	Fatal	MSS and M&O	
Unable to establish link to/invoke DLL	Error	DLLs are invoked for newly added data type services.	SDSRV	SDSRV	Operations must issue a trouble ticket and have the problem analyzed and resolved	High	MSS and M&O	Negative response
Internal queue overflow	Error	This is a result of a poorly tuned system. Potential loss of service requests could result.	SDSRV	SDSRV	Operations staff would need to immediately throttle back system processing thresholds for requests. Operations staff would analyze system off-line and tune	High	MSS and M&O	Capacity exceeded
Unable to allocate disk space	Error	Unable to allocate working storage space using STMGT CSCI.	SDSRV	SDSRV	Alert operations staff who would immediately lower system thresholds for requests. Operations staff would analyze system off-line and tune.	High	MSS and M&O	Capacity exceeded

Table 6.4.6-2 Strawman List of Errors/Faults for DDSRV

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Unable to allocate disk space	Error	Unable to allocate working storage space using STMGT CSCI.	DDSRV	DDSRV	Alert operations staff who would immediately lower system thresholds for requests. Operations staff would analyze system off-line and tune.	High	MSS and M&O	Capacity exceeded
Initialization File/ Environment Corrupt	Error	This would be seen during a system startup process and would result in one or more executables not starting	DDSRV	DDSRV	M&O staff will perform recovery procedures	Medium	MSS and M&O	Faulty state or logic

Table 6.4.6-3 Strawman List of Errors/Faults for DDIST (1 of 2)

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Unable to allocate disk space	Error	Unable to allocate working storage space using STMGT CSCI.	DDIST	DDIST	Alert operations staff who would immediately lower system thresholds for requests. Operations staff would analyze system off-line and tune.	High	MSS and M&O	Capacity exceeded
Internal queue overflow	Error	This is a result of a poorly tuned system. Potential loss of service requests could result.	DDIST	DDIST	Operations staff would need to immediately throttle back system processing thresholds for requests. Operations staff would analyze system off-line and tune	High	MSS and M&O	Capacity exceeded
Unable To Allocate Distribution Device	Error	Unable to allocate a media distribution device due to device off-line, device powered-off, or device in use by another process.	DDIST	DDIST	Report alert to operations staff who would check device status, perform corrective action or notify maintenance.	Medium	MSS and M&O	Faulty state or logic

Table 6.4.6-3 Strawman List of Errors/Faults for DDIST (2 of 2)

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Initialization File/ Environment Corrupt	Error	This would be seen during a system startup process and would result in one or more executables not starting	DDIST	DDIST	M&O staff will perform recovery procedures	Medium	MSS and M&O	Faulty state or logic

Table 6.4.6-4 Strawman List of Errors/Faults for STMGT (1 of 4)

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Parity Error	Error		Several CI Components may experience this error (e.g., Tape, CD-ROM, Staging Disk)	Several CI Components may detect this error	STMGT	low if recoverable/fatal if unrecoverable	MSS	Faulty/Unexpected input
Configuration file corrupted or deleted	Error	The Configuration file has been deleted or corrupted	DsStResourcePolicy	DsStResourcePolicy	Manual recovery procedures will be defined for M&O	show stopper until operators have restored or recreated file from last snapshot backup	MSS and M&O	Faulty state or logic
Corrupted schedule	Error (possibly Fault)	Schedule becomes corrupted	DsStResourceSchedule	DsStResourceSchedule	Manual recovery procedures will be defined for M&O	Show stopper until schedule is restored from snapshot backup	MSS and M&O	Faulty state or logic

Table 6.4.6-4 Strawman List of Errors/Faults for STMGT (2 of 4)

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Initialization file corrupted	Error	Initialization files or files used at startup time have been corrupted	DsStResourcePolicy, DsStResourceSchedule	DsStResourcePolicy, DsStResourceSchedule	Manual recovery procedures will be defined for M&O	Low if recoverable, show stopper until backup have been restored	MSS and M&O	Faulty state or logic
Insufficient staging disk space	Error	All available disk space is in use (hard limit for request has been reached).	DsStStagingDisk	DsStStagingDisk	STMGT; Request will be terminated	Low severity, for subsequent requests for staging disk	MSS and M&O	Capacity exceeded
Queue full	Error	Queue is full when a request comes in	DsStResourceQueue	DsStResourceQueue	Manual recovery procedures will be defined for M&O; request must be resubmitted	High severity	MSS, M&O, SDSVR, DDIST, INGS T	Capacity exceeded
User Pull Area full	Error	User pull area is already filled with files waiting to be pulled and no files can currently be deleted	DsStPullMonitor	DsStPullMonitor	M & O personnel should reevaluate and reset parameters associated with use of the data pull area.	Low severity, will probably occur when system is new. Subsequent pull data requests will be delayed.	MSS, M&O, DDIST	Capacity exceeded
Staging List corrupted	Error (Possibly Fault)	The file containing the staging list has been corrupted	DsStStagingDataList	DsStStagingDataList	DsStStagingDataList,	Low severity if recoverable, fatal if not recoverable	always MSS and M&O; If Fault, include SDSVR, DDIST, INGST	Faulty state or logic

Table 6.4.6-4 Strawman List of Errors/Faults for STMGT (3 of 4)

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
FSMS failure	Fault	File open error/file not found. The filename has been corrupted and there is actually a request for an unknown file	DsStArchive	DsStArchive	Science Data Server, takes care of problem if there are more than 3 errors for a file request	Low if recoverable, high if files are lost and have to be recreated from other source	always MSS, M&O,SDSV R	
Printer Failure	Fault	Printer error during write operation. Caused by hardware failure	DsStPrinter	DsStPrinter	Operator and job requester will be informed. Operator response needed	Fatal to request	MSS, M & O, DDIST	
CDROM drive failure	Fault	Hardware failure in the physical drive unit	DsStCDROM	DsStCDROM	Manual operator intervention and/or remedial maintenance required	Fatal to request	MSS and M&O, DDIST	
CDROM Media failure	Fault	Physical media cannot be written to or has physical damage	DsStCDROM	DsStCDROM	Media must be replaced, manual operator intervention required	Fatal to request	MSS and M&O, DDIST	
Fax failure	Fault	Fax machine/board/hardware breaks down	DsStFax	DsStFax	Manual operator intervention and/or remedial maintenance required	Fatal to request	MSS and M&O, DDIST	

Table 6.4.6-4 Strawman List of Errors/Faults for STMGT (4 of 4)

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Network failure	Fault	Network communication failure; No traffic gets through, or data corrupted	DsStNetwork	DsStNetwork	Manual recovery, needs maintenance intervention.	Fatal to request	M&O and MSS, DDIST, INGST	
Checksum error	Error	This error is created during a file retrieve operation. When data files are retrieved from the deep archive a checksum is performed and compared to the checksum performed when the data was archived and the result stored as file metadata.	DsStArchive	DsStArchive	DsStArchive	always fatal to retrieval request	MSS , M&O, DDIST, PLAN G, SDSVR	Input/State inconsistency

Table 6.4.6-5 Strawman List of Errors/Faults for GTWAY (1 of 2)

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Invalid ODL message	Error	This error is created if V0 client sends an ODL message which has invalid syntax or invalid groups that are not known to the system	Request processing module	Request processing module	Request processing module	low	Version 0 Client	Faulty/Unexpected input
Invalid browse request	Error	This error is created when an invalid browse parameters are received.	Request processing module	Request processing module	GTWAY	low	Version 0 Client	Faulty/Unexpected input
Invalid product order	Error	This error is created when the product order does not contain all necessary information to fulfill the order	Request processing module	Request processing module	GTWAY	low	Version 0 Client	Faulty/Unexpected input
Incomplete schema export	Error	This error is generated when the data server exports information that is incomplete	Data Server Interface module		GTWAY	low	SDSRV, MSS	Faulty state or logic
Database execution error	Error	This error is generated whenever query has syntactical or semantical errors and limitations such as more than 252 values in IN clause.	Request processing, Mapping Service	Request processing, Mapping Service	GTWAY	low if recoverable/fatal if unrecoverable	GTWAY. If unrecoverable MSS and M&O will be notified	Faulty/Unexpected input

Table 6.4.6-5 Strawman List of Errors/Faults for GTWAY (2 of 2)

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Database update error	Error	This error is generated during updates to the database if the data input format is not correct	Request processing, mapping service	Request processing, mapping service	GTWAY	low if recoverable/fatal if unrecoverable		Faulty/Unexpected input
Database access error	Exception	This error is generated due to the corruption of the page allocation or corruption of the database	Request processing module, Mapping service	Request processing module, Mapping service	M&O can restart the DBMS server or restore the database if necessary	low if recoverable/fatal if unrecoverable	M&O, MSS	Faulty state or logic
Failure to connect to gateway DBMS server	Exception	This error is generated if incorrect login parameters are used or if the DBMS server is not running	Request processing module	Request processing module	M&O can start the server in case of a server crash. GTWAY is responsible for correcting the login errors.	low if recoverable/fatal if unrecoverable	Version 0 Client, M&O, MSS	Negative response
Failure to allocate memory	Exception	This error is generated whenever there is not enough memory for dynamic memory allocation	All	All	DMGHW	fatal	MSS, M&O	Capacity exceeded
Database lock table full	Exception	This error can occur when there is high traffic in the system	Request processing	Request Processing	Request Processing , M&O can increase the size of the lock table entries	low	Version 0 Client, MSS, M&O	Capacity exceeded
Database media failure	Fault		DMGHW	DMGHW	Recovery as defined by M&O	Fatal	MSS, M&O	Negative response

Table 6.4.6-6 Strawman List of Errors/Faults for DSKTP

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Invalid Username	Error	This error is created when the user enters a wrong username in the Login screen	Desktop	Desktop	Desktop	low	MSS	Faulty/Unexpected input
Correct Username & Invalid Password	Error	This error is created when the user enters a correct username & wrong password in the Login screen	Desktop	Desktop	Desktop	low	MSS	Faulty/Unexpected input
Error in Drag & Drop	Error	This error occurs while the drop targets are not of correct type.	Desktop	Desktop	Desktop	low		Faulty/Unexpected input
Server down	Fault	This error will occur when client does not execute properly if the server is down	Data Server/LIM/DIM	Data Server/LIM/DIM	Desktop	fatal	MSS	Negative response
Process creation error	Fault	Cannot spawn a process for Desktop Application due to error in the Operating System, or NW	Desktop/CSS	Desktop	Desktop	fatal	MSS	Negative response

Table 6.4.6-7 Strawman List of Errors/Faults for PRONG and PLANG (1 of 2)

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
1. Sybase read error	Error	This error is created during a data read from the database. It is generated as a result of improper format, invalid input, etc.	Data Manager, PGE Execution Manager, or COTS Component	Data Manager, PGE Execution Manager, or COTS Component	(Depends on severity) Data Manager, PGE Execution Manager, or COTS Component	low if recoverable/fatal if unrecoverable	COTS. If unrecoverable, subsystem manager will generate fault and notify MSS and M&O	Faulty/Unexpected input
2. Sybase write error	Error	This error is created during a data write operation to the database. It may a result of improper format, invalid input data, etc.	Data Manager, PGE Execution Manager, or COTS Component	Data Manager, PGE Execution Manager, or COTS Component	(Depends on severity) Data Manager, PGE Execution Manager, or COTS Component	low if recoverable/fatal if unrecoverable	COTS. If unrecoverable, subsystem manager will generate fault and notify MSS and M&O	Faulty/Unexpected input
3. Sybase index error	Error	This error is generated as a result of a duplicate key supplied	Data Manager, PGE Execution Manager, or COTS Component	Data Manager, PGE Execution Manager, or COTS Component	(Depends on severity) Data Manager, PGE Execution Manager, or COTS Component	low if recoverable/fatal if unrecoverable	COTS. If unrecoverable, subsystem manager will generate fault and notify MSS and M&O	Faulty/Unexpected input
PGE Failure	Error	A PGE terminated abnormally	PGE	COTS	COTS		Log info, Perform canned recovery activities	Stop

