

## 6. CSS Functional Requirements

---

Section 6 contains the requirements associated with the Communications Subsystem (CSS). This includes the Common Facilities Service, Object Services and Distributed Object Framework Services. This section specifies the functional requirements associated with the Distributed Computing Configuration Item (DCCI) and is organized by the following sections:

- 6.1 General Requirements
- 6.2 Common Facility Services
- 6.3 Object Services
- 6.4 Distributed Object Framework

## 6.1 General Requirements

### 6.1.1 CSS Interface Requirements

The CSS interfaces are identified in Figure 6.1-1.

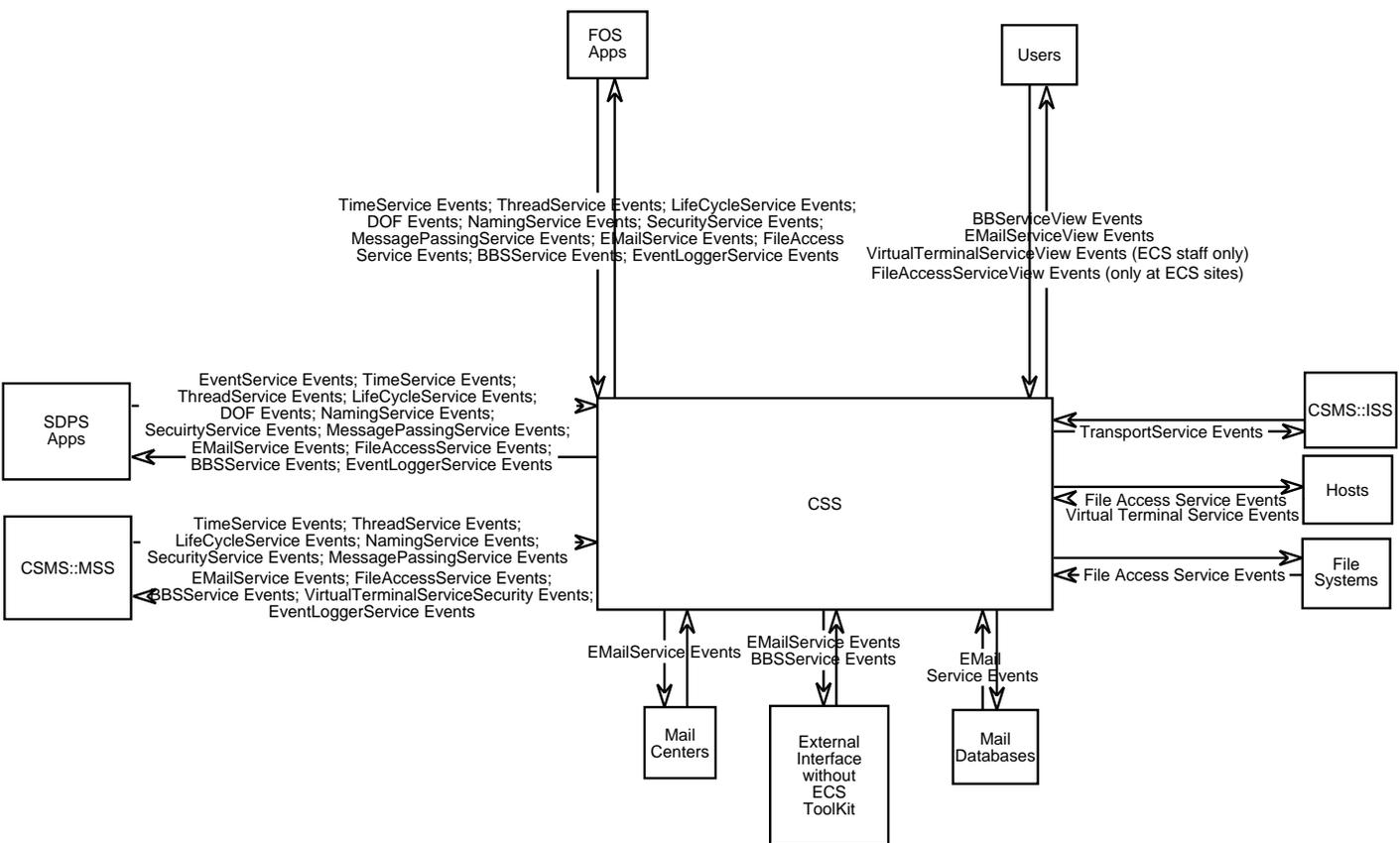


Figure 6.1-1. CSS Interface Diagram

### 6.1.1.1 CSS/External Interface Requirements

The CSS interfaces with external systems for NTP time source.

The CSS interface requirement with external systems follows:

C-CSS-10090            The CSS shall interface with a TBS external time source for coordinated universal time (UTC).

### 6.1.1.2 CSS/SDPS Interface Requirements

Table 6-1 summarizes the CSS interface with the SDPS subsystem and includes the source and destination of the interface, and a brief description of the data item.

**Table 6-1. CSS/SDPS Subsystem Interface**

Source	Destination	Data Description
CSS API	Any SDPS subsystem	Event Event logger Time Message Passing File Access
CSS server	Any SDPS subsystems	Electronic Mail Bulletin Board Virtual Terminal Directory/Naming Security
Any SDPS subsystem	CSS server	Service request Authenticate V0 clients to ECS

The CSS interface requirement with the SDPS subsystem follows:

C-CSS-10100            The CSS shall interface with the SDPS subsystems to exchange the data items in Table 6-1 as specified in the ECS internal ICDs, 313-DV3-003.

### 6.1.1.3 CSS/FOS Interface Requirements

Table 6-2 summarizes the CSS interface with the FOS subsystem and includes the source and destination of the interface, and a brief description of the data item.

**Table 6-2. CSS/FOS Subsystem Interface**

Source	Destination	Data Description
CSS API	Any FOS subsystem	Event Event logger Time Message Passing File Access
CSS server	Any FOS subsystem	Electronic Mail Bulletin Board Virtual Terminal Directory/Naming Security
Any FOS subsystem	CSS server	Service request

The CSS interface requirement with the FOS subsystem follows:

C-CSS-10200            The CSS shall interface with the FOS subsystems to exchange the data items in Table 6-2 as specified in the ECS internal ICDs, 313-DV3-003.

**6.1.1.4 CSS/MSS Interface Requirements**

Table 6-3 summarizes the CSS interface with the MSS subsystem and includes the source and destination of the interface, and a brief description of the data item.

**Table 6-3. CSS/MSS Subsystem Interface**

Source	Destination	Data Description
CSS API	MSS	Event logger Message Passing Time File Access
CSS server	MSS	Electronic Mail Bulletin Board Virtual Terminal Directory/Naming Security Message Passing
MSS	CSS server	Service request

The CSS interface requirement with the MSS subsystem follows:

C-CSS-10300            The CSS shall interface with the MSS subsystems to exchange the data items in Table 6-3 as specified in the ECS internal ICDs, 313-DV3-003.

### 6.1.1.5 CSS/ISS Interface Requirements

Table 6-4 summarizes the CSS interface with the ISS subsystem and includes the source and destination of the interface, and a brief description of the data item.

**Table 6-4 CSS/ISS Subsystem Interface**

Source	Destination	Data Description
CSS	ISS	Lower layer ISO services (TCP/UDP/IP)
ISS	CSS	None

The CSS interface requirement with the ISS subsystem follows:

C-CSS-10400            The CSS shall interface with the ISS subsystems to exchange the data items in Table 6-4 as specified in the ECS internal ICDs, 313-DV3-003.

### 6.1.2 CSS Performance Requirements

C-CSS-00200            The CSS services shall allocate 10% of development resources for IV&V activity.

### 6.1.3 CSS RMA Requirements

C-CSS-00010            The CSS services shall have an operational availability of .998 and an MDT of 20 minutes or less for critical services.

C-CSS-00020            The CSS services shall have no single point of failure for functions associated with network databases and configuration data.

C-CSS-00030            The CSS services shall be extensible in its design to provide capability for growth and enhancement.

C-CSS-00040            The CSS services shall be compatible with POSIX-compliant Unix platforms.

C-CSS-00100            The CSS directory services shall maintain multiple copies of the namespace on different hosts to provide fault tolerance.

### 6.1.4 CSS General Requirements

C-CSS-00500            The CSS client services software shall be made available in the form of a CSS toolkit to the developers.

C-CSS-00510            The CSS shall provide access to ECS data and services to the clients at the DAACs and SCFs without distinction using ECS provided software.

Note: Providing same service requires that SCFs use DCE similar to the DAACs.6.2  
Common Facility Services

Common Facility Services are defined as those communication interfaces and uniform semantics that are shared across applications. These services make applications easier to develop and maintain across multiple application domains and include interactive tools for the operators and programmatic interfaces for applications.

## **6.2 Common Facility Services**

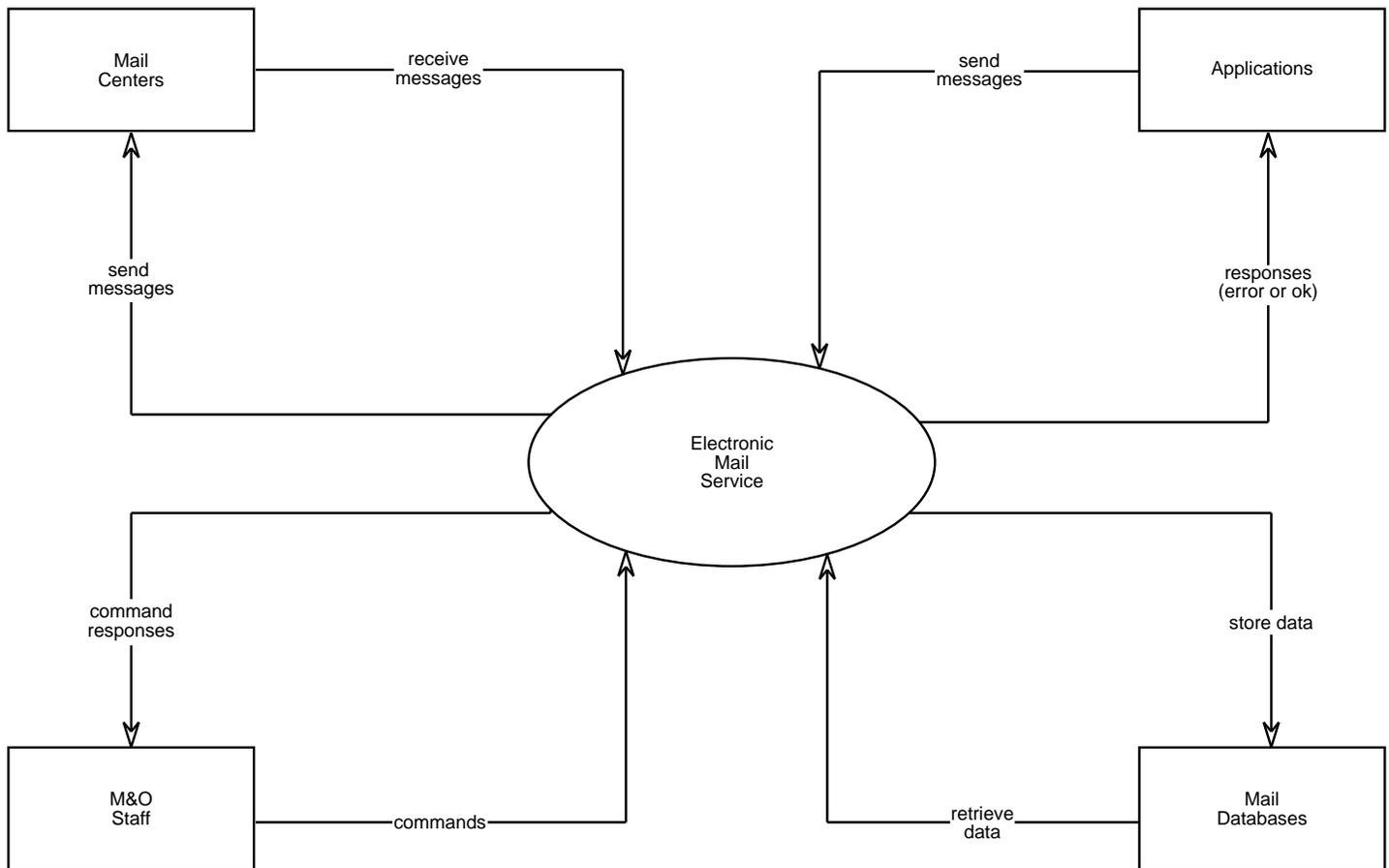
### **6.2.1 Electronic Mail Service**

#### **6.2.1.1 Overview Electronic Mail Service**

This service provides interactive (for the operators) and programmatic (API, for the applications) interfaces to manage electronic mail messages. A operator, for this discussion, is defined as a member of the M&O staff. It is not the intention of the ECS to provide interactive mail capability or mail stores for external end users.