**Table 6.4.6-7 Strawman List of Errors/Faults
for PRONG and PLANG (2 of 2)**

Identify Error/fault	Fault/Error (specify error or fault)	Description of the error/fault	CI Component where error/fault occurs	CI component where the error/fault is detected	CSCI/HWCI responsible for recovery action	Severity (fatal, warning, high, low, medium)	CI to be reported (including, MSS and M&O)	Detector class
Data staging/destaging (data insert into Data server was unsuccessful)	Error	Data Acquire did not work. Could be a result of lots of causes. Data Server does not have data. Communication problems, etc.	Data Manager	Data Manager	COTS		Log Info, Inform MSS, M&O	Negative response
1. Planning DBMS server failure	Fault		PLNHW	PLNHW	Manual recovery procedures will be defined for M&O	Fatal	MSS, M&O	
2. Planning DBMS failure	Fault		PLNHW DBMS	PLNHW DBMS	Manual recovery procedures will be defined for M&O	Fatal	MSS, M&O	

Rules for failure handling when application domain specific (Only MSS logging and MSS Fault Manager not involved)

- Propagating errors among various subsystem components must be accomplished through Public class GLError
- All errors must be logged in real time or near real time using the MSS logging services
- Handling of exceptions and errors must be in accordance with the Software Development Plan Project Instructions
- All events (errors, exceptions and faults) must have parent-child identifiers (UUIDs generated by CSS provided class) for unique identification. An event at the lowest layer generates and returns a UUID to the caller, after logging the message to the local history log. The caller generates another UUID, and passes the received UUID as well to its caller, after logging the event to the log. This propagates all the way up the procedure call tree. This ensures the ability to correlate events since a management application can then analyze the notifications and determine what notifications caused what.

Rules when using MSS services for Fault handling

- Every fault must have a recovery procedure that will be handled through the interface class by MSS
- Recommended severity level of errors, exceptions and faults must be provided to MSS (framework will be provided by the interface team)
- Although recovery from faults is not fully automatic, hooks must be provided for building automatic recovery for future releases
- MSS will notify all dependents of a failure due to a fault activation through its notification feature
- The MSS notification and call back features must be used for all recoveries that cannot be handled locally after a failure. The MSS will launch the recovery procedure and notify all dependents of the failure.
- All dependencies for exception and error propagation and notification for each subsystem component must be provided by the CI designer to MSS. Only the first level of dependency need to be provided. The MSS will maintain the dependency tree and do the appropriate notification and launching of a recovery method.

Other proposed rules/guidelines

- There will be a single operational log. There will be a capability to route outputs of test operation to specific, separate log files.
- There will be a generic browse and viewing capability for the operational log which will be available at all operator stations
- An object which receives an error return from another object should be able to assume that the particular error has been previously logged by an object lower in the calling chain
- If the error cannot be handled, but also leads to an error within that particular object, the object should log the error occurrence only if it is fair to assume that the object can add informational value to the log. Otherwise, the object should refrain from logging the error.
- The generic error object shall be dynamically configurable to log or not log certain errors or types of errors for a given application (managed object).

6.4.6.12 Summary

A general infrastructure and framework is presented for the uniform handling of events, errors, faults and exceptions for ECS. A classification scheme is outlined and major subsystem errors are identified and categorized. More errors and faults will be identified during the implementation phase of the detailed design.

6.4.7 Management and Operations Reporting

6.4.7.1 Overview

Operations personnel and management need to be able obtain reports about various categories of system performance. This is a very important aspect of overall system management. Many of these reports are generated in an ad-hoc fashion, usually in response to some problem or concern. For example, operations may have noticed a recent slip in the average time needed to fill orders for a certain product collection. To investigate the cause, operations may run various statistics to find patterns which may indicate the source of the problem. The ability to produce such reports is, therefore, an important aspect of the system management architecture.

Table 6.4-2 identifies over 20 different Management and Operations (M&O) Positions, ranging from operators and technicians which support a particular ECS subsystem (e.g., data processing, or ingest, or database operation) to user support staff, to capacity planning and DAAC management. Each of these operator positions will have unique reporting needs, and they will change with time as old problems get resolved and new ones arise.

In response, ECS provides three levels of reporting and two methods of reporting. The two reporting levels are:

- reports which are specific to the operation and management of a subsystem and are generated by that specific subsystem - for example, the physical allocation of database resources are reported directly by the DBMS for use by a database administrator;
- site level reports which integrate information across several subsystems or provide mechanisms for obtaining long terms statistics about the ECS - for example, regular statistics about the various categories of data production and distribution; and
- system level reports which integrate information across ECS sites, allow comparison of ECS operation at different sites, or support end-to-end tracking and reporting of inter-site activities.

The two reporting methods are:

- ad-hoc reports - ECS will provide tools to allow operations personnel to create reports in an ad-hoc fashion, direct them to a printing device or a file, and save such reports for repeated execution.
- canned reports - operations will have a number of predefined reports which will be executed regularly, and through their common formats allow comparison and long term trend analysis.

The three levels of reporting correspond to the three levels of logging which are performed within ECS and which were described in Section 6.4.4. Subsystems perform their own logging and performance tracking to support the personnel directly involved in operating that subsystem. For example, data processing events are logged within the Planning and Processing database, and that database can also provide information about the current processing status for each group of data products. Concurrently, processing events are reported via an MSS interface to the management subsystem. The logs generated at each host are consolidated on a regular basis with a configurable

time constant (which typically would be in the order of 15-30 minutes) into a database of site-related system events. Also on a regular, configurable basis, excerpts and aggregates of this information are sent to the SMC where they are stored in a database for reporting and tracking of activities and requests which span sites.

The mechanisms offered for reporting at the subsystem level depend on the specific configuration of each subsystem. Release A subsystems provide the following off-the-shelf capabilities for this purpose. Most of these components support regular reporting through saved reports, in addition to ad-hoc reporting:

- Sybase database administration and reporting writing tools
- Autosys and Autoexpert reporting capabilities
- ClearCase and other configuration management tools
- HP OpenView reports
- Fault and performance management applications (the product selection process in this area is still on-going)

Site and system level information is collected in a Sybase DBMS, and reporting is provided via the Sybase report writing tools and Sybase queries.

6.4.7.2 Report Data

There is a general requirement to be able to reconstruct the performance of the ECS, its operators, and its users as part of performance analysis, error debugging, and testing.

All logged items will be identified by date and time so that the sequence of events can be reconstructed. The history log will be organized such that the information can be selectively retrieved based on logical combinations of criteria such as: user, order, platform, instrument, data product, data product type (e.g., standard, interim), production mode (i.e., routine, on-request), operating mode (i.e., live vs. reprocessing vs. test), production string, Subsystem/CI/CSC/subroutine/object, interface source and destination, and specific logging IDs.

Typical information logged in a history log database includes:

- operations data,
- resource data, and
- user statistics.

They are briefly discussed in the following paragraphs.

6.4.7.2.1 Operations Data

Logging of operations data is intended to establish an electronic record of key data, inputs, outputs, and faults. Suggested classes of logging are:

Initialization Data. Each ECS service will log all initialization/control files, values, and resource status each time it initializes.

Data Transfers.	Source/destination, data identification/type, media type, and size of all data transfers to/from each application subsystem, including external and internal ECS transfers. Logging will provide enough information to associate each event with the original stimulus (i.e., a particular user's request). Logging will unambiguously identify the ECS source of the data within the subsystem, and the data destination (e.g., another ECS service, a user, or an external system). Network addresses will be included where appropriate.
Human Actions	Source (i.e., operator, analyst, user services, etc., and user) and values of human operator inputs which request changes to the state or configuration of a service.
Fault Messages	All faults will be logged (see Section 4.??). Contents, and destination of all operator and user alarms, alerts and notifications will be logged.
Intermediate Results	ECS application services will be able to log selected intermediate results of processing or activities that provide insight into the operation of the CI. This type of logging will be configurable and can be turned on or off by operations.
Product Generation	Processing history and statistics will be kept and will include information on PGE execution times, comparison to plan, the number and timing of replans, information on the data products created, PGE resource utilization, and QA attributes of the produced data.

6.4.7.2.2 Resource Utilization Data

ECS will monitor all computational, storage, archival, ingest, network and distribution resources and be able to construct short term resource utilization reports (e.g., reflecting current throughput and resource utilization rates) and long term trending (i.e., resource utilization over longer reporting periods).

CPU	CPU utilization
Computer memory	Memory utilization
Local storage	Local disk/mass storage utilization
Archives	Archive utilization (by file and science product, number of bytes in use, number of bytes in use, I/O, number and size of retrievals, number and size of storage actions)
Hard media	Hard media distribution utilization (i.e., number of media by type produced, number of bytes produced).
LANs	LAN (including gateways, routers and bridges) loading and performance.
WANs	WAN traffic in and out of ECS.

6.4.7.2.3 User Data

User satisfaction will be a key metric for ECS. ECS will be able to track an order from the time it is placed until the order is fulfilled, either electronically or via hard media. ECS will collect statistics on volume by distribution type, timeliness (elapsed time from order receipt to data ready to data delivered), quality (did the user receive what was ordered), and resources utilized in servicing users. Data products will be logged to at least the granule level. ECS will be able to provide information about:

Order development	Number and nature of data product requests (e.g., if product A is ordered, product B is always ordered too); mechanism used for placing an order (electronic or through User Services); amount of data requested.
Order processing	Size and development of the order queue; elapsed time needed to respond to the various types of orders.
Order delivery	Size of the distribution queue; data products and files included in the order; size of order; distribution mechanism (i.e., electronic, numbers and types of hard media); time and resource utilization for delivery (i.e., awaiting pickup or media creation); elapsed time from placement of order; elapsed time to deliver the order; order fulfillment success rate.
Order satisfaction	Incomplete orders; rejected orders; orders never picked up; orders never shipped.
Order resources	System resources utilized by ordering, and distribution over orders.

6.4.7.2.4 Security Data

Security relevant events and statistics which are of interest to security analyses will also be collected, including:

Security Violations	Attempted connections, requests for services which failed due to a lack of authorization, and other types of attempted security breaches will be reported.
Unusual Patterns	The information collected for system monitoring purposes can also be used to examine logged events for patterns of activities which might indicate attempts to compromise ECS security.

6.4.7.3 Reporting Capabilities

Although ECS will include a set of canned reports, it was decided not to make these reports part of the formal system design. The decision is based on the following considerations:

- Canned reports generally include less detail and more aggregates and statistics than ad-hoc reports. Other than that, there is little difference between ad-hoc and canned reports. In general, a canned report is a version of an ad-hoc report, saved (and perhaps

parameterized) for future execution. ECS operations personnel will be fully qualified to create ad-hoc reports, and tailor any canned reports which ECS initially provides over time to their needs.

- Making the canned reports part of the formal system design would place them under formal configuration control. This would make it much more difficult for system operations to change the regular reporting to suite their needs based on their experience with the operation of the ECS.

The reporting capabilities provided by the individual ECS subsystems are discussed in the individual subsystem design documents. However, it was felt that from an M&O perspective, the topic truly crosses subsystem boundaries, and that an integrated presentation is needed. Therefore, the remainder of this section provides an overview of the reporting capabilities which Release A will provide. The capabilities are organized into four functional areas:

- Fault Management
- Performance Management
- Accountability Management
- Security Management, and
- Configuration Management

For each of these area, a table is provided listing various M&O topics and the corresponding reporting capabilities. As indicated above, these reporting capabilities will be developed in an incremental fashion involving the ECS M&O organization as well as the DAACs, but ECS will develop models for these reports to facilitate their further development by M&O.

6.4.7.3.1 Fault Management Reporting

HP OpenView is the ECS enterprise management framework. Management data for all devices that are monitored via SNMP agents is collected and can be displayed by HP OpenView. The operator can select one or more icons representing the managed object(s) for which reports are to be generated. HP OpenView provides both standard reports for real-time monitoring of network devices and ad hoc reports for real time and near-real time monitoring of any management information defined in the loaded MIBs.

Fault Management reports will include summary and detailed reports on managed objects' faults, i.e., hardware, software, and network faults occurring at each site. Reports may be generated to cover all or portions of the data captured in the database for variable amounts of time. For example, the ground resource fault and maintenance reports generated by LSM at each site may include fault type and description, time of occurrence of fault, effect on system, fault resolution, fault statistics, etc., or any other parameters required.

HP OpenView provides four predefined reports for real-time fault management reporting. The Ethernet Errors and SNMP Errors reports provide statistical information about errors that occur from the time that the operator selects the report for operator-specified managed object(s). The SNMP Authentication Failures report and the SNMP Events Log provide reports of past events

that have been logged by HP OpenView, again for the operator-specified nodes (although the SNMP Events Log can also be reported for the entire system).

The majority of fault reports will be generated on an ad hoc basis. The ad-hoc reports can be generated on as needed basis by HP OpenView, the fault/performance management application, the report generation application, and the trouble ticketing application to include detailed and summary information on the fault management of ground resources as required by the operator.

6.4.7.3.2 Performance Management Reporting

Performance Management has a number of different aspects. It includes the performance aspects of the various system resources, as well as the performance of ECS services with respect to their service requests. The reporting capabilities are provided by ECS performance management tools (which are still in procurement), ECS COTS tools used by the various services, and Sybase report writing tools (to extract performance data from the information in the site and SMC history log databases).

6.4.7.3.3 User Services and Accountability Reporting

These reports will be based on accountability data stored in the Sybase management database and will be generated using the Sybase report writing tool. The purpose of these reports is to track and profile ECS usage by external (e.g., science) users. The purpose of these reports is to identify usage trends to identify service needs/shortfalls and assist in the planning of future capacity needs. For Release B, this reporting area will include accounting reports.

6.4.7.3.4 Security Management Reports

Security management application report on security events for its management domain, i.e., a given site (for LSM), or the entire system (for EMC). The security management application is currently in the procurement phase (no products have been selected yet). At a minimum, the products that will be selected will provide the capability to store security management data in the management database, from which the Sybase report generator will be capable of creating standard and ad hoc security management reports. The security management application may or may not provide separate reporting capabilities. In addition, virus detection and other security reports will be provided via an ad hoc reporting capability and generated on an as-needed basis

6.4.7.3.5 Configuration Management Reporting

The Configuration Management applications (Baseline Manager, Software Change Manager, and Change Request Manager) will be COTS products and will provide functionality which will support the management of ECS resources. These products will track what constitutes ECS baselines; make available functional and physical characteristics data needed to operate and maintain the system; aid in managing system requirements and changes; and provide report generation capabilities.

Custom reports generation will be supported by the built-in capabilities of the ClearCase and other CM COTS.

Table 6.4.7-1. Fault Management Reports

Report Topic	Report Contents
Ethernet Errors	Available only for nodes using the HP-UX MIB. Provides a graphical representation of real-time Ethernet errors detected on operator-selected nodes.
SNMP Errors	This report provides a graphical representation of the SNMP protocol errors detected by the SNMP agent on the operator-selected node(s). The operator has various options for changing the view of the graph and options for printing or saving the report.
SNMP Authentication Failures	This report lists the management systems that caused an authentication failure on the operator-selected node(s). An authentication failure occurs whenever the management system sends an SNMP request with an invalid community name. The operator has the option of sorting the report and save or print the report.
SNMP Events Log	This log provides a listing of all SNMP events reported to HP OpenView for the operator-selected node(s). The logged events are presented in chronological order, with the most recent events presented at the bottom of the list. The operator can set filters for the events to be displayed in the log, clear filters, view a description for a selected event in the log, delete events from the log report, sort events, print events, or save the log report to file.
Site Host Errors	<p>This report provides a summary of the types of errors logged for each host at the site over an operator-specified period of time. The report will be generated by an OTS fault management application (procurement in progress). Reported faults include</p> <ul style="list-style-type: none"> • Performance Degradation (as measured against thresholds) • CPU errors • Memory errors • Disk errors • File system errors • Archive errors • Network errors • Queue errors • Critical processes missing • Processes looping • Processes failed • Cron jobs missing • NFS errors
EMC Host Errors	The EMC Fault Errors report will be identical to the Site Fault Errors report, except that the report will summarize the type of errors logged at each site instead of the number of errors logged on each host at the site.
Trouble Status	<p>This report provides a summary of the problems reported at the site for which trouble tickets have been opened, including:</p> <ul style="list-style-type: none"> • Number of trouble tickets opened • Number of trouble tickets closed • Number of trouble tickets remaining open at end of time frame • Average length of time between trouble ticket opening and closure

Table 6.4.7-2. Performance Management Reports (1 of 4)

Report Topic	Report Contents
System Capacity Utilization	
CPU Load Report	The CPU load report provides a graphical representation of the average number of jobs in the run queue for each selected node. The graph plots the average number of jobs (over the past 1, 5, and 15 minutes) starting from the time that the report is selected, and continues to chart the data as long as the report window is open or iconified. If the window is left open (or is opened after being left iconified), performance trends for the selected host(s) will begin to emerge. This report is only available for HP hosts using the HP-UX MIB.
Interface Traffic	This report provides a graphical representation of packet statistics for operator-selected SNMP node, including the total number of incoming and outgoing packets, and the corresponding counts of packets with errors.
Ethernet Traffic	This report, available only for nodes being monitored by the HP OpenView SNMP agent, provides a graphical representation of ethernet packet statistics for operator-selected nodes.
SNMP Traffic	This report provides a graphical representation of incoming and outgoing SNMP traffic on operator-selected nodes. It is a useful tool for determining the percentage of traffic to and from the selected node(s) that is due to the SNMP management of the node(s).
SNMP Operations	This report provides a graphical summary of the SNMP operations requested of and performed by the SNMP agent on the selected node(s). The graph provides a display of the number per second for each specific type of SNMP operation that has occurred since the report was started.
SNMP Event Log	Provides a list of SNMP resource utilization events for selected managed object(s). These events will generally represent occasions when resource utilization thresholds are exceeded. This report can be filtered to only allow the display of threshold events.
Site Host Resource Utilization	<p>The site host resource utilization report provides an overview of the host utilization over the time frame specified by the operator. The overview includes a look at minimum, maximum, and average values over the specified time frame.</p> <p>For each host at the site, the report will list minimum, maximum, and average values for the following performance metrics:</p> <ul style="list-style-type: none"> • Percentage of physical memory utilized • Percentage of CPU time utilized • Rate of physical disk I/O's • Rate of logical disk I/O's • Rate of packets sent (per second) • Rate of packets received (per second) • Rate of LAN collisions (per minute) • Rate of LAN errors (per minute) • Number of users • Number of processes running • Rate of process swaps (per minute) • Rate of system calls (per second) • Number of processes in run queue

Table 6.4.7-2. Performance Management Reports (2 of 4)

Report Topic	Report Contents
EMC Host Resource Utilization	The EMC Host Resource Utilization report will be identical to the Site Host Resource Utilization Report except that resource utilization information will be provided for all hosts at all reporting sites over the operator-specified sites and time frames.
Resource Capacity Planning Reports	<p>These reports will be used to detect the need for required system capacity upgrades. HP OpenView provides disk space reports for systems using the HP-UX MIB. Other resource capacity planning reports will be provided by the fault/performance management application on an ad hoc basis.</p> <p>As an example, the disk space report provides:</p> <ul style="list-style-type: none"> • Name of the file system • Total number of kilobytes of disk space • Number of kilobytes used • Number of kilobytes available • Percent of total capacity used • Directory name on which the file system is mounted
Storage Management Utilization and Performance	
Storage Management Performance Report	The report provides information about archive activities, including archive read and write statistics in terms of staging and destaging requests, file transfers, data volume, and I/O throughput.
Storage Utilization	The report provides information about archive storage resource utilization and data growth rate.
User Services Utilization	
User Service Performance	<p>User service feedback information will be received either electronically (through a user screen or through e-mail), by telephone or by surface mail. User feedback summary information received from any source is captured using a trouble ticketing application, or by using office automation tools. The user service performance report will be generated periodically to provide user feedback statistics and include:</p> <ul style="list-style-type: none"> • Number of complaints received per reporting period by method of receipt (e-mail, fax, letter, phone, in person) • Number of complaints as % of the total number of orders or requests per month • Number of kudos received per month • Number of kudos as a % of total orders or requests received per month

Table 6.4.7-2. Performance Management Reports (3 of 4)

Report Topic	Report Contents
Order Processing and Distribution Performance	
Order and Distribution Performance - Detail and Summaries	The report provides detail and summary information to analyze the order turn-around time. Any associated delays, if any, can be investigated and corrected to improve performance. The report lists information about each data distribution request (e.g., user, data type, times of key events in the distribution event flow, and any quality data that may be available). The summary portion provides totals and averages for the reporting period of key performance characteristics (e.g., total order volume, average elapsed time for each step in the event flow) as a whole, as well as daily and weekly totals and averages. The report also provides information on current and backlog trend.
Media Distribution Profile	This report provides statistics on methods of data distribution for a specified reporting period, such as Electronic Distribution via various transfer mechanisms, and hard media distribution by media type.
Data Set Order History	This report provides order performance and volume statistics for each product group (e.g., dataset or collection) for the reporting period, and lists the most frequently ordered products.
Order and Distribution Error Summary	The report provides summaries of error occurrences during the reporting period, grouped and sorted by type of error, type of product, and type of distribution media.
Returned Product Summary	This report provides statistics on all products returned by users for the reporting period including product type, reason for return, action, and status.
Ingest Performance	
Ingest Status	Provides operations staff with the capability to view status on ongoing ingest processing. Operations staff may view a specific request (identified by request ID), all ongoing requests entered by a specified user/external data provider (e.g., TSDIS), or all ongoing requests. Displayed status includes the external data provider, ingest request ID, total ingest data volume, and current request state (e.g., "data transferred").
Ingest Request History	<p>The report supplies operations staff with a view of ingest request completion performance. The report can be generated for specified time periods and executed on a regular basis. It provides a detailed log of the ingest requests in the reporting period (including requester, data source, data type, the times of various ingest events such as request receipt and completion, data volume, etc.).</p> <p>The report also provides summary statistics for the reporting period, such as completed vs. unsuccessful requests, backlog development, average ingest volumes and processing times broken down by various categories.</p>

Table 6.4.7-2. Performance Management Reports (4 of 4)