Interactive interfaces provide a full set of functionality (see requirements below). The API is limited in scope as it only addresses sending a message from within an application.

Following context diagram shows the data flows and interaction of the service with external entities.



**Figure 6.2-1. Electronic Mail Context Diagram**

### 6.2.1.2 Electronic Mail Service Functional Requirements

Requirements for this service are divided in three sections.

*General* section addresses the mail server requirements and the protocols & standards the service needs to support.

*MailTool* section addresses the functionality of the interactive tool provided as part of ECS. It is expected that these requirements will map directly into a COTS package and therefore will be used for making a COTS selection.

Finally, the *API* section addresses the requirements for the programmatic interface provided to the applications.

#### General

C-CSS-61010 The CSS Electronic Mail Service shall interoperate and exchange messages with external mail systems based on SMTP and X.400 protocols.

Note: Simple Mail Transfer Protocol (SMTP) is described in IETF RFC 821.

C-CSS-61020 The CSS Electronic Mail Service shall be capable of sending and receiving the Multi-purpose Internet Mail Extensions (MIME) messages.

Note: MIME is described in IETF RFCs 1521 and 1522.

C-CSS-61030 The CSS Electronic Mail Service shall use the existing X.400 gateway available at GSFC to support X.400 operations.

C-CSS-61040 The CSS Electronic Mail Service shall provide translation between SMTP and X.400 protocol.

C-CSS-61050 The CSS Electronic Mail Service shall be accessible in interactive mode.

C-CSS-61060 The CSS Electronic Mail Service shall be accessible in non-interactive mode via API.

C-CSS-61290 The CSS Electronic Mail Service shall provide functionality to send reply for a received message to

- a. the author (Reply)
- b. to all destinations addressed in the incoming message (ReplyAll)
- c. named destinations (Forward).

#### MailTool

C-CSS-61310 The CSS Electronic Mail Service shall provide a MAILBOX where all incoming messages for operators will be stored.

Note: Only M&O staff will be provided a MAILBOX. ECS does not provide mail account to end users.

- C-CSS-61320 The CSS Electronic Mail Service shall provide operator defined folders to store messages for long term archive.
- C-CSS-61330 The CSS Electronic Mail Service shall allow copying and/or moving messages from the MAILBOX to the operator specified folders.
- C-CSS-61360 The CSS Electronic Mail Service shall be capable of showing a summary of all messages in the MAILBOX or in a folder which minimally contains:
- a. title/subject of the message
  - b. name of the author
  - c. date/time of the message origination
- C-CSS-61370 The CSS Electronic Mail Service shall provide an editor to compose a message.
- C-CSS-61380 The CSS Electronic Mail Service shall provide a title/subject field for a message.
- C-CSS-61390 The CSS Electronic Mail Service shall allow a message to be sent to multiple destinations.
- C-CSS-61400 The CSS Electronic Mail Service shall allow destinations of the following types:
- a. a single user
  - b. a position which may be managed by one or many operators
  - c. a site which may consists of several operators
- C-CSS-61410 The CSS Electronic Mail Service shall provide a capability to maintain public mailing lists (each list may contain multiple destination) which are accessible to all operators.
- C-CSS-61420 The CSS Electronic Mail Service shall provide a capability to maintain private mailing lists (each list may contain multiple destination) for individual operators.
- C-CSS-61430 The CSS Electronic Mail Service shall allow attaching either text or binary files to a message.
- C-CSS-61440 The CSS Electronic Mail Service shall allow discarding message(s) from the MAILBOX without saving.
- C-CSS-61450 The CSS Electronic Mail Service shall have the capability to forward a message.

- C-CSS-61460      The CSS Electronic Mail Service shall allow cut/copy/paste/delete/undo operations in the editor.
  - C-CSS-61470      The CSS Electronic Mail Service shall provide navigation methods to go the next or previous message in the MAILBOX or selected folder.
  - C-CSS-61490      The CSS Electronic Mail Service shall provide the capability to search for keywords in messages.
  - C-CSS-61500      The CSS Electronic Mail Service shall provide the capability to search the MAILBOX or a folder for keywords in title text.
  - C-CSS-61510      The CSS Electronic Mail Service shall provide the capability to search the MAILBOX or folders for a specific author.
- Note: Above requirements for the search capability (490,500,510) may not be achievable with COTS. These are under discussion and may be changed or removed in a later version of this document.
- C-CSS-61520      The CSS Electronic Mail Service shall accept mailing lists as valid destinations.

**Application Program Interface (Non-interactive, API)**

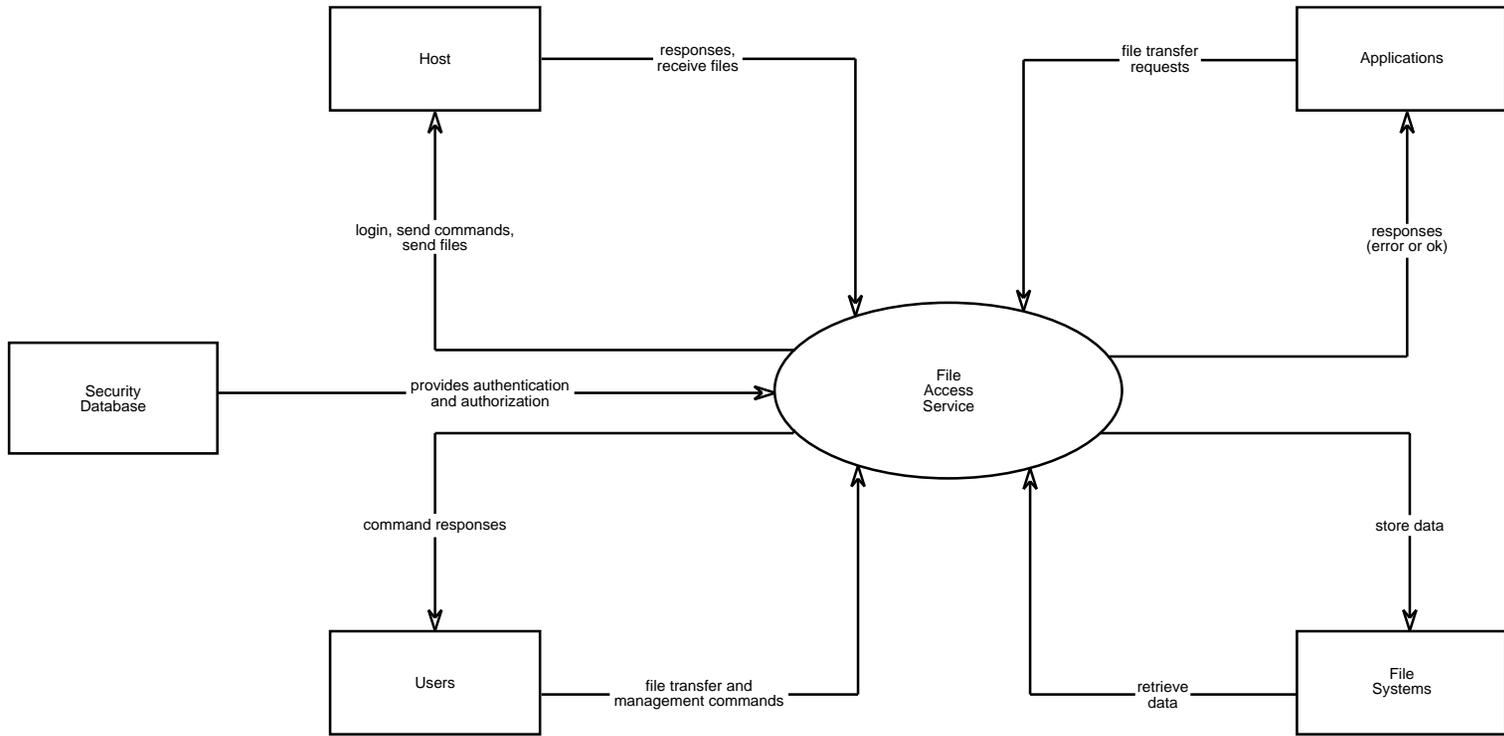
- C-CSS-61800      The CSS Electronic Mail Service shall provide the capability to send an electronic mail message non-interactively from an application.
- C-CSS-61810      The CSS Electronic Mail Service shall allow attaching multiple text or binary files to the mail message.
- C-CSS-61820      The CSS Electronic Mail Service shall accept a file name as input for the message text.
- C-CSS-61840      The CSS Electronic Mail Service shall be capable of sending a message to multiple destinations.
- C-CSS-61850      The CSS Electronic Mail Service shall accept mailing lists as valid destinations.

**6.2.2 File Access Service**

**6.2.2.1 Overview File Access Service**

The file access service provides functionality for file transfers and management. The major functions are to provide interactive access and application interfaces for distributing files.

Following context diagram shows the data flows and interaction of the service with external entities.



**Figure 6.2-2. File Access Context Diagram**

## 6.2.2.2 File Access Service Functional Requirements

Requirements for this service are divided in two sections.

*Remote File Access (RFA)* section requirements address functionality for providing transparent access to remote files as if they were part of the local file system. RFA refers to the ability to mount remote files and access them just like local files. For example, NFS and Distributed File System (DFS, which is a component of OSF DCE) offer these capabilities.

The *File Transfer* section addresses the interactive and non-interactive file transfer capabilities provided by the service.

### Remote File Access

C-CSS-60300            The CSS File Access Service shall provide transparent access to remote files.

Note: Files on remote file systems, once mounted, are accessed just like local files. All operations for the local files are available for the remote files.

C-CSS-60310            The CSS File Access Service shall support access control for the remote files.

Note: User privileges are taken into account for granting access to remote files.

C-CSS-60320            The CSS File Access Service shall provide location independent naming for the remote files.

Note: Name and location of a file in the file system is independent of the user or file location. The files appear at the same location in the distributed file system.

### File Transfer Requirements

File transfer requirements are further divided into four sections.

*General* requirements address the protocols and types of files supported by the service.

*Interactive Mode* requirements address the functionality of the interactive tool used for file transfers.

*Non-interactive mode* requirements address the scheduling option for file transfers which will be done without user presence.

The *File Transfer API* section requirements address the functionality for the application interface, which will be used by applications to transfer files.

## **General**

- C-CSS-60500 The CSS File Access Service shall provide functionality for interactive and non-interactive transfer of files (send and receive) between two host systems.
- C-CSS-60510 The CSS File Access Service shall be capable of transferring ASCII and binary files.
- C-CSS-60520 The CSS File Access Service shall support the File Transfer Protocol (FTP).
- Note: File Transfer Protocol (FTP) is described in IETF RFC 959.
- C-CSS-60530 The CSS File Access Service shall support the kerberized version of File Transfer Protocol for secured file transfers.
- Note: ECS provides kerberos server, ftp daemons and clients that support kerberos protocols described in IETF RFC 1510.