Report Topic	Report Contents
Ingest Data Set History	<p>The report supplies operations staff with a view of ingest operation by data type (it is specially sorted and summarized version of the ingest request history report).</p> <p>The report can be generated for specified time periods and executed on a regular basis. It provides a detailed log of the ingest requests, broken down by data category (including requester, data source, data type, the times of various ingest events such as request receipt and completion, data volume, etc.), information about current status, and summary statistics similar to the request history report.</p>
Ingest Errors	<p>The Ingest Error Report is a summary report of the frequency of errors of different types encountered during ingest processing. The report consists of two sections--Data Set Summary and Error Class Summary. The Data Set Summary lists a count of reported errors, by error class, for each data set type. The Error Class Summary lists a count of reported errors for each error class.</p>
Data Processing Performance	
Processing History - Detail	<p>The report provides a detailed account of all data processing activities performed in the reporting period at the product group, product, and PGE level.</p>
Processing History - Summaries	<p>The report provides summaries of data processing activities and resource utilization at the product group, product, and PGE level. The summary include totals (e.g., the count of products produced), as well as averages and other aggregates (e.g., average, minimum and maximum run & CPU times vs. predicted run times).</p>
Production Workload and Turn-around Time	<p>The report provides statistics about production turn-around (i.e., elapsed time between submission of a data processing request and its completion), production delay (i.e., scheduled vs. actual completion time); and production workload (i.e., currently scheduled work).</p>
Production Status	<p>The report compares the active production plan with current processing status at the product group, product, and PGE level, and gives account of production backlogs.</p>
Production Errors	<p>The report provides a detailed list of all data processing errors which occurred in the reporting period, as well as summaries by product, product collection, and PGE.</p>
Quality Assurance Summary	<p>The report provides a detailed list of QA exceptions which occurred during the reporting period, and summary statistics of QA results for each product type.</p>

Table 6.4.7-3 User Services and Accountability Reports

Report Topic	Report Contents
User Characterization	This report will include information on ECS users identifying each user class, the number of users in each class, and a categorization of their interests, affiliation, and access patterns.
User System Access Profile	This report profiles user system accesses and time spent on the system to help analyze the efficient usage of the system resources and identify probable bottlenecks. The report will include statistics such as the following, in total and broken down by user class: <ul style="list-style-type: none"> • Number of Data Items Searched/Browsed • Number of Local On-Line System Accesses by External Users / DAAC Staff • Average Session Times • Number of separate user accounts which accessed the system • Number of Accesses by Access Mechanism (e.g., Version 0, http, ftp, BBS).
Utilization of User Services Personnel	This report provides the information on user contacts in handling any ECS related inquires with user services personnel, and includes statistics such as the following: <ul style="list-style-type: none"> • Number of contacts, by method (e.g., phone, e-mail, US mail, in person) • Number of User Inquires: Number of inquires for each of following, by topic (e.g., Information Requests, Data Order Inquiries, Data Usage Questions). • Number of DAAC / non-DAAC Referrals • Number of Non-DAAC Referrals • Number of Data Documentation Requests

Table 6.4.7-4. Security Management Reports

Report Topic	Report Contents
Security Compromises	Security violation statistics will be generated regularly to report any security violations or attempts of intrusions. This report will include the total number of attempts detected, a categorization by type of violation and method, by server, and by data category.

Table 6.4.7-5. Configuration Management Reports (1 of 3)

Report Topic	Report Contents
Configured Articles	The report is an indented list detailing the exact, as-built configuration of deployed, ECS configuration items. Identifies control items and associated assembly structures that comprise configuration items as operationally baselined at ECS sites.
Baselined Documents	The report is a list of the documents applicable to deployed system resources. Each document associated with specified baseline is identified, title, version, and date are provided, and the latest document change notice is referenced.

Table 6.4.7-5. Configuration Management Reports (2 of 3)

Report Topic	Report Contents
As-Built Resources	This report identifies individual control items and assemblies deployed system-wide, including their revision level.
Baseline Manager Profiles	<p>This report lists full or partial details about selected system resources (hardware, software, and assemblies), documents, and baselines. The provided information varies depending on the profile selected by the operator.</p> <ul style="list-style-type: none"> • For each baseline, the report lists site, version, type, associated release, approval date, effective date, prior version, and status. • For each configuration item, the report lists identifier, name, description, serial number (if any), associated specification identifier, associated drawing number (if any), revision level, revision date, and supplier. • For each configured device, the report lists identifier, description, release date, install date, implementation status, point of contact, characteristics, and location. • For each individual, hardware and software resource, the report lists identifier, version, developer/vendor, make/model, serial number (if any), release date, install date, implementation status, scope, point of contact, characteristics, dependencies. • For each document, the report lists identifier, title, publisher, publication date, approval level, status, and latest document change notice
Baseline Changes	This report lists the as-built configuration items, assemblies, and control items that differ between two compared sets of baselined resources and depicts the listed items as added, deleted or revised, as appropriate. Also lists the configuration change requests implemented with each of the baselines.
Software Library Objects	This report identifies all or a selected subset of the objects contained in a software library, including its directories, principle elements, derived objects, and links. For each identified object that also exists in the user's present view of the library, the view rule that it satisfies is also presented.
Software Library Builds	This report lists the build ingredients used in creating specified, derived objects, such as the user who created the build, the host and view used for the build, the date and time of the build, file versions used as input, the derived objects produced by the build, the values of make macros used, and the build script executed.
Software Library Version Tree	This report lists part or all of the version tree of one or more software library elements.
Software Library Registered Views	This report lists the views registered on the local host.
Software Library View Specification	This report is a list of the specifications (rules) that define a specified view of the elements in the software library.
Software Library Checkouts	This report lists the library elements that are currently checked out.

Table 6.4.7-5. Configuration Management Reports (3 of 3)

Report Topic	Report Contents
Software Library Event History	<p>This report lists logged event records in reverse-chronological order. The following kinds of lists can be produced:</p> <ul style="list-style-type: none"> • Source Data History - events concerning the library's file system objects, including derived objects • Type History - events concerning metadata types that have been defined in the library • Storage Pool History - events concerning the library's storage pool • Library Object History - events concerning ClearCase's library data
Change Requests	<p>This report provides details about individual configuration change requests (CCR), non-conformance reports (NCR) or deficiency reports (DR), each of which is a proposal to change the configuration of a baselined system.</p>
Change Request	<p>This report provides a list of submitted CCR/NCR/DRs and selective information about each one.</p>
Change Request Metrics	<p>The contents of the report depend on operator selection and include, for example, a table of problems by project by state or severity or assigned engineer and severity. Report contents can be sorted and grouped by a number of other characteristics. In addition, it is possible to obtain statistical graphs, e.g., of problem distribution by severity, or of repair time and difference between estimated and actual fix time.</p>

6.5 User Interface Architecture

The user interface architecture for Release A ECS has two major components. The first is the science user interface, which is achieved by reuse of the Version 0 Client. The second is the operator user interface, which is both custom development and COTS, for Release A ECS. Each of these components are described in the following sections.

6.5.1 Release A Science User Interfaces

In order to reduce technical risk and schedule pressure for the design and development of Release A ECS, the decision was made to reuse the Version 0 Client as the ECS Release A client. The Release A client is the primary agent for science users to access ECS services and data. The Version 0 client provides the following functionality for users:

- Directory Search - allows a user to retrieve high-level information about data sets held at the DAACs
- * Access to the Global Change Master Directory (GCMD) - allows a user to retrieve high-level information on earth science datasets which may not be held as V0 data sets at the DAACs.
- Guide Search - Provides the user with detailed descriptions about data sets, their acquisition, projects they are associated with and the data centers that hold them
- Inventory Search - allows a user to identify specific observations or collections of observations (granules) that are available from a data center.

- Browse Information - provides the user with a visualization of the granules resulting from an inventory search.
- Product Ordering - allows a user to order data products based on the granule information obtained from an inventory search.

In order to reuse the Version 0 Client as the Release A Client the following approach has been implemented:

- 1) System level IMS components from V0 substitute for the functionality that the Client and Data Management Subsystems would provide in Release B ECS.
- 2) The V0 Client is incrementally improved so that it supports access to the ECS components at Release A.
- 3) The ECS Version 0 Gateway will provide translation between Version 0 protocols and the ECS protocols to allow the V0 Client to access ECS services and data.

A full description of the modifications to be made to the Version 0 client and the integration of the client with ECS for Release A can be found in the document "Implementation Plan for the Release A Client" (441-TP-001-001).

The ECS Release B user interface architecture and design are on-going and will be documented in the Client Subsystem volume of the Release B ECS System Design Specification.

6.5.2 Operator User Interfaces

One key element of the success of the ECS program is the user-friendliness and consistency in design of the graphical user interface (GUI). This section addresses (1) the GUI framework within which the ECS GUI will be designed, (2) the GUI implementation method to be used in developing the GUIs for each ECS Release A Subsystem, (3) the GUI design information that is contained in the DID 305 Volumes for each ECS Release A Subsystem, and (4) an example of a GUI template to be used in developing GUI screens that are compliant with the ECS User Interface Style Guide, Version 5.

This GUI Framework will be used in Release A to develop the Operator User Interfaces. This framework will also be used in future releases to develop the remaining Operator and Science User Interfaces.

6.5.2.1 GUI Framework

The ECS GUI framework represents a consensus between the ECS human factors engineers and the ECS project personnel as to the overall ECS GUI design concepts that will be employed in ECS Release A. The ECS GUI framework has three segments. The first segment consists of the GUI metaphors and interaction paradigms for the ECS desktop and workbench applications. The second segment consists of the ECS GUI design principles and guidelines contained in the ECS User Interface Style Guide. The final segment consists of the ECS GUI screen layout templates and widget sets used with the *Builder Xcessory* GUI tool to develop the actual ECS GUI screens. The segments are described in the following subparagraphs.

GUI Metaphors, Interaction Paradigms, and COTS Conventions. The formal definition of GUI metaphors and user interaction paradigms is fundamental to the successful design of GUIs. The selection of the Open Software Foundation's Motif user interface standard for ECS greatly simplifies the selection and implementation of the metaphors and interaction paradigms for the ECS program. The GUI metaphor and interaction paradigms are divided between those that support the design of the ECS desktop and those that support the design of the ECS workbench applications.

ECS Release A supports two desktops. The preferred desktop uses the new Common Desktop Environment (CDE) standard. This desktop will be installed on those workstations who run CDE. The existing ECS Window Manager is installed on the remaining workstations, running as a service on the native desktop that runs on each workstation. CDE will be standard for Release B. Therefore, the CDE will become the ECS desktop. The ECS desktop employs the use of Motif-compliant icons and iconic 'drag and drop' processes and multiple desktop workspaces as the primary desktop metaphors. The ECS desktop will employ a standard set of icons that represent the object type or process to which the icon is intended to be associated. The selection of an object or process icon invokes a specific action sequence that greatly simplifies the operator's interactions with the system. Object icons include data, documents, and other sorts of information that are manipulated by ECS by means of a variety of desktop office automation tools or through processes available from ECS applications built into the ECS workbench. Process icons represent the office automation tools available on the ECS desktop or tools and applications available on the ECS workbench.

Multiple desktop workspaces represent another metaphor that will be used in the ECS desktop. Multiple desktop workspaces provide operators with the capability to perform independent ECS functions using separate instances of the same desktop workspace. This will permit operators to lay out their desktop workspace in one manner for one ECS function, while maintaining a separate desktop workspace to support another ECS function. This provides operators a great deal of flexibility in optimizing their modes of operation for each ECS function and for tailoring their desktop environments to their own preferences. Multiple desktop workspaces may be 'saved' as icons by operators, allowing them the capability to 'restore' the workspace to the configuration that was saved - even after logging off the system and logging back on.

The major metaphor of the ECS workbench applications is the use of standard Motif screen layouts, including the use of pull-down menus, dialogs, and message boxes, among other widgets. Those COTS applications which employ a modifiable user interface, are tailored to comply with the ECS User Interface Style Guide to the maximum extent practicable.

The interaction paradigms of the ECS desktop and workbench employ variants of the Motif Object-Selection Model. This model controls the manner by which operators navigate the workspace. The ECS desktop employs an iconic object-process interaction paradigm. This paradigm defines the basis of iconic representation on the ECS desktop and the relationship between objects (entities which operators manipulate, such as documents) and processes (entities which manipulate objects). Simply stated, objects can be manipulated (processed) by 'dragging-and-dropping' the object icon onto an appropriate process icon (e.g., in order to print a document).

Applications on the ECS workbench are also designed to implement the Motif Object-Selection Model. However, instead of the iconic object-process interaction paradigm, applications will use

(a) traditional menu or dialog-based interaction techniques and (b) interactions using iconic processes located on a Motif Toolbar located directly below any pull-down menus on the screen.

ECS User Interface Style Guide. The second component of the GUI Framework is the ECS User Interface Style Guide, Version 5. The Style Guide documents the GUI metaphors and interaction paradigms described above. Further, the Style Guide contains human factors rules, principles, conventions, and heuristics as standards for designing and implementing ECS operator/user interfaces. Its purpose is to guide ECS GUI Developers in the creation of effective, user-friendly interfaces. Consistent application of these standards will ensure the implementation of a common "look and feel" across ECS operator/user interfaces.

ECS GUI Screen Development Templates and Widget Sets. The templates and widget sets selected specifically for the ECS program are used to design and implement ECS workbench applications that must be developed using custom code and COTS software applications that are custom 'wrapped' to support a consistent "look and feel" and comply with the Style Guide. The *Builder Xcessory* tool has been selected as the GUI Builder for those workbench applications that require a custom interface. The *Builder Xcessory* tool has been customized to create an ECS standard, using User-Defined Widgets that represent pre-defined GUI development templates and pre-defined styles that contain approved attributes (e.g., color, font, font point size) for any additional widgets that are created. Each pre-defined widget and pre-defined attributes have been developed to comply with the human factors guidelines contained in the ECS User Interface Style Guide, Version 5. Appendix B of the Style Guide lists and illustrates these ECS standard Widgets.

To comply with many of the human factors guidelines contained in the ECS User Interface Style Guide, the GUI Developer/Programmer only has to use the pre-defined widget templates and style manager containing the pre-defined widget attributes to develop the GUI screens. This method has been developed to achieve human factors-compliant GUI screens by rewarding the GUI Developers/Programmers with a rapid development process that meets established GUI development requirements. This method permits GUI Developers/Programmers to focus their effort on the aspects of GUI development that is most important to the users, namely, the implementation of meaningful operator interactions and dialogs in a windows environment.

6.5.2.2 GUI Implementation Method

The development of the ECS GUI follows accepted human factors practice for screen layout and design and involves the rigorous application of knowledge in human factors research of the human-computer interface. As shown in Figure 6.5.2-1, the practical application of this methodology is proceeding through three stages. The first stage, GUI Development Guidelines, which involves establishing the GUI framework (discussed above) has been completed. This resulted in a revision to the ECS User Interface Style Guide, Version 4. The second stage, GUI and Screen Layout Templating, involves (a) the definition of the ECS desktop and (b) development of GUI templates and styles for use in conjunction with a GUI Builder tool. Among other things, this has resulted in the development of GUI templates that assist and structure the GUI developer's use of the GUI Builder Tool (i.e., *Builder Xcessory*). The third stage, GUI Generation, involves the preparation of actual screens using the GUI builder, templates, and guidelines. The human factors engineers and the GUI designers cooperatively design each ECS

subsystem GUI through a three-step process, as discussed below. Finally, the completed products, the GUI guidelines, templates, and screens are assessed using established human factors criteria and user feedback. Revisions and updates are prepared as a result of these assessments and feedback.

As stated, the implementation method defined above ties the knowledge base of human factors research on human-computer interactions (HCI) to best commercial practice for the rapid development of GUI screen layouts. The theoretical underpinnings of this method are illustrated by the model shown in Figure 6.5.2-2.

As shown in the figure, consideration is given to the ECS design requirements for the computer software configurations (CSCs) that possess a custom user interface. The methodology requires the identification of the data input and output requirements of the CSCs. On the other hand, the methodology requires consideration of the human performance requirements associated with performance of tasks using the system. This includes the identification of data, command and control, and task requirements of the human operator in system interactions.

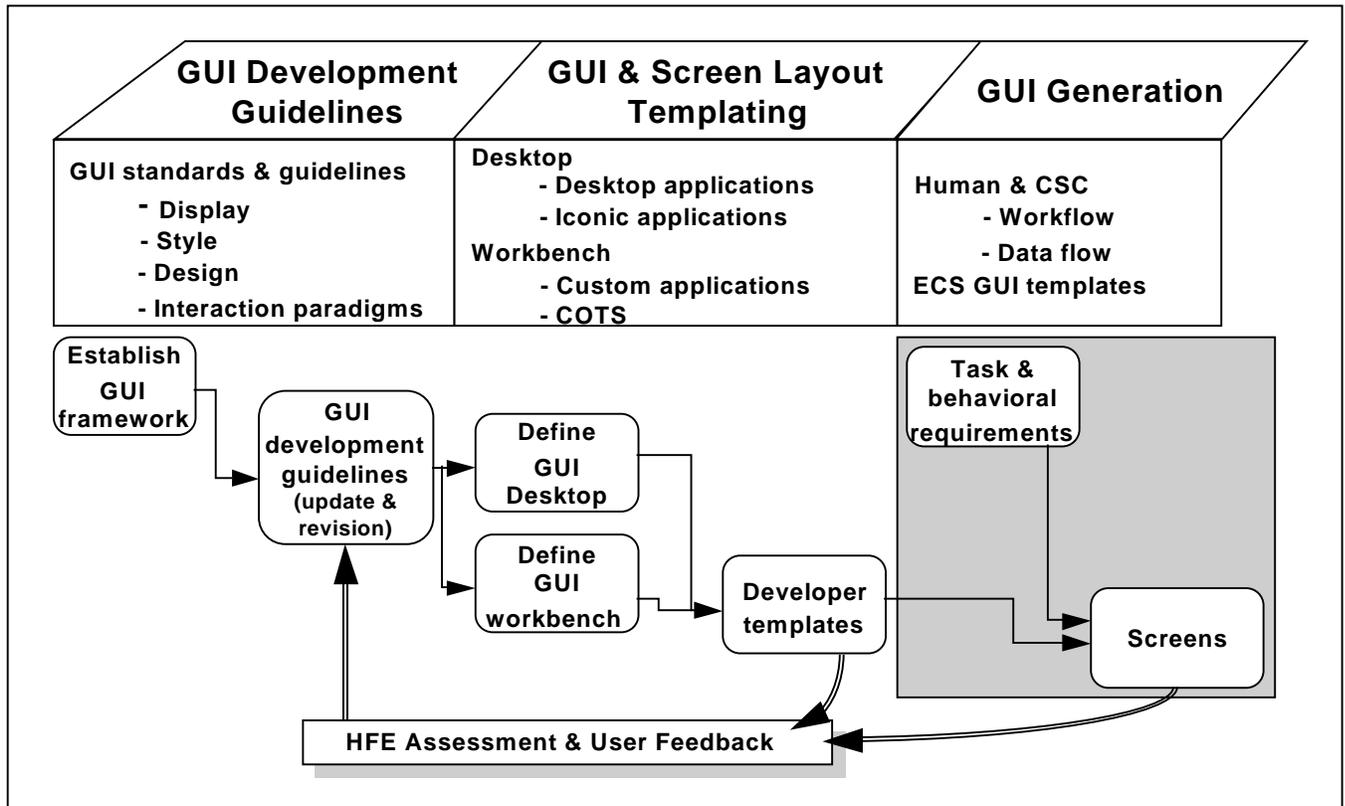


Figure 6.5.2-1. The GUI Implementation Method for ECS Release A

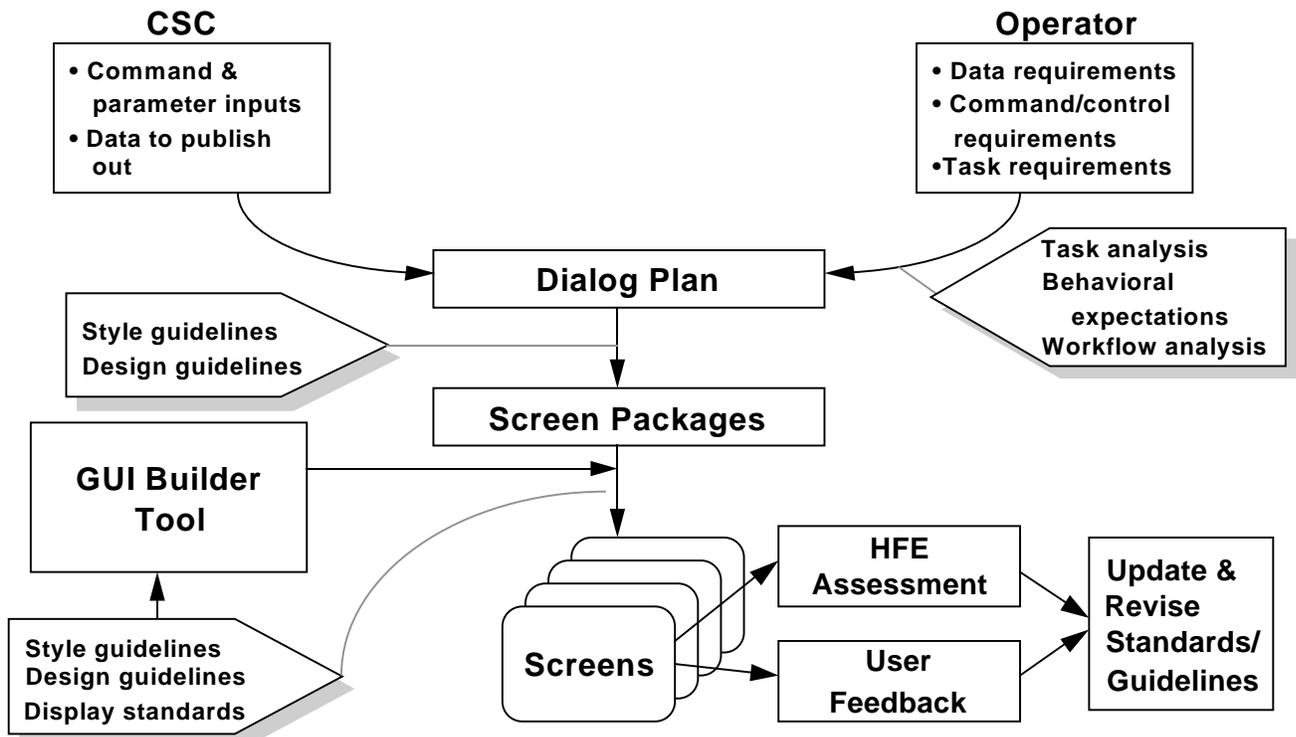


Figure 6.5.2-2. Model for the Development and Assessment of the ECS Release A GUI

The human factors engineer applies knowledge of the human factors research on HCI and acquired knowledge about the computerized system to assist the ECS Subsystem Designer in the development of the workflows for each ECS subsystem. Workflow analysis is used to determine the human-computer dialogs required for the human to interact with the computer to accomplish specified tasks. The human factors engineers and GUI Designers then take these dialogs, and using the GUI style and design guidelines selected specifically for application to the ongoing development effort, prepare a series of screen packages (or screen layout diagrams) that provide storyboards for GUI Developers to use in developing actual computer screens. The GUI Developers use the GUI Builder Tool, templates, and widget set to rapidly develop computer screens that implement the dialogs and storyboard screens. The resulting screens can be reviewed by the human factors engineers for compliance with the GUI style and design guidelines. At this point, completed sets of screens can be subjected to a series of human factors assessments and user feedback on the adequacy of the HCI in meeting program and user goals. Finally, comments and human factors issues can be used to update and revise the GUI guidelines as well as the screens themselves. In this manner, the model shown in the figure depicts a complete set of HCI dialogs and screens that can be rapidly developed and maintained by the human factors engineering and ECS development staff.