## **Interactive Mode**

- C-CSS-60600 The CSS File Access Service shall provide connection oriented operation for file transfers.
- Note: A connection, once open, may be used to send/receive multiple files.
- C-CSS-60610 The CSS File Access Service shall allow selection of the file type (ASCII or binary).
- C-CSS-60620 The CSS File Access Service shall support proxy mode of operation which enables transfer of files between two remote hosts.
- C-CSS-60630 The CSS File Access Service shall provide capability to list remote files.
- C-CSS-60640 The CSS File Access Service shall support wildcards in files on the remote host.
- C-CSS-60650 The CSS File Access Service shall support anonymous FTP which allows read access to all users.
- Note: File transfers require users to be authenticated (must have valid accounts at the remote host). This capability allows any user (without an account) to copy files from designated paths.

## **Non-interactive Mode**

- C-CSS-60800 The CSS File Access Service shall provide an option for scheduling file transfers in a batch mode.

- C-CSS-60810      The CSS File Access Service shall log results of the non-interactive operations to operator specified log files.
- C-CSS-60820      The CSS File Access Service shall provide an option to send alarms and generate events if a scheduled operation fails.

### **File Transfer API**

- C-CSS-60900      The CSS File Access Service shall provide an API which allows applications to transfer files.
- C-CSS-60910      The CSS File Access Service shall allow for file type selection (ASCII or Binary).
- C-CSS-60920      The CSS File Access Service shall accept authentication information for file transfers.

## **6.2.3 Bulletin Board Service**

### **6.2.3.1 Overview Bulletin Board Service**

This service provides a forum for sharing ECS related information. The bulletin board service consists of multiple bulletin boards (newsgroups) organized according to subjects. All users (anyone on the internet, referred to as BBS User in this section) are allowed to browse and post messages to the bulletin boards.

ECS staff post information that relates to ECS services, products, status, events and news for the ECS users.

Users (both ECS and internet users) browse the bulletin boards, report problems, suggest enhancements, ask questions, get information and share experiences with other users.

The following context diagram shows the data flows and interaction of the service with external entities.

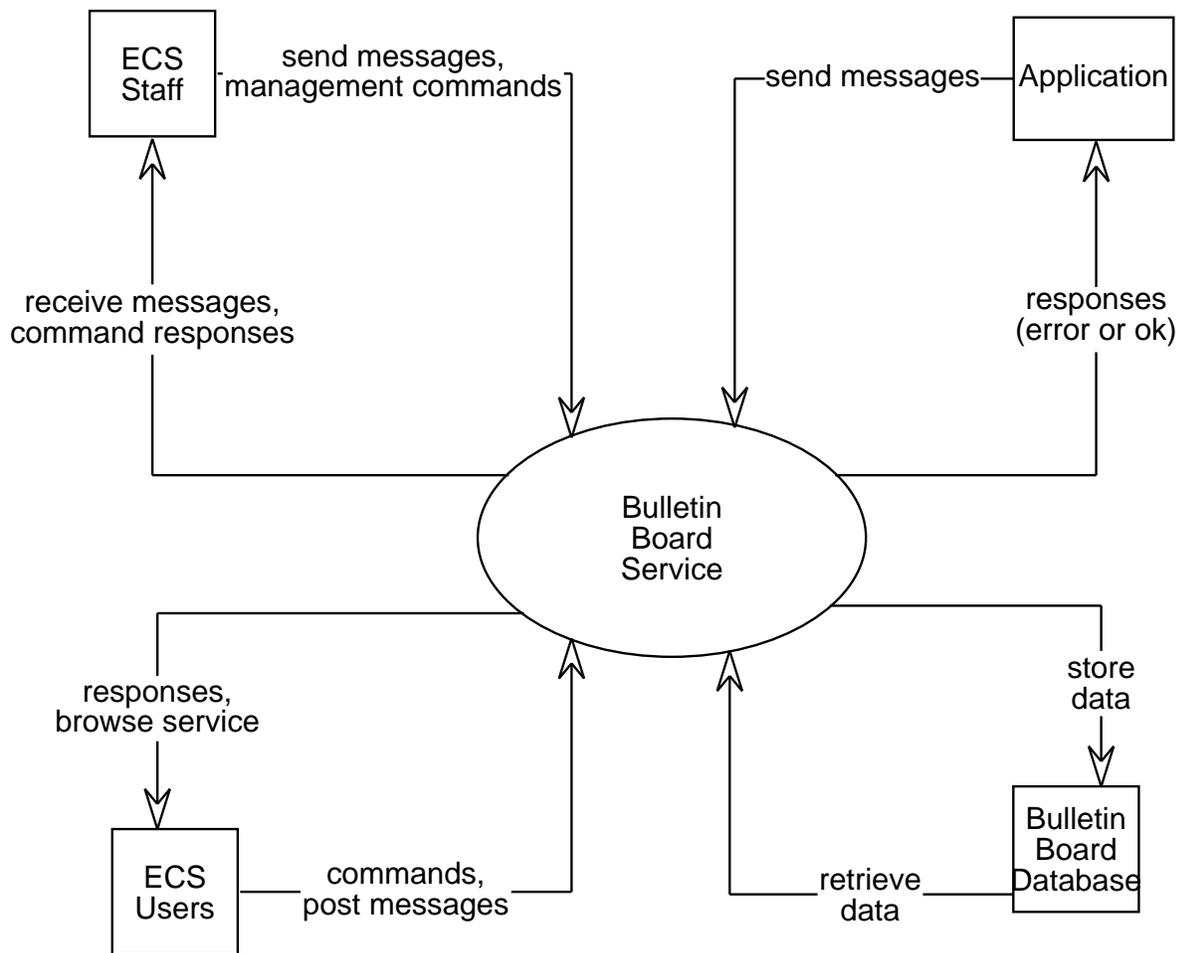
### **6.2.3.2 Bulletin Board Service Functional Requirements**

Requirements for this service are divided in three sections.

*General* section addresses the bulletin board server requirements and the protocols/standards the service needs to support.

*Bulletin Board Tool* requirements address the functionality of the interactive tool used for browsing the bulletin boards.

The *Bulletin Board API* section requirements address the functionality of the interface provided to the application programs for posting messages to ECS bulletin boards.



**Figure 6.2-3. Bulletin Board Context Diagram**

## General

- C-CSS-62000 The CSS Bulletin Board Service shall be based on the following standards:
- a. TCP/IP
  - b. NNTP
  - c. SMTP
  - d. Usenet message standard (IETF RFC 850, 1036)
- Note: Network News Transfer Protocol (NNTP) is described in IETF RFC 977
- C-CSS-62010 The CSS Bulletin Board Service shall support multiple (configurable) bulletin boards (newsgroups).
- Note: Different users may be interested in different kind of information. For this reason, we need to provide multiple newsgroups classified by subject.
- C-CSS-62030 The CSS Bulletin Board Service shall provide concurrent access to multiple users (registered or non-registered).
- C-CSS-62040 The CSS Bulletin Board Service shall allow multiple messages for each bulletin board.
- C-CSS-62050 The CSS Bulletin Board Service shall host the user registration service.
- Note: New users register using the bulletin board.
- C-CSS-62060 The CSS Bulletin Board Service shall provide the capability for copying files.
- Note: This will be the building block for toolkit distribution. Users can copy files to their local system.
- C-CSS-62070 The CSS Bulletin Board Service shall support download of ECS toolkits.
- C-CSS-62080 The CSS Bulletin Board Service shall collect and maintain access history and statistical information for the service.
- C-CSS-62100 The CSS Bulletin Board Service shall provide capabilities to authorized users (M&O staff) for:
- a. creating new bulletin board
  - b. deleting existing bulletin board
  - c. deleting message(s) from a bulletin board
  - d. backing up bulletin boards

- e. forcing users off a bulletin board or the entire bulletin board service for backup.
- f. collecting access history and/or statistical information.
- g. backing up bulletin boards.

C-CSS-62120 The CSS Bulletin Board Service shall provide the capability to respond to a posted message on a bulletin board by sending the response message to:

- a. the bulletin board (follow up)
- b. author of the original message (respond to author)
- c. named destinations (forward)

C-CSS-62130 The CSS Bulletin Board Service shall provide a “What’s new” feature which informs the user of the new information available on the bulletin boards.

### **Bulletin Board Tool**

C-CSS-62300 The CSS Bulletin Board Service shall be available to the users in interactive mode.

C-CSS-62305 The CSS Bulletin Board Service shall allow user to subscribe to bulletin boards.

C-CSS-62310 The CSS Bulletin Board Service shall allow user to unsubscribe bulletin boards.

C-CSS-62320 The CSS Bulletin Board Service shall allow user to select a subscribed bulletin board for viewing summary of all messages in it.

C-CSS-62330 The CSS Bulletin Board Service shall provide the capability to respond to a message by sending the response to the bulletin board and/or to the author of the message and/or any other operator specified destination.

C-CSS-62340 The CSS Bulletin Board Service shall provide capability:

- a. to search for a string in message headers or in message text.
- b. to search by author
- c. to search by subject.

C-CSS-62350 The CSS Bulletin Board Service shall provide a catch-up feature which excludes user specified messages from appearing in the bulletin board when it is viewed next time.

C-CSS-62360 The CSS Bulletin Board Service shall allow the users to post messages to bulletin board(s).

C-CSS-62380           The CSS Bulletin Board Service shall allow users to copy/save a message to their local system.

C-CSS-62390           The CSS Bulletin Board Service shall allow attaching ASCII or binary files to a message.

### **Bulletin Board API**

C-CSS-62800           The CSS Bulletin Board Service shall interface for the applications to post a message to bulletin boards.

C-CSS-62810           The CSS Bulletin Board Service shall allow attaching ASCII and binary files to a message.

C-CSS-62820           The CSS Bulletin Board Service shall allow a message to be posted to multiple bulletin boards.

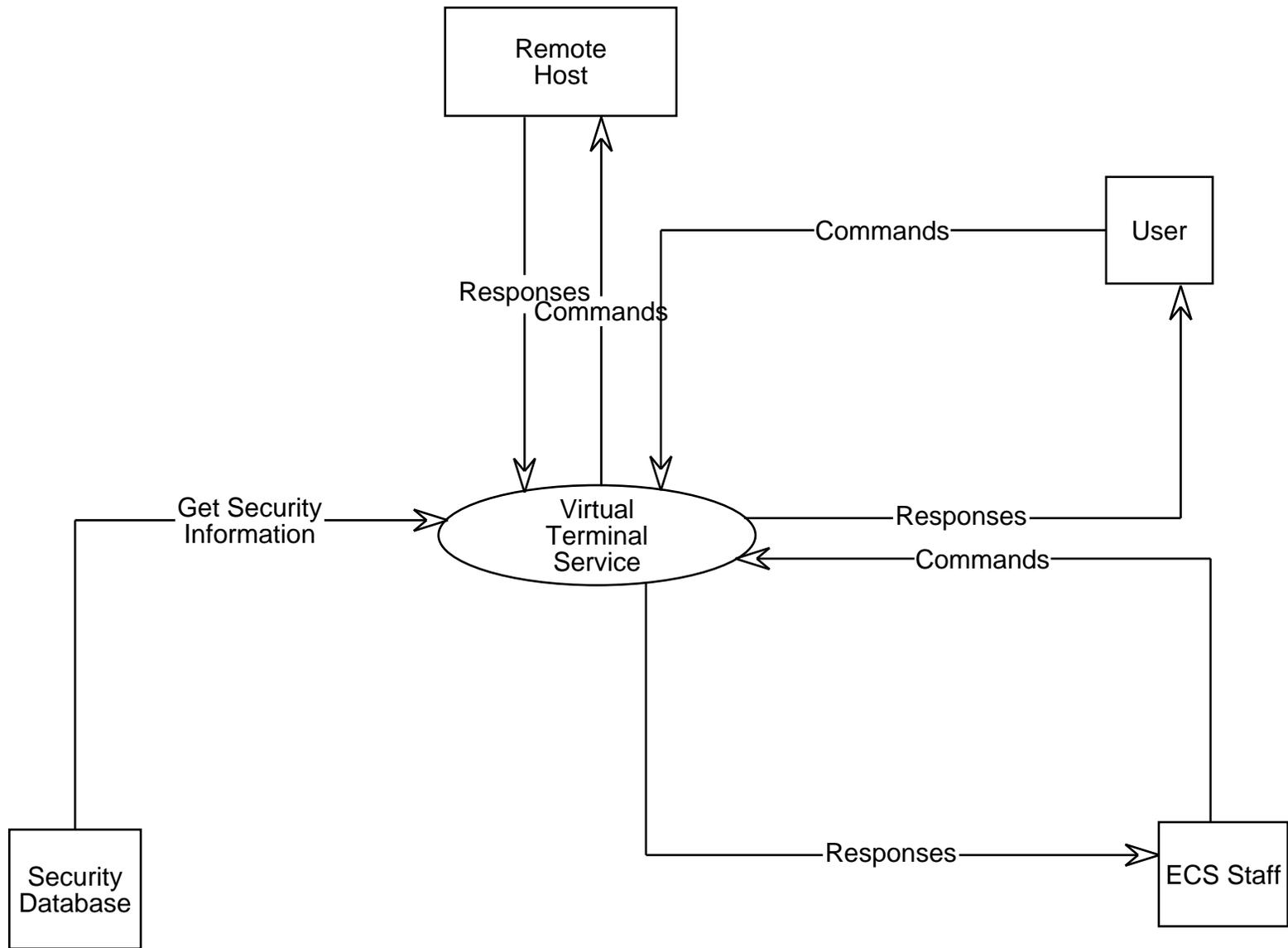
Note: The proposed API requirements may not be achievable in COTS. These are under discussion and may be changed or removed in a later version.

## **6.2.4 Virtual Terminal Service**

### **6.2.4.1 Overview Virtual Terminal Service**

Virtual terminal (VT) hides the terminal characteristics and handling conventions from both the operator and server host by allowing both parties to deal with a virtual device that has similar capabilities. VT provides operators the capability to remotely log into ECS machines.

The following context diagram shows the service and its interactions with external sources.



**Figure 6.2-4. Virtual Terminal Context Diagram**

## 6.2.4.2 Virtual Terminal Service Functional Requirements

C-CSS-63000      The CSS Virtual Terminal shall provide a virtual device which hides the physical terminal characteristics and handling conventions from both the operator and the server host.

Note: Both parties deal with the virtual device that provides a basic set of capabilities.

C-CSS-63010      The CSS Virtual Terminal shall provide means to enhance characteristics of the basic virtual device by mutual agreement between the two communicating parties (option negotiations).

C-CSS-63020      The CSS Virtual Terminal shall be based on industry standard and accepted protocols (telnet and ktelnet).

Note:              Telnet is described in IETF RFC 854, Kerberized telnet is described in IETF 1411.

C-CSS-63040      The CSS Virtual Terminal shall provide guest access to non-registered users to log into the ECS guest server.

C-CSS-63050      The CSS Virtual Terminal shall support kerberized version of the telnet protocol for secure authentication of users.

C-CSS-63060      The CSS Virtual Terminal shall support X applications.

Note: This will unable the users to run GUI applications on the desktop.

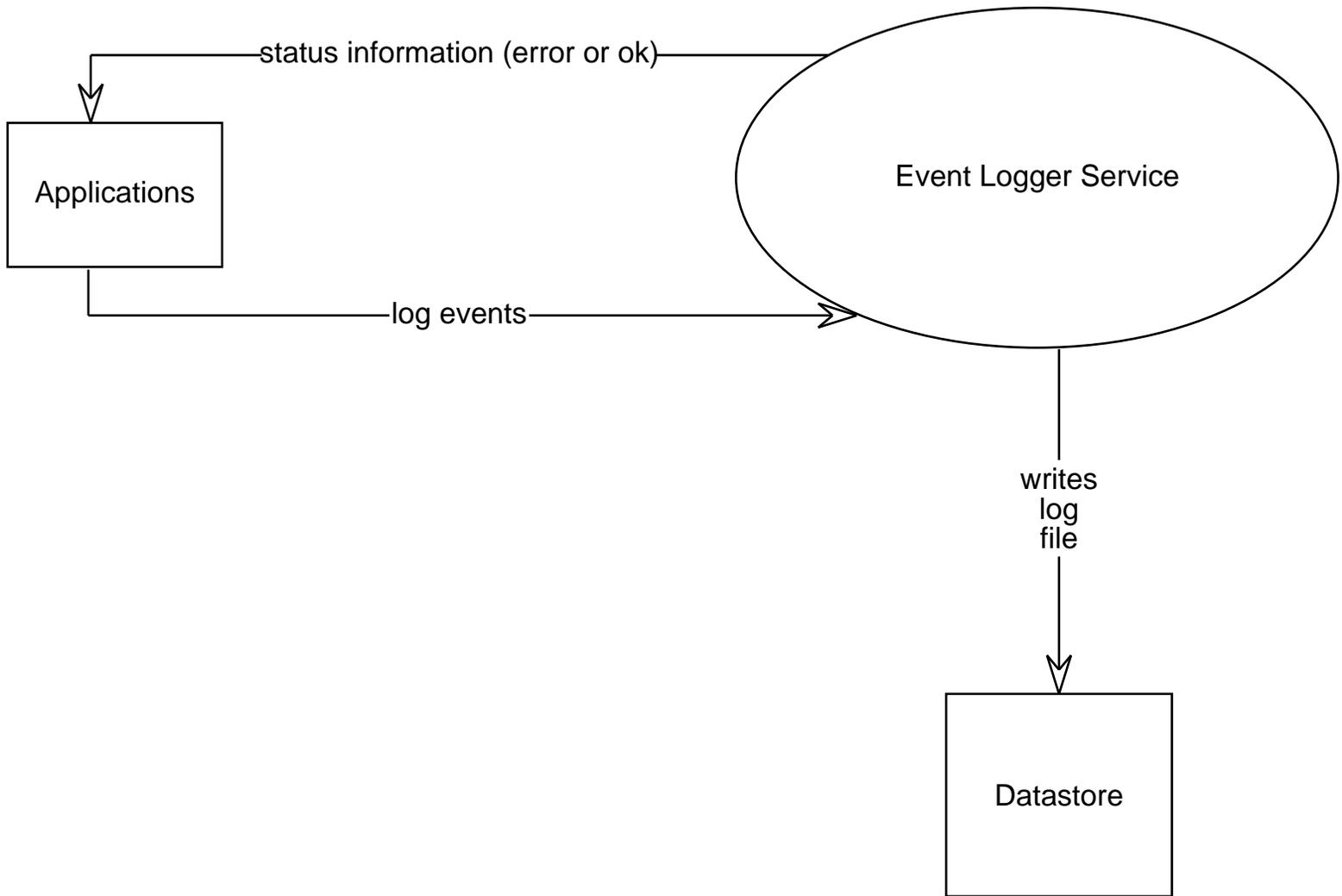
## 6.2.5 Event Logger Service

### 6.2.5.1 Overview Event Logger Service

This service allows applications to log event and history information to a application defined file which can later be used for fault, performance or statistical analysis. The service supports application defined events and collects management and fault data. Each event in the file is given a unique identifier.

The Event Logger service provides an API which records information to a log file. Browsing and consolidation of the log files is supported by the Management Data Access Service.

The following context diagram shows the service and its interactions with external sources.



**Figure 6.2-5. Event Logger Context Diagram**

### 6.2.5.2 Event Logger Service Functional Requirements

- C-CSS-28000 CSS Event Logger Service shall provide capability to record event and history data to a application specified log file.
- Note: The data is logged to a flat file and later imported into a database. Each event is assigned a unique identifier (eventid) to distinguish it from other events in the system. The eventid is recorded in the log file.
- C-CSS-28010 CSS Event Logger Service shall accept and record event time (when the event was generated, obtained from the Time Service) information.
- C-CSS-28020 CSS Event Logger Service shall accept and record the application information (name and version of the calling application).
- C-CSS-28025 CSS Event Logger Service shall support predetermined logging levels that provide different levels of information
- Note: Three levels maybe defined as nominal, moderate and debug.
- C-CSS-28030 CSS Event Logger Service shall accept and record event message information.
- Note: Message text for the event will be based on c printf format and the application will be able to pass in values for variables in the format specification.
- C-CSS-28040 CSS Event Logger Service shall accept and record the event type information. (Type of the event: fault, performance)
- C-CSS-28060 CSS Event Logger Service shall inform M&O staff if the event disposition narrative by the application demands so.
- Note: In this case, in addition to logging the event, the M&O staff is informed.
- C-CSS-28070 CSS Event Logger Service shall record the operator/principle information that is relevant for the generated event.
- C-CSS-28080 CSS Event Logger Service shall record the environment information for the generated event.
- Note: Environment information includes the OS name and version, the DCE cell name. The type of information recorded will depend on the event and the recording level.

### 6.3 Object Services

Object Services are defined as a collection of services (interfaces and objects) that support basic functions for using and implementing objects. Object Services are general purpose, domain independent and used for the construction of any distributed application.

## 6.3.1 Event Service

### 6.3.1.1 Overview Event Service

The purpose of the Event Service is to support asynchronous communications between objects. A standard remote procedure call results in the synchronous execution of an operation by an object. If the operation defines parameters or return values, data is communicated between the client and the server. A request is directed to a particular object. For the request to be successful, both the client and the server must be available. If a request fails because the server is unavailable, the client receives an exception and may take some appropriate action.

Event Service allows a more decoupled communication model between objects. For example, a system administration tool may monitor and detect if a disk runs out of space. The software managing a disk is unaware of the existence of the system administration tool. The software simply reports that the disk is full. When a disk runs out of space, the system administration tool opens a window to inform the operator which disk has run out of space.

The event service defines two roles for objects: the supplier role and the consumer role. *Suppliers* produce event data and *consumers* process event data. Event data are communicated between suppliers and consumers by issuing remote procedure calls.

There are two approaches to initiating event communication between suppliers and consumers. The two approaches to initiating event communication are called the *push* model and the *pull* model. The *push* model allows a supplier of events to initiate the transfer of the event data to consumers. The *pull* model allows a consumer of events to request the event data from a supplier. In the push model, the supplier is taking the initiative; in the pull model, the consumer is taking the initiative. The communication type is *generic*, that is, all communication is by means of generic push or pull operations, that take a single parameter that packages all the event data.

An *event channel* is an intervening object that allows multiple suppliers to communicate with multiple consumers in an asynchronous fashion. An event channel is both a consumer and a supplier of events.

Communication with an event channel is accomplished using DCE remote procedure calls.