Using the model discussed above, the practical implementation of an ECS Subsystem GUI is conceived of as an iterative, three-step design process, as shown in Figure 6.5.2-3. This process is depicted by the gray area in the figure to involve workflow analysis, preparation of screen packages, and development of the GUI screens. As shown in the figure, the data required by ECS operators/users is used to analyze the workflow requirements of information manipulated by ECS operator/users in performing functions assigned to the CSCs. The workflows provide a series of block diagrams that step a hypothetical ECS operator/user through each action required to interact with the ECS subsystem software. The workflows provide the input required for human factors engineers and GUI Designers to design the screen layouts using screen layout diagrams. The ECS User Interface Style Guide is used as the source of human factors guidelines to support the design of screen packages. Each diagram in the screen packages relates to the operator interactions identified in the workflows. Finally, the third step of this process is to develop the GUI screens using the GUI Builder tool. The screen packages are handed off to the GUI Developer/Programmer for implementation using the GUI Builder, the templates, widgets, and GUI style attributes that have been defined directly into the GUI Builder tool itself. The GUI screens prepared using this three-step, iterative process can be subjected to a series of internal reviews, human factors assessments, and external operator/user reviews to provide the necessary feedback required to improve the screen layouts. Iterative reviews of development GUI screens may occur at any stage in the GUI development process.

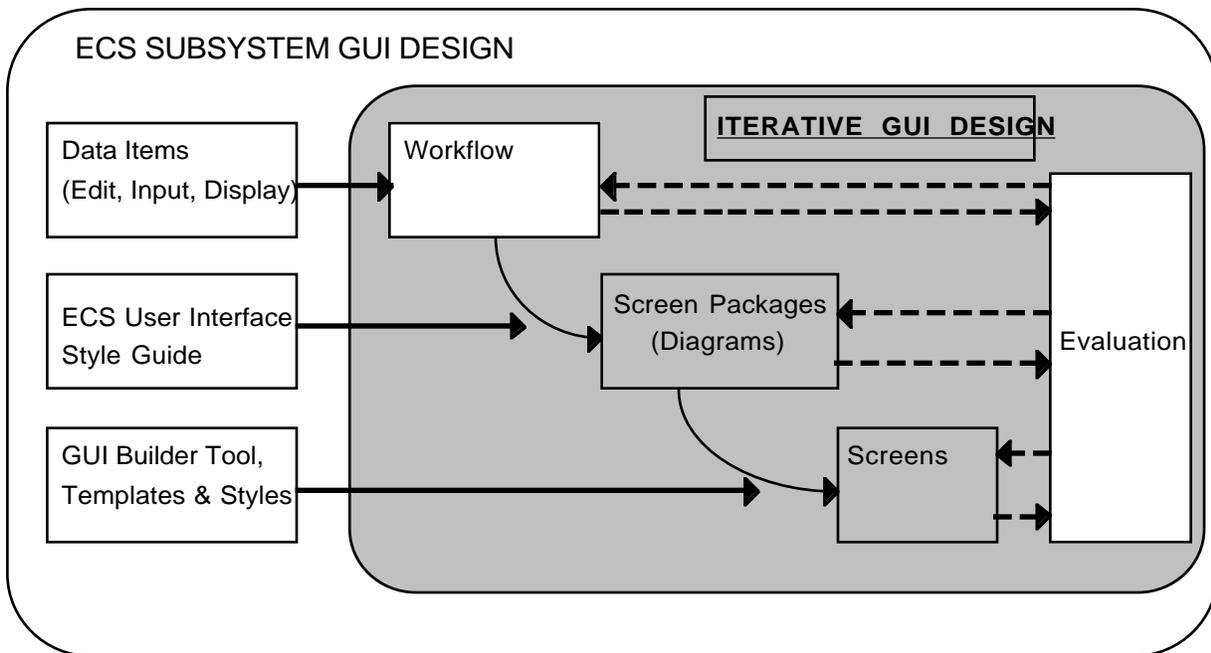


Figure 6.5.2-3. Description of the Three-Step ECS GUI Design Process

6.5.2.3 Description of GUIs

Preparatory to the pursuit of the three-step GUI design process depicted in Figure 6.5.2-3, each ECS Release A Subsystem development team is required to specify their data requirements with respect to the data that must be available to an operator/user in order for them to perform functions associated with the subsystem. Operators/users interact with or manipulate data in at least three ways, namely, edit, input, and observe data on a display. Data on these interactions provide the information required to define the operator workflows and screen layout displays for each ECS Release A Subsystem. Later, each ECS Release A Subsystem development team will prepare the workflows, screen packages, and screens based, in part, on these data. The required data on interactions include identification of the functions and associated data elements, and specific identification of the operator/user interaction with each data element, viz., whether it is 'display only,' (i.e., data elements which are displayed but will not be changed), 'edit,' (i.e., data elements whose current value will be displayed, but may be changed by the operator/user), or 'input only,' (i.e., data elements whose current value is blank or null, and which will be entered on that occasion).

Each ECS Release A Subsystem development team prepared a complete listing of the data elements required for operators/users to perform functions using the subsystem, in accordance with the interaction data requirements. The information for each ECS Release A Subsystem is located in the relevant GUI section of the DID 305 Volume assigned to each ECS subsystem.

6.5.2.4 Example of a GUI

The GUI for ECS workbench applications makes use of Motif widgets, including, main windows, pull-down menus, toolbars containing icon buttons, button bars containing push buttons, and application workspaces that may be used to display a variety of widgets such as, radio boxes, scrolled windows, bulletin boards, and other container widgets. Figures 6.5.2-4 and 6.5.2-5 present a facsimile of the "look and feel" of a hypothetical instance of an ECS workbench application. Figure 6.5.2-4 shows the screen structure, and Figure 6.5.2-5 shows a sample screen print-out illustrating the implementation of the structure. These widgets and whole sets of others are accessible to the GUI Developer/Programmer through a widget palette provided with the *Builder Xcessory* software tool. Widget attributes have been defined for each of these widgets in accordance with the ECS User Interface Style Guide, Version 5. Additionally, collections of pre-defined widgets have been prepared using *Builder Xcessory* that provide a set of approved GUI development templates for use by GUI Developers/Programmers in developing GUIs for specific ECS Release A subsystems. The GUI development templates provide a full range of approved screen layouts that comply with the color, font, font size, screen positioning of widgets, and widget layouts defined in the Style Guide. GUI Developers/Programmers can rapidly develop their GUIs using widget collections that already comply with the Style Guide. GUI Developers/Programmer can readily make modifications to each instance of a GUI development template, by adding and deleting widgets, or by changing the widget attributes of those attributes that are not controlled by the ECS User Interface Style Guide. This affords the GUI Developer/Programmer a standardized GUI development framework that encourages innovation in GUI design. It is flexible and ensures incorporation of human factors principles by providing structure, as illustrated in the figure.