### 6.3.1.2 Event Service Functional Requirements

- |             |   |
|-------------|---|
| C-CSS-23010 | The CSS Event Service shall provide asynchronous communication between objects  |
| C-CSS-23020 | The CSS Event Service shall provide a <i>push</i> API that allows a supplier of events to initiate the transfer of the event data to consumers.         |
| C-CSS-23030 | The CSS Event Service shall provide a <i>pull</i> API that allows a consumer of events to request the event data from a supplier                        |
| C-CSS-23040 | The CSS Event Service shall provide an intervening object that allows multiple suppliers to communicate with multiple consumers in a decoupled fashion. |

- C-CSS-23050      The CSS Event Service shall provide an API that communicates push event data to the consumer from a supplier by invoking the operation and passing the event data as a parameter.
- C-CSS-23060      The CSS Event Service shall provide an API that terminates the push event communication between supplier and consumer.
- C-CSS-23070      The CSS Event Service shall provide an API that blocks until the pull event data is available or an exception is raised. It returns the event data to the pull consumer.
- C-CSS-23080      The CSS Event Service shall provide an API that terminates the pull event communication between supplier and consumer.
- C-CSS-23090      The CSS Event Service shall provide an API that connects a push supplier to the intermediary for the push consumers.
- C-CSS-23100      The CSS Event Service shall provide an API that connects a pull consumer to the intermediary for the pull suppliers.
- C-CSS-23110      The CSS Event Service shall provide an API that connects a pull supplier to the intermediary for the pull consumers.
- C-CSS-23120      The CSS Event Service shall provide an API that connects a push consumer to the intermediary for the push suppliers.
- C-CSS-23130      The CSS Event Service shall provide an API that returns a proxy that is then used to connect a push-style consumer.
- C-CSS-23140      The CSS Event Service shall provide an API that returns a proxy that is then used to connect a pull-style consumer.
- C-CSS-23150      The CSS Event Service shall provide an API that returns a proxy that is then used to connect a push-style supplier.
- C-CSS-23160      The CSS Event Service shall provide an API that returns a proxy that is then used to connect a pull-style supplier.

## **6.3.2 Directory/Naming Service**

### **6.3.2.1 Overview Directory/Naming Service**

Directory/Naming is one of the fundamental facilities needed in distributed environments to uniquely associate a name with resources/principals along with some information so they can be identified and located by the name even if the named resource changes physical address over time.

Naming may be used more generally to store and retrieve any general information that required to be made available across a network. This information could include a server's (an ECS search program that is going to search the databases for a specified criteria) binding information, fileset

(a file containing the forest vegetation for a specific time) locations, and a resource (a printer) locations in a network, information about principals (the security namespace containing user passwords, telephone numbers). The Naming service organizes this information in namespaces.

Servers use the Naming Service to register their location and protocol in the namespace. Clients, knowing the name of the service they need to access, obtain from the Naming/Directory Service, the location and other needed information such as the communication protocol, and then bind to that service.

Similarly fileset location is kept in the namespace and users can retrieve files in a transparent manner irrespective of their physical location. Naming service provides a naming interface that support the basic naming operation. The Naming Service itself may be implemented in a variety of ways. Therefore, a standard naming interface would define a uniform interface to a large class of naming systems which can help improve portability of client-server applications across networks with Naming Services implemented with different technologies.

Standard Naming interfaces also improve the modularity and the flexibility of distributed computing. A directory agent maintains the namespace and performs directory service operations. Directory agents can communicate with other standard namespaces to provide directory services across multiple namespaces.

There are two widely known Name service specifications: ISO/CCITT X.500 and ARPA's DNS. GDS is an implementation of the X.500 and BIND is an implementation of the DNS. Of the two, BIND is widely used. Another widely used namespace is the OSF DCE's CDS. CSS will provide an implementation of both the DNS and the X.500 namespaces, along with the OSF DCE's CDS namespace.

### **6.3.2.2 Directory Service Functional Requirements:**

Basic Functionality: The Directory service will provide the basic functionality.

C-CSS-20000            The CSS Directory service shall provide the following basic functionality to save and retrieve information into the local namespace.

- a.     Create/Delete/Get context (key)
- b.     List context
- c.     Set/Get attributes
- d.     Create/Delete attributes.
- e.     List attributes.
- f.     Set/Get attribute information.

C-CSS-20010            The CSS shall provide implementations of the DNS and the X.500 namespaces.

**Replication:** To improve fault tolerance, copies of the namespace should be replicated across the network to facilitate clients retrieving binding information in reasonable time. While all the

replicas may be used for retrieval of information, only one of the replicas (master) should be writable. In order to maintain the integrity of the namespace against inconsistent updates. A mechanism should be provided to periodically propagate changes in the master to the read-only replicas.

The CSS Directory Service will maintain multiple copies of the namespace on different hosts to provide fault tolerance as mentioned in the CSS RMA section.

C-CSS-20020        The CSS Directory service shall provide a mechanism to periodically update copies of the namespace from the namespace designated as the master.

C-CSS-20025        The updating of the namespace shall be done

- a.        automatically.
- b.        manually by the administrator.

**Distribution:** Instead of storing the entire namespace in one place, it should be possible to break it into several parts and distributed in several places. Replication can then be achieved at this fine grain level allowing the administrators to have more replicas of those parts that are highly used.

C-CSS-20030        The CSS Directory Service shall provide the capability to partition the namespace and distribute and maintain them at different hosts on the network.

C-CSS-20040        The CSS Directory Service shall provide the capability to replicate partitions of the namespace on different hosts.

C-CSS-20050        The CSS Directory service shall provide multiple directory agents which cooperate among themselves through referral and chaining to perform directory operations.

C-CSS-20060        The CSS Directory service shall provide a way to denote the relative root of the namespace.

**Local cache:** Since the namespace can grow to become very large, efficiency is achieved by providing a local cache mechanism at clients where by all the latest binding information retrieved so far is stored in the local cache and looked up first when it tries to resolve a binding on behalf of a client. The disadvantage with this is that, the master database can be changed without changing the local cache where by the binding information obtained from the local cache may not be current.

C-CSS-20070        The CSS Directory Service client shall maintain local cache to keep recent lookup information from the namespace for more efficient further lookups.

**Security:** The directory module should interact with the security module to protect the contents of the directory, to provide the information only to the authorized user, and to allow that user to change the contents of the database as well as the permissions associated with the services.

C-CSS-20080            The CSS Directory Service shall interact with the Security Service to provide host based security to the entries in the namespace.

C-CSS-20085            The CSS Directory Service shall interact with the Security Service to provide principal based security to the entries in the CDS namespace and an enhanced host based security for the entries in the GDS namespace

**Attributes:** Sometimes it is advantageous for the servers to associate server defined attributes to services. These are also called properties. Properties can then be used by clients or other programs to prune services depending on a given criteria. Naming service should have provisions to let the application developers define the schema and enter/retrieve attribute value pairs in entries.

**Extensibility:** In order for the directory service to be extensible, it should support the X/Open Federated Naming specification to be able to interact with the other standard directory services supporting X.500, as well as the popular Domain Name Services (DNS). The Naming syntax is different for these standards and Directory service should provide a way to translate the different naming syntaxes among the local, X.500 and the DNS services. While supporting the basic functionality supported by the local enterprise naming, CSS Directory service shall provide a way to communicate with X.500 and DNS standard name services for name resolution.

C-CSS-20090            The CSS Directory service shall define a minimum of 20 user defined attribute types for application users to store/retrieve attribute information.

The GDS/CDS uses attribute ids in place of an attribute name while storing information in the namespace. The namespace has a constraint that an entry in the namespace can not use the same attribute type more than once. The attribute ids should be unique to avoid conflict with the attribute ids used in other namespaces. As such unique ids for the attribute types need to be obtained from standard bodies like the X/Open. DCE currently has ids for about 40 attribute types. Application programmers can make use of these attribute types as long as the entry they want to populate/edit won't use them. It is assumed that another 20 attributes types should be enough for the application programmers to save additional application related information.

C-CSS-20110            The CSS Directory service shall determine which naming service to use from a given context.

C-CSS-20120            The CSS Directory service shall provide a mechanism to communicate with both X.500 and DNS naming services in resolving lookups.

C-CSS-20130            The CSS Directory Service shall provide namespaces that are compatible with the existing NASA X.500 and DNS directory services.

### **6.3.3 Security Service**

#### **6.3.3.1 Overview Security Service**

In distributed systems, applications rely on services provided by servers running in different address spaces running on heterogeneous platforms. Servers are independent and their main functionality is to listen for client requests, process the request and send the results back to the

clients. This division of processing can be done for any number of reasons such as efficiency, availability of data etc. In addition to a client invoking a request, and the server processing that request, both the client and the server may need to use mechanisms to protect resources as well as the integrity of the data exchanged. These mechanisms comprise authentication, authorization, data integrity and data privacy. While authentication is always used in every conversation between a client and a server, the mechanisms for authorization, data integrity and privacy may be used based on the need for those mechanisms. These are explained in detail in the section below.

### **6.3.3.2 Security Service Functional Requirements**

**Authentication:** Authentication is the process of verifying the validity of a principal. In order to verify the identity of principals, systems usually associate a password (user given) with the principal names and store and maintain these passwords inside the security registry. Users may belong to groups. This group definition also needs to be maintained in the security registry. In a network environment, principals want to access the resources over the network. In doing so, the principal has to present his password in clear text along with the access request. But presenting the password in clear text poses a number of risks and should be avoided. An authentication mechanism to identify principals without passing the password in clear text is needed.

One way to deal with is to provide a central security service which keeps all the passwords in encrypted form. A user who has properly identified a user name can then get an encrypted password from the central database and then the local host which prompts the user to provide the password is then encrypted the same way and then compared to see if a principal is a valid one. There are various other ways (using secret key encryption, public key encryption, temporary conversational key encryption) to achieve this and any one of them would be able to authenticate principals with out passing the password in clear text over the network.

If temporary conversation keys are used to authenticate principals, usually they expire after a system set finite time, before which each principal has to be re authenticated. Client/Server programs running for more than this set time, have to re authenticate themselves with the security server. This will prevent someone from stealing this temporary key and use it at a latter time, as decrypting the temporary key should take longer than the key's expiration period. This should be taken into consideration while designing the authentication service.

The authentication is usually done at three places. Initially when a principal wants to log in to the system, the system should authenticate the user. When a client wants to make a request to a server, the client may want to authenticate the server who is providing the service and similarly the server may want to authenticate the client who is making the request. In the last two cases (verifying the identify of the server and client), the authentication process can be done several times depending the user needs. For example, the user may want to authenticate the server every time a connection is made, or every time a request is made or for every packet of information flowing.

There are two kinds of principals: active and non active principals. Active principals (persons) while logging in, supply the password, which is then used to authenticate the user. Non active principals can not supply the passwords interactively. In order to serve these passive users

(servers), CSS will provide a way to get the password interactively and save it locally after encrypting it. This local encrypted password will then be used to set the login context associated with a server.

In order to provide services to non DCE clients, a gateway should be provided where a DCE client is running. A non DCE user who can log (e.g. k-12 users) into the gateway is given access to a DCE client application to communicate with the DCE servers. This local DCE client will then assume a different identity given by the administrator. The CSS will provide APIs to assume a different login context for applications.

- C-CSS-21000        The CSS Security service shall provide an API to verify the identity of users.
- C-CSS-21005        The CSS Security service shall provide a unique session key for each client session.  
  
                         Note: The session key will be different for each client session, to prevent an intruder from replaying a session.
- C-CSS-21010        The CSS Security service shall not transmit passwords in clear text across networks.
- C-CSS-21020        The CSS Security service shall provide the capability to create/modify/delete user accounts and privileges in the security registry.
- C-CSS-21030        The CSS Security service shall provide the capability to define/modify/delete group information in the security registry.
- C-CSS-21040        The CSS Security service shall provide an API to limit the time after which a login context will expire.
- C-CSS-21050        The CSS Security Service shall provide an API to refresh login contexts before they expire.
- C-CSS-21060        The CSS Security Service shall provide an API to accept server keys associated with services interactively at the startup of a service.
- C-CSS-21070        The CSS Security Service shall provide an API to store server keys associated with servers to a disk file.
- C-CSS-21080        The CSS Security Service shall provide an API to retrieve the server keys associated with services from a disk file at startup time to authenticate the service.
- C-CSS-21090        The CSS Security Service shall provide an API to change the identity of an application process through server keys.
- C-CSS-21100        The CSS Security service shall provide an API to challenge the client/server to authenticate itself at the following three levels.
  - a.        connect level

- b. request level
- c. packet level

C-CSS-21105 The CSS Security Service shall notify the MSS Management Agent Service upon a predetermined number of unsuccessful login attempts.

**Authorization:** Users/principals may be associated with several groups. Authorization is the processing of deciding whether a given users/principals should be allowed to access a specified services/resources and then allow/deny the service. In authorization, each resource is associated with an access control list (ACL). Each ACL contains a list of entries specifying access permissions to users or group of users. In host based authentication, the permissions set and the principal set are fixed, and provide limited authentication capabilities: be able to authenticate only one principal/group as opposed to multiple principals/groups. CSS will provide a mechanisms for the programmer to define the permissions set and associate permissions to multiple principals/groups.

Host based authentication allows only the owner of a resource to modify the ACL associated with that resource. There will be provision to store these ACLs in persistent store. Besides the server, any principal who is authorized to change the ACLs associated with a resource, such as the M&O staff should be able to change the ACL. In order for other (authorized) users to edit the ACLs, the server will provide a well known interface. Using this well known interface, third party vendors can develop products to maintain the ACLs.

C-CSS-21110 The CSS Security service shall authenticate the principal before checking whether the principal is authorized to access a service/resources.

C-CSS-21120 The CSS Security service shall provide an API to check the authorization privileges of principals to access/control services/resources.

C-CSS-21130 The CSS Security Service shall provide an API to define the permission schema associated with a server/resource.

C-CSS-21140 The CSS Security Service shall provide an API to create and maintain the ACLs associated with the server/resource in a database.

C-CSS-21150 The CSS Security Service shall provide an API to save/retrieve the ACL database onto persistent store.

C-CSS-21160 The CSS Security service shall provide the following APIs to MSS security management applications to retrieve/modify the access control lists associated with the ECS services/resources.

- a. to identify the permissions available to a principal
- b. to identify all the ACL managers protecting an object
- c. to get the printable representation of the permissions
- d. to locate the server with the writable copy of the ACL

- e. to read an ACL
- f. to write an ACL
- g. to test if the calling principal has some permissions
- h. to test if another principal has some permissions.

**Integrity:** When data is transmitted over the network from one application to another, there should be provision to preserve the integrity of the data. This is to make sure that the copy of the data the receiver gets is exactly same as the data that the sender sends.

C-CSS-21170 The CSS Security service shall provide an API to maintain the integrity of the data passing between processes by using checksums at the following three levels:

- a. connect level
- b. request level
- c. packet level

**Encryption:** Encryption is the process of encoding a message into cipher text using a key. The process of decoding the cipher text to its original form using a key is called decryption. The encryption/decryption algorithms need not be secret. It is the key that protects the ciphered text to be decrypted. The CSS security service will provide a mechanism to encrypt and transfer messages and decrypt the received messages to maintain the privacy of the data that is being transferred.

C-CSS-21180 The CSS Security service shall provide an API to encrypt and send the data passing between processes at the following three levels:

- a. connect level
- b. request level
- c. packet level

C-CSS-21190 The CSS Security service shall provide an API to receive and decrypt the data passing between processes at the following three levels:

- a. connect level
- b. request level
- c. packet level

C-CSS-21200 The CSS Security service shall support the Data Encryption Standard (DES) to encrypt and decrypt data.

**General:** In order to provide access information to the management applications, CSS security service will log information into security logs whenever authentication and authorization services are used.

- C-CSS-21210      The CSS Security service shall provide the capability to log audit information into security logs whenever authentication and authorization services are used. The audit information will contain the following:
- a.      Date and time of the event
  - b.      User name
  - c.      Type of event
  - d.      Success or failure of the event
  - e.      Origin of the request

### **6.3.4 Message Passing Service**

#### **6.3.4.1 Overview Message Passing Service**

ECS distributed computing consists of several clients and server applications running on different platforms. Clients send data to servers, which process the data and return the result to the client. This interaction can be classified into 3 categories: synchronous, asynchronous and deferred synchronous.

In synchronous mode, a client makes a request and passes control to the server. The server services the request and returns the result back to the client, at which point the client gets back the control. The program execution on the client side is blocked until the server returns from the service. This is a blocking call and is called synchronous.

In asynchronous mode, the client makes a request with out losing control. The call won't return anything, rather is used just to pass data to a service. Client processing can continue simultaneously with the server processing. This is used in FOS applications to send real time telemetry data to SCFs asynchronously.

In deferred synchronous mode, the client makes a call and gets a ticket back from the server. Both the server and the client can continue with the processing simultaneously. The client can at a latter time communicate with the server to get the result of the request made earlier by presenting the ticket. This is used mainly for computationally intensive applications, where the return result is not needed right away to proceed with the processing at the client side. This is used in FOS and SDPS applications which are computationally intensive.

FOS applications like the Off-line Analysis Request process uses this service to send analysis data to the Off-line Analysis process and to receive the results of such analysis.

The FOS ECS Operations Control uses this to send schedule information to the ISTs.

SDPS process-intensive applications send the intermediate processing state/results to the user interface to display the results of the process done so far.

Distributed Object Framework (Section 6.4) supports synchronous message passing. Only asynchronous and deferred synchronous message passing is address here. In order to achieve

asynchronous and deferred synchronous message passing, an intermediate buffering is maintained, which collects all the messages sent and then sends them to the intended receivers.

#### **6.3.4.2 Message Passing Service Functional Requirements**

The CSS Message Passing should support both asynchronous and deferred synchronous message passing.

C-CSS-22000           The CSS Message service shall provide an API for senders to send messages to receivers asynchronously without waiting for the receivers to receive it.

C-CSS-22010           The CSS Message service shall provide an API for senders to send messages to receivers in a deferred synchronously manner through an intermediary where by they can contact the intermediary at a latter time to receive the result.

A sender may want to send the same message to different receivers. In such cases, the message queue(intermediary) should save only one copy along with the information about different servers.

C-CSS-22040           The CSS Message Service shall provide an API for the sender to designate multiple receivers for asynchronous messages.

While the sender wants to send a message to a receiver, it sends the message first to the intermediate message queue. Sending the message from the sender to the message queue is synchronous. In order to improve the performance, the message queue should be running on the same host where the sender is running so the message passing doesn't involve passing it over the network. There should be multiple instances of the message queues, so each sender that needs to send messages can instantiate a separate instance of this service.

C-CSS-22050           The CSS Message Service shall support multiple message queues so different groups of processes can use different message queues.

Unclaimed messages after some set period of time, will be purged. If a message could not be delivered in time and is being purged, then the message queue will send an event to the MSS management agents.

C-CSS-22060           The CSS Message Service shall purge a message from the message queue after a user specified time irrespective of its delivery to the receivers.

C-CSS-22065           The CSS Message Service shall log event messages to the MSS management agents whenever the message service could not deliver a message to any receiver in the time period set by the sender of the message.

This service may use persistence service, in order to save itself onto disk and to read itself from disk.

C-CSS-22070           The CSS Message Service shall store undeliverable messages and retrieve and transmit them later.

Message queues should support two kinds of models: push and pull. In push model, when a message queue receives a message, it sends them to the receivers who have registered interest in receiving that type of a message. In the pull model, it saves the messages, so that the receivers can contact the message queue and pull the message from the message queue. The receiver call should not be blocking, i.e., it should not wait until a message for it arrives at the queue. Rather it should return with a message if one exists at the message queue, or return null to indicate that there are not any messages available at the message queue.

C-CSS-22080           The CSS Message Service shall provide an API for the receiver to register interest in receiving messages from a certain sender.

C-CSS-22090           The CSS Message Service shall provide the capability to locate and send (push model) the messages to receivers.

C-CSS-22100           The CSS Message Service shall provide a non blocking API for the receiver to contact the message queue and get (pull model) the message.

Messages sent using this services, should support guaranteed delivery of the message to the receivers.

C-CSS-22110           The CSS Message service shall support guaranteed delivery of the message to the receiver.

The service should support acknowledgments, where a sender wants to know whether a message reached the receiver.

C-CSS-22120           The CSS Message service shall provide an API for the sender of the message to get the acknowledgment information the message service receives from the receivers.

In deferred synchronous message, a message can be sent to only one receiver, in order to receive the result associated with the operation. There should be a way for the sender to get the results of a computation in deferred synchronous message passing.

C-CSS-22130           The CSS Message service shall associate the receiver to a returned value and maintain that information locally until the sender requests that information.

C-CSS-22140           The CSS Message Service shall provide an API for the sender of the message to receive return information stored at the message queue.

If a receiver is not running at the arrival of a message, it should be stored and forwarded to the receiver at a latter time.

C-CSS-22150           The CSS Message Service shall defer sending a message to a receiver, if the receiver is not active, and should try sending the message periodically with a set interval of time until the receiver is active.

## 6.3.5 Time Service

### 6.3.5.1 Overview Time Service

The Time Service keeps clocks in a network approximately in sync by adjusting the time kept by the operating system at every node. Timestamps are used by many applications when recording event occurrences to log. Most of the implementation detail of the Time Service is invisible to the software developer.

Time Service provides operations to obtain timestamps based on Coordinated Universal Time (UTC). The Time Service also translates different timestamp formats and perform calculations on timestamps. The Time Service API provides the following functionality:

- Retrieving timestamp information
- Converting between binary timestamps that use different time structures
- Converting between binary timestamps and ASCII representations
- Converting between UTC time and local time
- Manipulating binary timestamps
- Comparing two binary time values
- Calculating binary time values
- Obtaining time zone information.

The Time Service includes clerks, local servers, global servers, couriers and time providers. Every node has a clerk object. The clerk compare local system clock time with the correct time. When the clerk observes incorrect system time, it queries local time servers for the correct time. A calculation is made of the most probable time and an inaccuracy factor. Adjustments are made to the nodes system time if needed.

Local time servers compare time periodically to keep their clocks in sync. Local time servers only occur within a single local area network (LAN). Global time servers provide time outside of their own LAN. Time communication between LANs is different than within a single LAN. Couriers request time information from Global servers on behalf of local servers.

A Time Provider provides access to standardized or government controlled time devices such as radios, satellites, or telephone lines. The servers within a Time Service query the Time Provider for the current time. The Time Providers are considered the most accurate source of time information.

### 6.3.5.2 Time Service Functional Requirements

- |             |   |
|-------------|---|
| C-CSS-25010 | The CSS Time Service shall adjust the time kept by the operating system at every node.                      |
| C-CSS-25020 | The CSS Time Service shall be used to obtain timestamps that are based on Coordinated Universal Time (UTC). |

C-CSS-25030	The CSS Time Service shall provide an API to retrieve timestamp information.
C-CSS-25040	The CSS Time Service shall provide an API for converting between binary timestamps that use different time structures.
C-CSS-25050	The CSS Time Service shall provide an API for converting between binary timestamps and ASCII representations.
C-CSS-25060	The CSS Time Service shall provide an API for converting between UTC time and local time.
C-CSS-25070	The CSS Time Service shall provide an API for manipulating binary timestamps.
C-CSS-25080	The CSS Time Service shall provide an API for comparing two binary time values.
C-CSS-25090	The CSS Time Service shall provide an API for calculating binary time values.
C-CSS-25100	The CSS Time Service shall provide an API for obtaining time zone information.
C-CSS-25110	The CSS Time Service shall utilize a UTC based time provider.
C-CSS-25120	The CSS Time Service shall provide the utilities required to synchronize system time across all components.
C-CSS-25130	The CSS Time Service shall have the capability to synchronize it's time to one or more external time sources.
C-CSS-25140	The CSS Time Service shall maintain an accuracy of 500 milliseconds within all ECS distributed components.

### **6.3.6 Lifecycle Service**

#### **6.3.6.1 Overview Lifecycle Service**

The purpose of the Life Cycle Services is to define services and conventions for creating objects in different locations. A factory is an object that creates another object. Factories have well defined IDL interfaces and implementations in some programming language. A factory is a creation service providing "create\_object" operation. To create an object, a client possesses an object reference for a factory and issues an appropriate request on the factory. As a result, a new object is created and typically an object reference is returned.