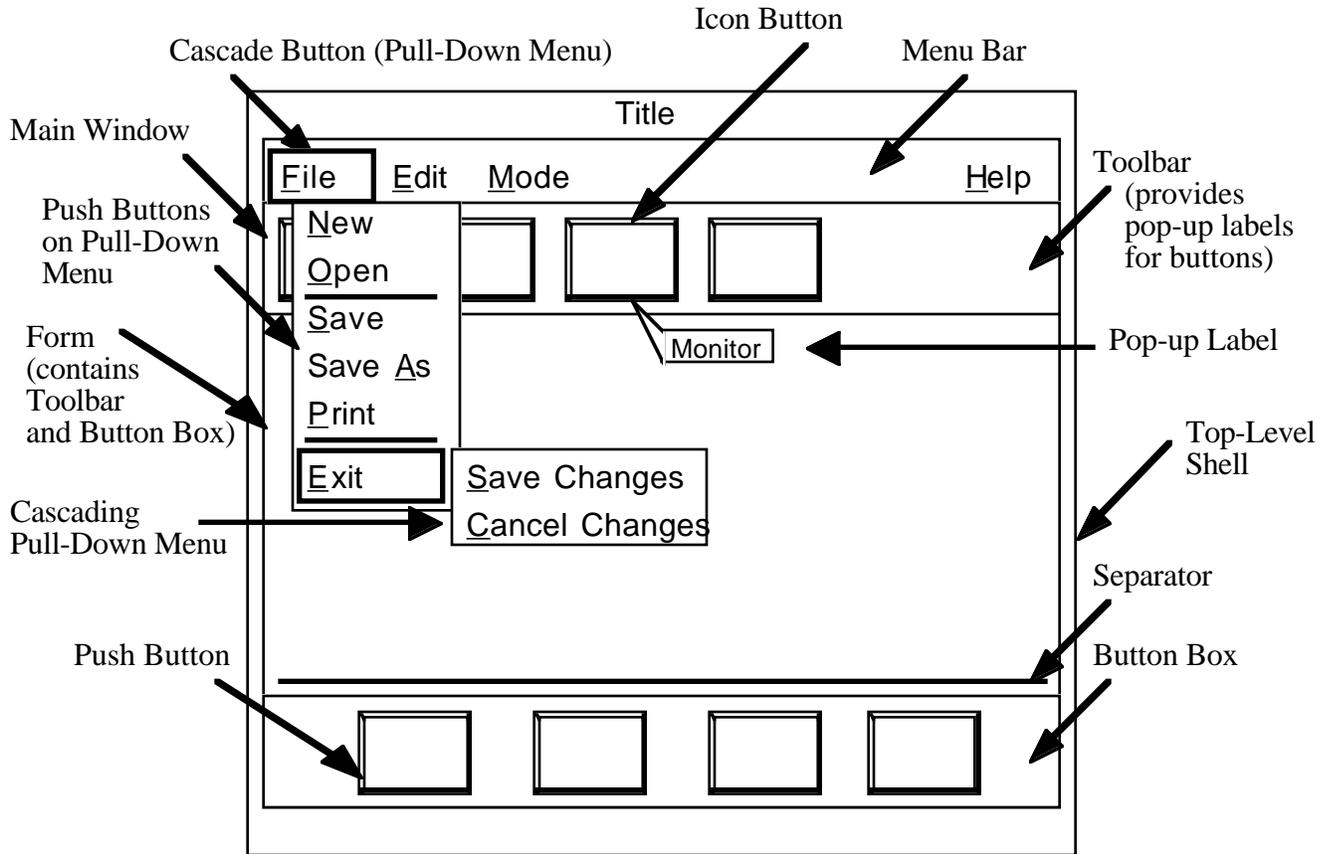


Figure 6.5.2-4. Sample Structure for an ECS Graphical User Interface

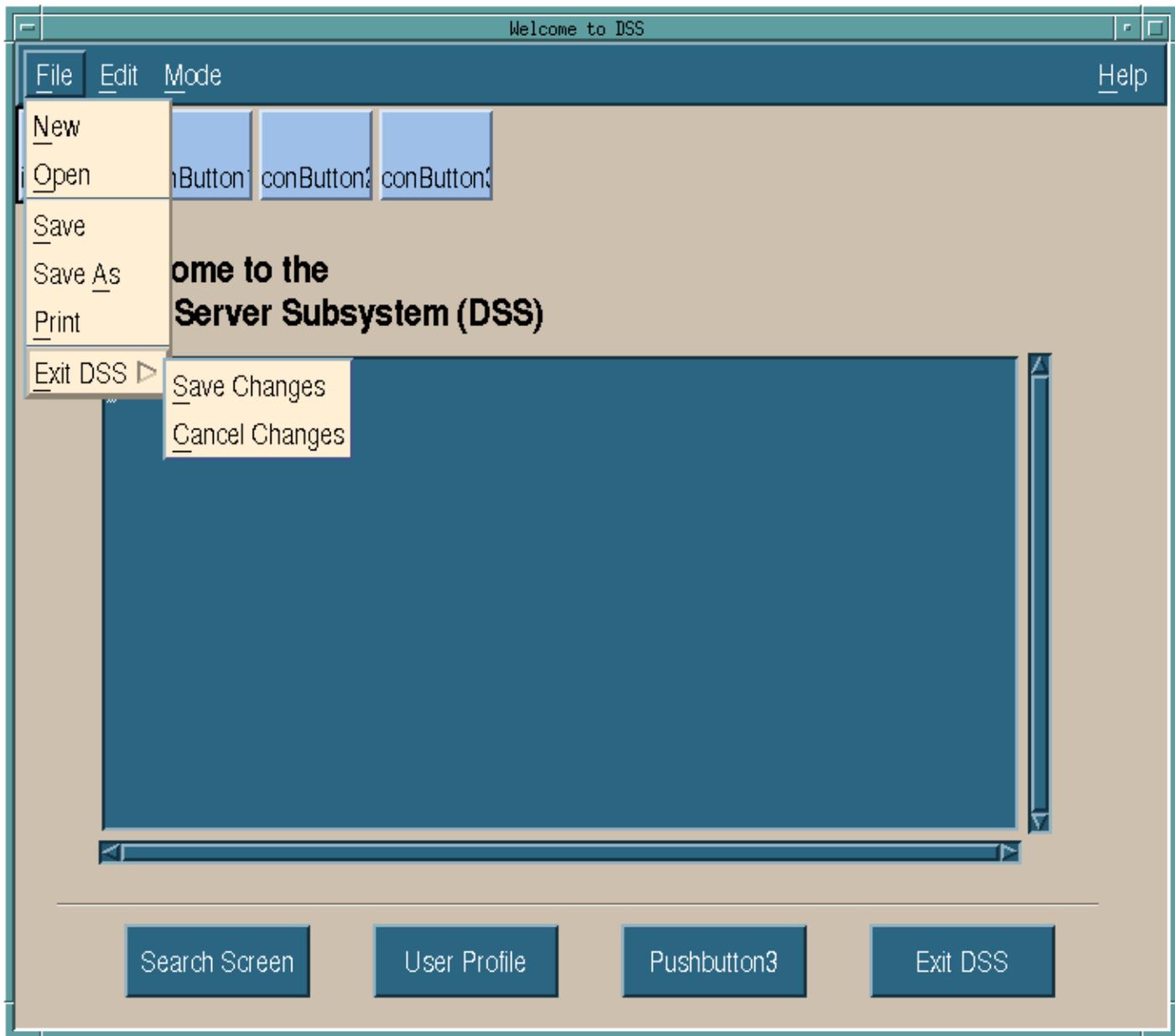


Figure 6.5.2-5. Print-out of Sample Showing ECS GUI Screen Structure

7. Methodology Overview

The remainder of this document provides further detail on the design of ECS hardware and software architectures as well as individual subsystems. Much of this presentation reflects the object-oriented design methodology adopted by ECS. The rest of this section includes an overview of this methodology to aid the reader of the remaining sections.

7.1 OMT

Object-oriented methodology is a development paradigm that organizes a system as a collection of objects, each of which has data structure and behavior and which has meaning within the context of the problem that is being modeled. The methodology being used on the ECS Program is the Object Modeling Technique (OMT) set forth by Rumbaugh, et al in the book Object-Oriented Modeling and Design. In object-oriented methodologies, the analysis and design are developed in terms of graphical models. The foundation of OMT is the object model, in which the complete static structure of the system is captured.

The following material provides a tutorial on how to read an OMT object model. The tutorial is in the form of a walk-through of a sample model. Although the sample does not use all of the available notation, it uses most of the notation that will be seen in models that have been constructed for ECS. Before starting the walk-through, the following definitions have to be understood.

- **Object:** An abstraction of something in the problem at hand, characterized by a unique name, distinct properties, and well defined behavior.
- **Class:** A group of objects with the same meaning, properties (*attributes*), behaviors (*operations*), and relationships (*associations*) with other objects.
- **Generalization:** Objects can be generalized into a more generic object class. For example, guides, program descriptions, and general system descriptions could be generalized into a common class called documents. The document class is then called the parent class of guides, program descriptions, and general system descriptions.
- **Attribute:** a named property of a class, describing data values held by each object in the class. Classes describe the data property (e.g., color). Each object holds a value (e.g., green) for each attribute defined for the class to which the object belongs.
- **Operation:** a part of the behavior of a class. Collectively, all of a class' operations define the things that objects of the class can do.
- **Link:** a physical or conceptual connection between object instances -- an instance of an *association* (see the next definition).
- **Association:** a group of links with common structure and common meaning -- a set of potential links.
- **Aggregation:** The model also recognizes a specific kind of relationship, called Aggregation. It indicates that objects of one class (the aggregate) are composed of objects belonging to other classes (the components).

The following design document uses several types of modeling diagrams:

- Object Model Diagrams depict the classes of objects which make up a design, their attributes and operations, and how they are related to each other. In essence, the concepts presented in the above list are presented diagrammatically (see Section 7.2).
- Event Trace Diagrams depict a sequence of events that occur in a scenario. At the preliminary design level, scenarios are concerned with the interactions that take place among objects. Events, therefore, represent messages which are sent from one object to another (see Section 7.3).
- State Transition Diagrams are used occasionally in the design to show how the messages affect the internal state of an object, and in particular, the events which cause state transitions (see Section 7.4).

7.2 OMT Diagram Tutorial

Figure 7.2-1 shows the notation used by the ECS Object Models. The rectangular boxes in the model denote classes. Each box, shown in full detail, consists of three sections. The name of the class fills the top section, its attributes go in the middle section, and its operations in the bottom section. Sometimes in high level drawings, only the top section of the box, showing the class name, is shown. A class may be the generalization of several other classes. In Figure 7.2-1, the "Parent Class" is the generalization of two other classes, each called a "Derived Class." Derived classes always include the attributes and operations provided by their parent classes. The diagrams, therefore, only show any additional attributes or operations which the derived class may have.

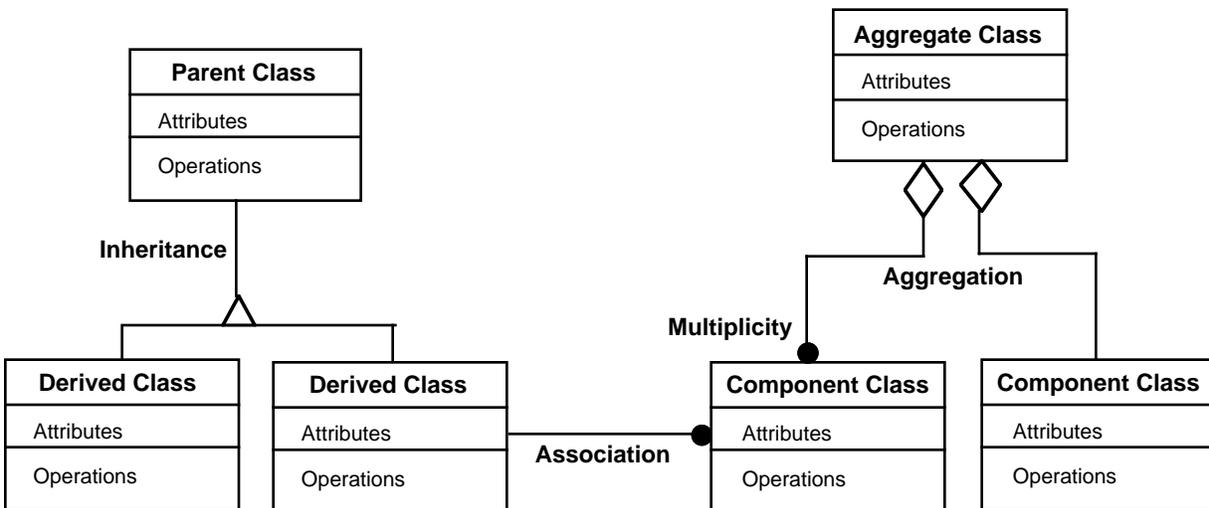


Figure 7.2-1. Object Model Diagram Notation

Figure 7.2-1 also shows that there are two classes, each called a "Component Class", have been aggregated into another class, called the "Aggregate Class". There may be design rules which determine how many components of each class an aggregate may have. This is shown by

providing an indication of the "Multiplicity" in the diagram. In 4-1, the left component may occur any number of times (zero, one, or many), the right component must occur exactly once. Finally, classes may have relationships, indicated by simple lines. On the design diagrams, they are labeled with the name of the relationship, and they carry an indication of multiplicity.

Figure 7.2-2 shows an example, taken from the Production Planning CSCI, showing an excerpt of the object classes supporting the management of production resources. All resources have a common set of attributes providing an identification, a name, and their current state, as well as operations to allocate and deallocate the resource and update its state. There is, therefore, a single class called "PIResource" which acts as the parent for all types of production resources.