The client-server model assumes that servers are always available and functional. A number of hosts have idling server processes waiting for client requests. In addition to poor server utilization, this is also a drain on system resources. Servers may also be unavailable due to network or system crashes and must be restarted. It is necessary to provide users with consistent

access to servers. Rather than running all the servers all the time, it must be possible to provide the illusion that servers are always active while efficiently utilizing resources.

In the Client-Server programming paradigm, application clients assume servers are always available and active. This may not be always the case:

1. Some servers may impose too high a resource drain to be run continuously. It would be more advantageous to start these infrequently or lightly used servers only on demand, thus saving on critical network and system resources. Thus the availability of a server depends on the tradeoff between demand for that service and the overhead of running that service.
2. Even servers that need to be run all the time may not live through system or network crashes. In the event of such crashes, a service is desired to restart these servers without manual intervention.

The Lifecycle Service instantiates server instances dynamically. It intercepts client requests and ensures that the necessary servers are running. If the server is not found, the Lifecycle Service will start the required server on a supporting host in the DCE cell. It does not detect server crashes. It merely ensures that a server is available to service a user request. Lifecycle maintains a universal listener on every host and spawns application servers on demand.

### **6.3.6.2 Lifecycle Service Functional Requirements**

C-CSS-24010	The CSS Lifecycle Service shall provide a generic instantiation capability that creates a new object for a client.
C-CSS-24020	The CSS Lifecycle Service shall provide an API that accepts state initialization information.
C-CSS-24030	The CSS Lifecycle Service shall provide an API that accepts resource preference information.
C-CSS-24040	The CSS Lifecycle Service shall provide an API that returns an object invocation handle.
C-CSS-24050	The CSS Lifecycle Service shall ensure that a server is available to service a user request.
C-CSS-24060	The CSS Lifecycle Service shall act as an intermediary during the client server connection phase.

## **6.3.7 Thread Service**

### **6.3.7.1 Overview Thread Service**

A thread can be defined as a lightweight process. Threads actually exist within a process. Like a process, a thread has a program counter and other state information. The thread executes a program. Unlike processes, all threads in a process share the same address space. Each thread has

an identifier, scheduling policy, scheduling priority, error value, and data binding and the required system resources to support flow of control.

The disadvantage of having threads is that they can be allocated concurrently. Since threads can change data visible to other threads, the access to the shared data must be managed. Developing with threads must account for the possibility that other threads may change shared data at any point. Code that functions properly with multiple threads is called thread-safe.

Threads differ from the other Object Services in that threads do not involve networking. Threads are a local service that affects the operation of a single program on a single node. Threads provide an efficient and portable way to provide for asynchronous and concurrent processing, both of which are requirements of network software.

Implementing a multithreaded server does not require calling any pthread routines. Simply calling `rpc_server_listen()` with an argument value greater than 1 will cause each invocation of a server operation to run as a distinct thread. Unless the server operation needs to create additional threads for some reason, there is no need to explicitly call the pthread routines. Since each server operation executes as a result of an RPC, there is generally no need to synchronize their termination.

Implementing a multithreaded server, however, does require protecting against conflicts between different threads accessing the same data. Conflicts can occur because a thread can be timesliced at any time. Whenever a thread accesses data that can be modified by another thread, there is a potential for inconsistent behavior.

The PthreadMutex Interface is included for synchronizing threads. A mutex is a data object that is in one of two states: locked or unlocked. Once a thread has locked a mutex, no other thread can use it or lock it until it has been unlocked. A thread can lock a mutex by calling `PthreadMutex::Lock()` and unlock it by calling `PthreadMutex::Unlock()`. If the mutex has already been locked by another thread, `PthreadMutex::Lock()` will not return until the mutex becomes available.

### **6.3.7.2 Thread Service Functional Requirements**

- |             |  |
|-------------|--|
| C-CSS-26010 | The CSS Thread Service shall allow the option that each invocation of a server operation to run as a distinct thread.  |
| C-CSS-26020 | The CSS Thread Service shall protect against conflicts between different threads accessing the same data.  |
| C-CSS-26030 | The CSS Thread Service shall take into account the possibility that other threads may change shared data at any point. Code that will function correctly when executed by multiple concurrent threads is called thread-safe. |
| C-CSS-26040 | The CSS Thread Service shall provide an API that synchronizes the access of shared data between concurrent threads.  |

C-CSS-26050	The CSS Thread Service shall provide a synchronizing object that is in one of two states: locked or unlocked.
C-CSS-26060	The CSS Thread Service shall provide an API that allows each thread to lock the synchronizing object before it accesses the shared data
C-CSS-26065	The CSS Thread Service shall provide an API to release locks associated with resources.
C-CSS-26070	The CSS Thread Service shall provide an API that allows each thread to unlock the synchronizing object when it is finished accessing that data.
C-CSS-26080	The CSS Thread Service shall if the synchronizing object is locked by another thread, block the thread requesting the lock.

## 6.4 Distributed Object Framework

### 6.4.1 Overview Distributed Object Framework (DOF)

Object Oriented applications consist of 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 and get responses to. 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. 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 a distributed object framework, objects are distributed in multiple address spaces, spanning across heterogeneous platforms. Object can reside anywhere in the network, but the basic contract between an object and the users is the interface. Objects can be spread across the network due to efficiency, availability of data. From the perspective of the requester of a service, the object location (location independence), invocation (invocation independence) should be the same no matter where the object is physically present.

Invoking methods amounts to passing messages between objects. If the objects are in different address space, then the messaging should be done via network. 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. Remote procedure call is a communication method that is used to implement the client/server paradigm.

The server provides a certain operation and is called a subroutine/function for the clients to use. In that sense, a normal program can be broken down into a number of subroutines and servers implement the subroutines. Clients call these subroutines as if they are local. When a client invokes one of these remotely implemented functions, program execution transfers from the client to the server where it is processed. Once the execution is done, the result is passed back to the caller, client and the program execution flow will be turned back to the client. In order to

achieve this, there must be a standard interface definition language (IDL) to express the interface in a clear way. This is a pseudo language for which mappings should exist so that the interface expressed in IDL can be converted to high level languages like C using an IDL compiler. Once the interface is expressed in a standard language, anybody (requester) who wants to make an invocation can do so by adhering to the signatures present in the interface. The IDL should support standard types, obey some lexical rules and have a language syntax to express the interface in a crisp and unambiguous way and by preserving the semantics of the interface. Since the data formats or internal representation of data may be different on different platforms, the RPC mechanism should provide a way to convert the data into a standard format so that both the receiver and the sender would interpret the data in the same way. The process of converting the data into a standard format is called marshaling and the process of converting the data from the standard format into a platform's internal format is called unmarshaling. This paradigm deals at the program function level and has no notion of objects.

Distributed Object Framework (DOF) is like the process explained above, but instead of differentiating at the functional level, it differentiates at the object level. Objects behavior is captured in the interface definition language. Object implementation is carried on remote hosts, which are responsible to execute procedures, update the object state and return the results. This paradigm can make use of inheritance while defining new interfaces by inheriting existing interfaces. Implementation inheritance may also be possible as long as the implementation of the super class exists within the scope of the current implementation. This paradigm also needs a standard interface definition language and provide the mechanism to marshal and unmarshal standard types. In this paradigm, an executing program (client) can instantiate an object at any host which provides the implementation of that object and query that object to do certain operations. In order to do that, two objects are created: one at the client and one at the server. The object created at the server is the real object which implements the behavior of the object. The object created at the client is called a surrogate object, whose main purpose is to marshal/unmarshal the arguments, make call to the real object, and get the results back to the calling program. From the clients perspective, the call is carried locally. The surrogate object does all the underlying remote connect, instantiating the object and invoking the procedure. This is transparent to the client and is done through the use of IDL and the supporting framework.

#### **6.4.2 Distributed Object Framework Functional Requirements**

Since the interface is a contract between the client and server (which may be running on different platforms) there must a standard way to express the interface. This is done by adhering to a standard IDL. There are several standard IDLs geared towards different needs, some already defined and some being defined. IDL is a pseudo language and as such needs compilers to transform the interfaces into standard high level languages. IDL compile generates stubs for the client and for the server. The client and server software include these stubs and write the code. This code is then compiled and linked with the runtime libraries to make it into a complete application.

C-CSS-01000            The CSS DOF Service shall provide a standards-based Interface Definition Language (IDL) and language mappings to at least C and C++ (limited) languages.

Each interface written in the IDL will have a major and minor version numbers. For upward compatible new interfaces, a new higher minor version will be used in the definition of the new interface and the new implementation can be implemented which can replace the old implementation. Clients requesting information, should then get the new implementation with out any change in the client software. For non upward compatible versions, client software has to be updated to reflect the changes in the interface.

C-CSS-01010           The CSS DOF provided IDL shall support versioning of the interface supporting minor and major versions.

C-CSS-01020           The IDL supported minor versioning shall be upward compatible that requires no changes in the client software to communicate with the new implementation.

Since the client and server run in different address processes, error propagating is normal. Errors occurring in the server should be propagated to the client in a consistent way. One way is to pass error flags between the client and server for each method invocation. This is not a preferred way because, error checking should be done after each method invocation. A more elegant approach is to provide exception capabilities, where the servers can catch exceptions in their code and through that exception in the clients code. This may pose some incompatibility problems as not all high level languages support exception handling. So the framework should support passing the general error status as automatically as a parameter.

C-CSS-01030           The CSS DOF Service shall support the passing of the general error status as a parameter in calls between the clients and servers automatically.

While making a request, the arguments are to be marshaled into a common network representation format and sent to the server where they are unmarshaled into the local representation. This will be transparent to the user of the service.

C-CSS-01040           The CSS DOF Service shall provide the capability to marshal and unmarshal the arguments and the returned value transparently while making a remote procedure call.

IDLs support how to pass standard types over the network, by providing marshaling and unmarshaling methods for the standard types. Objects themselves are types, but are user defined types. Object frame work (IDL) should provide a mechanism to transfer user defined types (objects) over the network by proving hooks to marshal and unmarshal them.

C-CSS-01050           The CSS DOF Service shall provide the capability to marshal and unmarshal standard types to/from a common standard format.

C-CSS-01060           The CSS DOF Service shall provide the capability to define marshaling and unmarshaling routines for user defined types.

In order to achieve location independence, DOF will make use of a central database to register services with a user friendly names. All the needed information about a service like the protocol, host, interface are will be saved in this central place so clients can find and bind to these services. An interface can have multiple implementations. As such there should be a mechanism to

distinguish implementations of an interface. Some times a client may want to use an a particular implementation on a particular host. In order to aid this, there should be provision to identify each object implementation uniquely. The DOF should register the services with the local endpoint mapper, so the client can get the port number on which the service is being run. In order to make the lookup of the services easier, there should be a way to classify the services into different groups. If a service is being removed, then there should be a way to remove the service information from the central database.

- C-CSS-01070        The CSS DOF Service shall provide server APIs to register/unregister services in the namespaces (in different administrative domains) under different views (server/group/profile).
- C-CSS-01080        The CSS DOF Service shall provide server APIs to register/unregister different implementations of an interface in the namespace.
- C-CSS-01090        The CSS DOF Service shall provide server APIs to register/unregister individual objects implementing an interface in the namespace.
- C-CSS-01100        The CSS DOF Service shall provide server APIs to register their services using different protocols in the namespace.
- C-CSS-01110        The CSS DOF Service shall provide server APIs to register their services with the local endpoint mapper with the proper port number.
- C-CSS-01120        The CSS DOF Service shall provide mechanisms to shutdown a service gracefully, by allowing the servers to unregister the server information from the namespace.

Since a server running can service several requests simultaneously, there should be a way to limit the number of threads to service the incoming requests. If all the threads are used, then the server manager has to keep the incoming requests into a queue and process them in that order.

- C-CSS-01130        The CSS DOF Service shall provide server APIs to limit the maximum number of threads to use in servicing the requests concurrently.