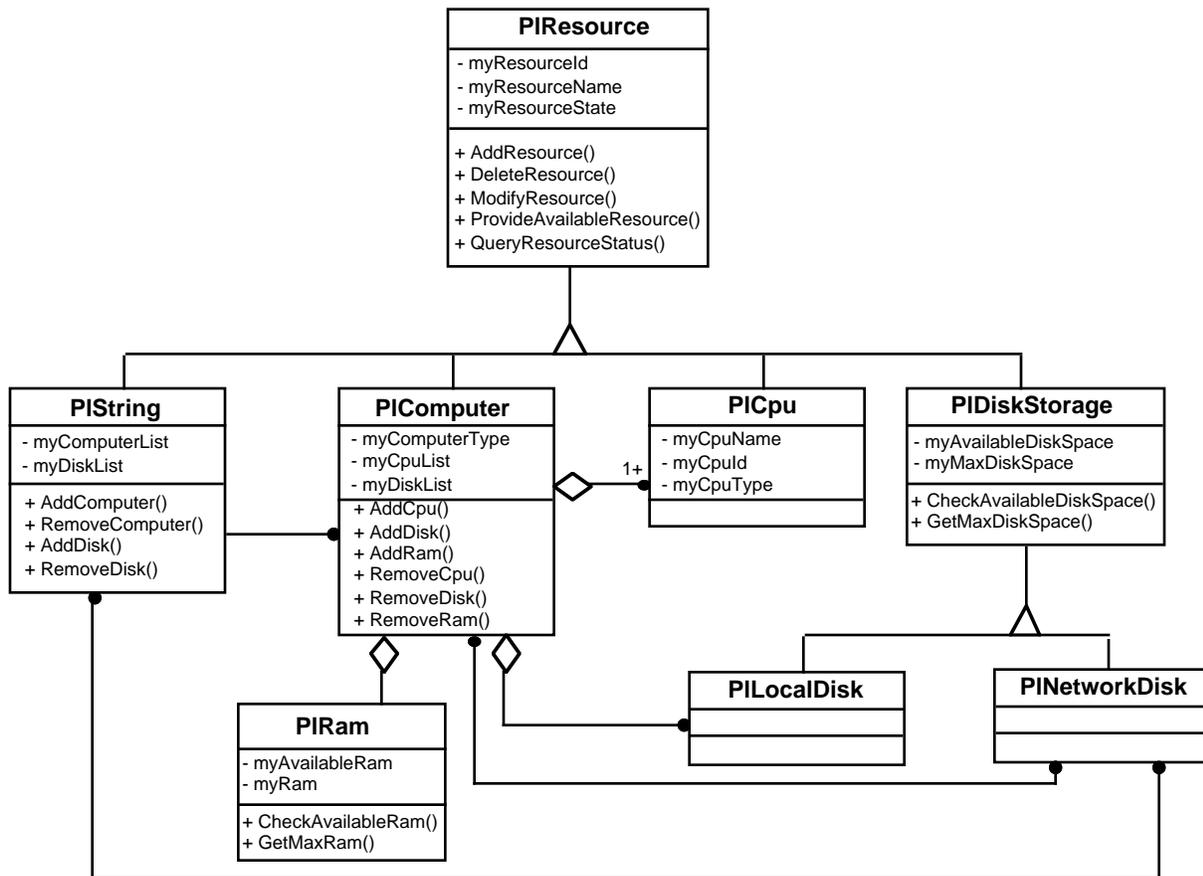


Figure 7.2-2. Example of an Object Model Diagram

The diagram shows several types of resources, each a derived class. For example, strings (represented by "PIString") may be associated with several computers and several network disks. A computing platform ("PIComputer") consists of several components, namely several cpu ("PICpu"), disks ("PILocalDisk") and main memory ("PIRam"). It may also use several network disks ("PINetworkDisk"), and those disks might be attached to several computers.

7.3 Event Trace Diagram Tutorial

Figure 7.3-1 shows an example of a event trace diagram, again taken from the Production Planning CSCI. The example shows a scenario in which a previously created plan is activated (the example has been chosen for its simplicity). The first even occurs in the user interface, when production planning staff selects a previously created plan and activates it. Correspondingly, the diagram shows an arrow from the Planning User Interface to the Plan.

The software associated with the Plan object now will perform a number of operations, such as verifying that the plan being activated is indeed valid, noting the differences between the currently active plan and the new one, and creating new processing requests (or updating existing ones) to implement the new plan. The result of this activity is a series of messages to the "DPR" class. Each instance of this class represents a pending job and has associated with it any information which must be provided as input to the processing CSCI when the PGE is initiated through a request for data processing (hence the name of the class). Each message creates a new instance of a pending job, or updates a currently existing pending job.

Concurrently, the planning CSCI monitors the inputs for which there are currently waiting jobs (i.e., for which there is processing in the currently active plan). In essence, each of the pending jobs waits for its inputs to become available. When they are, the pending job (i.e., the corresponding instance of "DPR") is sent to the Processing CSCI. In the object design, interfaces with external CSCI are typically represented as "Interface Classes". The "Processing Planning Interface" is such an interface class. In the Planning CSCI object model, it is the target of the DPR message.

7.4 State Transition Diagram Tutorial

Figure 7.4-1 shows an example of a state transition diagram. The rectangular box used to depict object classes this time is used to show the possible states of the object. In the case of a production plan ("PIPlan") there are two possible states:

- The plan can be a "candidate plan", i.e., one that has been fully planned but has not been activated. A candidate plan is created with the "create plan" command. This is the initial state of a plan. This is indicated in the diagram by showing that there is no previous state (i.e., with a filled black circle).

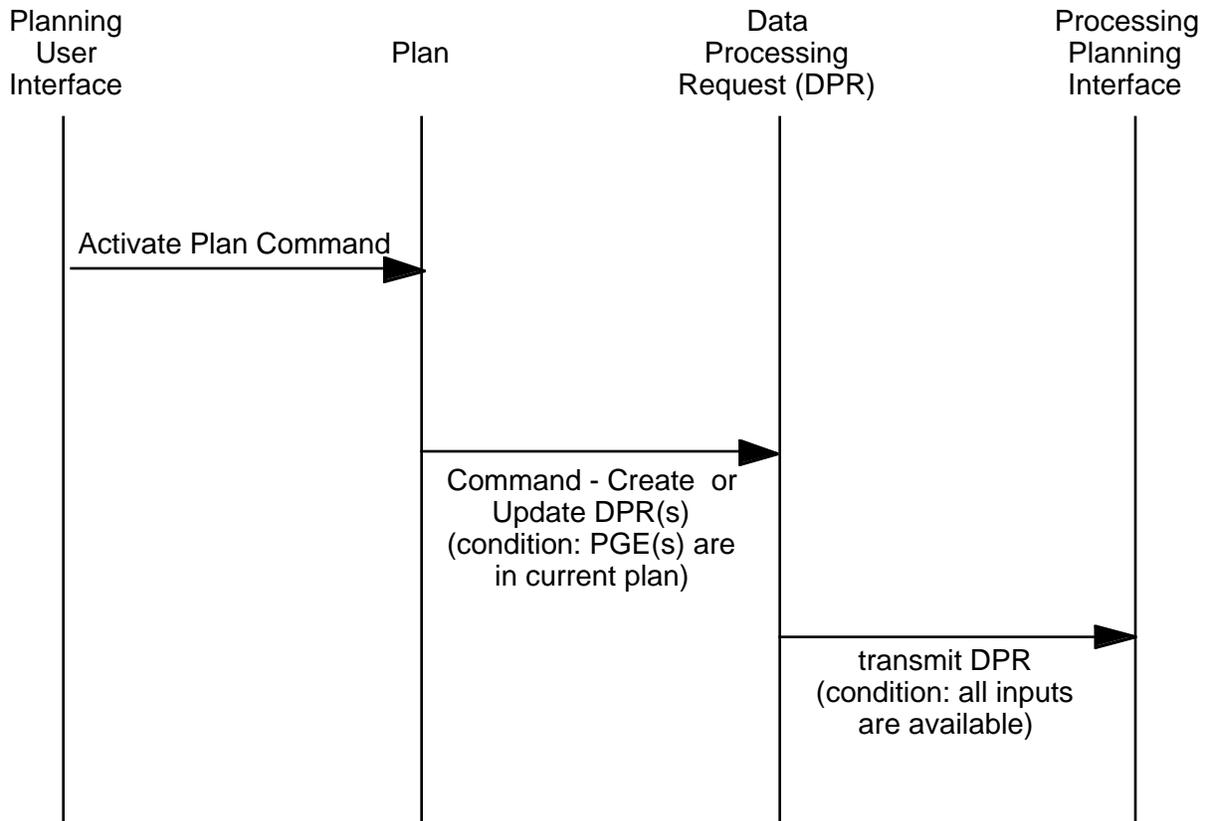


Figure 7.3-1. Example of an Event Trace Diagram

- A candidate plan can be made an "active plan" via an "activate plan" command. In this state, the plan receives alerts for on-demand production requests and data arrival notifications (DAN). It instructs DPR objects to transmit the corresponding data processing request message when the job is ready (as discussed in the scenario in Section 7.3). This is the final state of a plan. The plan is terminated when another plan is activated, or when this plan is canceled. The final state of an object is indicated by the symbol shown to the left of Activate Plan.

The diagram shows that during the creation of the candidate plan, the plan object interacts with a number of other objects: with "Data Processing Requests" to obtain information about the PGE in the plan; with "Resource Management" to determine the availability of resources needed by those PGE, and with the data server (represented by the "Data Server Interface" class) to store the candidate plan. Successful creation of the plan (or failure) also display a response on the planning user interface.

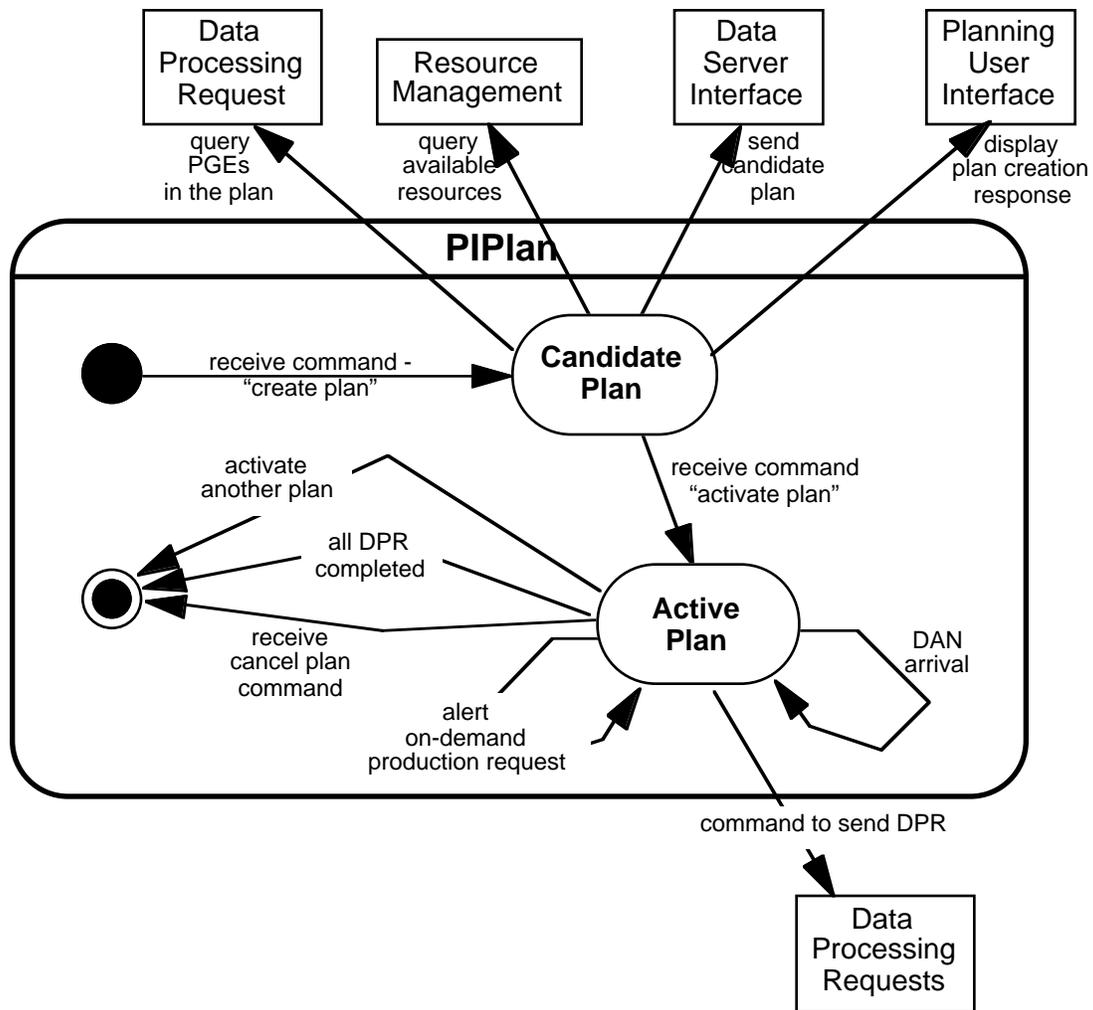


Figure 7.4-1. Example of a State Transition Diagram