In order to invoke a service, a client first has to find and bind to the service. The client obtains the binding information (binding handle) from the central database by specifying a protocol and a combination of a unique service name, a unique implementation type, a unique object name.

- C-CSS-01140        The CSS DOF Service shall provide client APIs to bind to services (registered in the local namespace as well as remote namespaces) by using any of the following information to achieve location transparency of services.
  - a.        a service name
  - b.        an interface name
  - c.        an object name
  - d.        a host name and communication protocol

- e. an object reference

C-CSS-01150 The CSS DOF Service shall return gracefully by throwing an exception or returning an error code when it can not retrieve the binding information or can not resolve a binding.

Since lookups from the central database is expensive, a local cache is maintained containing information about recent lookup information. While getting the information from the local cache is efficient, the information present in the local cache may not be up to date. As such, there should be a client selectable provision whether to use the local cache in the lookup.

C-CSS-01160 The CSS DOF Service shall provide client APIs to specify a confidence level of the binding information as follows:

- a. a low confidence level indicating the use of a local cache to obtain binding information
- b. a medium confidence level indicating the DOF to get the binding information from any of the directory replicas.
- c. a high confidence level indicating the DOF to get the binding information from the master copy of the directory services.

In order to provide security both the client and the server should be able to select the type of authentication and authorization, data integrity and data privacy to be used on a request. This is done using the security service. Since the server has a different identity than the identity of the user who brings up the server, there should be a way to set the identity of the server to a different principal.

C-CSS-01170 The CSS DOF Service shall provide APIs to set/get the authentication service type to be used between the server and the client.

C-CSS-01180 The CSS DOF Service shall provide APIs to set/get authorization service type to be used between the client and the server.

C-CSS-01190 The CSS DOF Service shall provide APIs to maintain the integrity of the data to be passed between the client and the server.

C-CSS-01200 The CSS DOF Service shall provide APIs to maintain the privacy of the data passed between the client and the server by encrypting and decrypting the data.

C-CSS-01210 The CSS DOF Service shall provide APIs to set the identity of a given principal to a given process.

The DOF should support the underlying standard TCP and UDP communication protocols, and should run on at least HP, DEC, IBM, SUN platforms. This framework should support clients/servers developed under DCE.

C-CSS-01220 The CSS DOF shall support the TCP and UDP communication protocols to communicate between the servers and the clients.

This page intentionally left blank.

## 7. ISS Functional Requirements

---

Section 7 contains the requirements associated with the Internetworking Subsystem (ISS). This includes requirements that pertain to the Networking Configuration Item (i.e., protocols associated with Transport, Network, and Datalink and Physical Services) and to the Internetworking Hardware Configuration Item (i.e., networking devices such as routers, concentrators, switches, and cabling).

Release A networking services provided by ISS will include capabilities necessary to support TRMM operations. They will also provide early interface test support for the AM-1, Landsat 7, and COLOR missions. Release A will provide interoperability with NSI and the capability for authorized users to access, search, access, and receive data holdings. ISS networks and services will support the SMC, the EOC, and at GSFC, MSFC, LaRC, and EDC DAACs. Connectivity between the four DAACs will be provided via the ESN WAN, which will have been transitioned from V0 to ECS. ISS will also support data flows between V0 and ECS, as a part of V0-ECS interoperability.

Section 7 is organized as follows.

- Section 7.1 contains general requirements pertaining to ISS, including interface requirements (both external and internal), performance requirements, RMA requirements, and evolvability requirements.
- Section 7.2 contains requirements pertaining to the Networking Configuration Item.
- Section 7.3 contains requirements pertaining to the Internetworking Hardware Configuration Item.

## **7.1 General Requirements**

### **7.1.1 ISS Interface Requirements**

Release A TRMM operations will require ISS internal (supporting SDPS and CSMS) and external interfaces at three DAACs--GSFC, LaRC, and MSFC. ECS will archive and distribute most of the data that TSDIS (the TRMM Science and Data Information Distribution System) produces. TRMM data will flow from TSDIS to GSFC and MSFC each day via the ESN WAN for archival at the two DAACs. Ancillary datasets will be sent from NOAA (Suitland) to GSFC. In addition, Level 0 data from two of the TRMM instruments, CERES and LIS, will be transported from the SDPF at Goddard to MSFC and LaRC, respectively, via NOLAN. At LaRC and MSFC, the SDPS subsystem will process the CERES and LIS data using algorithms ported from the SCFs. Algorithm integration and test will be performed by one SCF per instrument, and in both cases these SCFs are local (on the campus) to their DAACs.

Release A early interface testing will require ISS services for the following functions:

- testing of FOS, including communications with EDOS (using the EDOS simulator to send data via Ecom), communications to and from the ISTs, as well as planning, scheduling, monitoring, analysis and commanding functions within the EOC;
- testing of AM-1 ingest, requiring an ISS interface with Ecom at GSFC and LaRC;
- testing of Landsat ingest will test the connectivity between the Landsat facility and the EDC DAAC; and
- testing for Color will test the connectivity between the Color facility and the GSFC DAAC.

#### **7.1.1.1 ISS/External Interface Requirements**

ISS provides for transport and network layer interfaces with all external systems. Currently, all external interfaces use the Internet Protocol (IP) suite, and exchanges of data with all interfaces could be characterized as exchanges of data and network layer protocol services as described by the IP suite. Table 7-1 summarizes the ISS external interfaces to network providers and external end systems, as required for Release A.

**Table 7-1. ISS/External Interfaces**

<b>ISS LAN</b>	<b>Network Provider Interface</b>	<b>External End System Interface</b>
GSFC DAAC LAN	NSI	external users, including SCFs
GSFC DAAC LAN	TSDIS LAN	TSDIS
GSFC DAAC LAN	ESN WAN (ISS)	NOAA
GSFC DAAC LAN	ESN WAN (ISS)	International Partners
GSFC DAAC LAN	GSFC Campus Network	Color, external users located on the GSFC Campus Network
GSFC DAAC LAN	MSFC V0 DAAC LAN	V0 systems
MSFC DAAC LAN	NSI	external users, including SCFs
MSFC DAAC LAN	NOLAN	SDPF (L0 LIS data)
MSFC DAAC LAN	MSFC Campus Network	external users located on the MSFC Campus Network, including SCFs
MSFC DAAC LAN	ESN WAN (ISS)	International Partners
MSFC DAAC LAN	LaRC V0 DAAC LAN	V0 Systems
LaRC DAAC LAN	NSI	external users, including SCFs
LaRC DAAC LAN	NOLAN	SDPF (L0 CERES data)
LaRC DAAC LAN	LaRC Campus Network	external users located on the MSFC Campus Network, including SCFs
LaRC DAAC LAN	ESN WAN (ISS)	International Partners
LaRC DAAC LAN	LaRC V0 DAAC LAN	V0 Systems
EDC DAAC LAN	NSI	external users, including SCFs
EDC DAAC LAN	EDC Campus Network	external users located on the EDC Campus Network
EDC DAAC LAN	Landsat Production System Network	Landsat Production System
EDC DAAC LAN	ESN WAN (ISS)	International Partners
EDC DAAC LAN	EDC V0 DAAC LAN	V0 Systems
EOC LAN	Ecom	EDOS, EDF
EOC LAN	ESN WAN (ISS)	ISTs (TBD for Release A interface testing)
EOC LAN	GSFC Campus Network	ISTs located on the GSFC Campus

The external requirements for the ISS are as follows.

- C-ISS-01000      The ISS shall interoperate with the V0 Wide Area Network to provide IR-1 connectivity as specified in DID 220, "Communications Requirements for the ECS project".
  
- C-ISS-01010      The ISS shall provide an interface between the V0 WAN and the MSFC, LaRC and GSFC DAACs for the purpose of IR-1 interface testing.

- C-ISS-01020 The ISS shall interface with NSI or an alternate Internet provider at GSFC, MSFC, LaRC and EDC to provide DAAC access to science users in accordance with the following documents:
- a. DID 220, "Communications Requirements for the ECS Project" 194-220-SE3-001
  - b. Interface Requirements Document between EOSDIS Core System (ECS) and the NASA Science Internet (NSI), 194-219-SE1-001
- C-ISS-01030 The ISS shall provide for connectivity between the MSFC DAAC and NOLAN for the ingest of L0 LIS data.
- C-ISS-01040 The ISS shall provide for connectivity between the LaRC DAAC and NOLAN for the ingest of L0 CERES data.
- C-ISS-01080 The ISS shall reuse the V0 WAN in order to provide connectivity between V0 network nodes and V1 network nodes and to provide interoperability between the systems.
- C-ISS-01090 The ISS shall provide for local or metro area connectivity between V0 network nodes and V1 network nodes at GSFC, LaRC and MSFC DAAC sites in order to provide interoperability between the systems.
- C-ISS-01100 The ISS shall provide for connectivity with TSDIS in order to transfer TRMM data to the GSFC DAAC.
- C-ISS-01110 The ISS shall provide for connectivity with TSDIS in order to transfer TRMM data to the MSFC DAAC via the ESN WAN.
- C-ISS-01120 The ISS shall provide for connectivity to the MSFC campus network to enable transfer of data between SCF(s) located at MSFC and the MSFC DAAC.
- C-ISS-01130 The ISS shall provide for connectivity to the LaRC campus network to enable transfer of data between SCF(s) located at LaRC and the LaRC DAAC.
- C-ISS-01140 The ISS shall provide for connectivity to the GSFC campus network to enable transfer of data between SCF(s) located at GSFC and the GSFC DAAC
- C-ISS-01150 The ISS shall provide for connectivity between the Landsat system and the EDC DAAC to support the ingest of Landsat data.
- C-ISS-01170 The ISS shall provide for connectivity between the EOC and Ecom for AM-1 interface testing.
- C-ISS-01180 The ISS shall provide for connectivity between the EOC and the ESN Wide Area Network for AM-1 interface testing of EOC / IST communications.

- C-ISS-01185 The ISS shall provide for connectivity to the designated international partner (IP) pickup point for ASTER.
- C-ISS-01195 The ISS shall provide for connectivity with Ecom at the following ECS sites:
- a. GSFC DAAC
  - b. GSFC EOC
  - c. LaRC DAAC
  - d. MSF DAAC

### 7.1.1.2 ISS/SDPS Interface Requirements

The ISS provides the networks and services to transport data between SDPS components at each of the DAACs.

- C-ISS-01220 The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between SDPS components at the GSFC DAAC.
- C-ISS-01230 The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between SDPS components at the LaRC DAAC.
- C-ISS-01240 The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between SDPS components at the EDC DAAC.
- C-ISS-01250 The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between SDPS components at the MSFC DAAC.

### 7.1.1.3 ISS/FOS Interface Requirements

The ISS provides the networks and services to transport data between FOS components at the EOC. (Using GFE'd circuits, ISS also provides the networks and services to transport data between the EOC and ISTs. This is included in Section 7.1.1.1, ISS External Interfaces, C-ISS-1180)

- C-ISS-01190 The ISS shall provide LAN connectivity and OSI Layer 1 through 4 services between EOC components (in support of FOS interface testing at Release A).
- C-ISS-01200 The topology of the EOC LANs shall not inhibit the reconfiguration of FOS devices to support either operational or support functions.
- C-ISS-01210 The ISS shall provide the EOC with a separate network to support functions that will not interfere with the EOC's Operational LAN.

C-ISS-01215           The EOC's support LAN architecture shall be identical in function and performance to that of the operational network.

#### **7.1.1.4 ISS/CSMS Interface Requirements**

The ISS provides the networks and services to transport data between CSMS components at the DAACs, and between CSMS and SDPS and FOS components at the DAACs and EOC.

C-ISS-01255           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between CSMS components at the GSFC DAAC.

C-ISS-01260           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between CSMS components at the SMC.

C-ISS-01270           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between the SMC and the GSFC DAAC.

C-ISS-01280           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between the SMC and the EOC.

C-ISS-01290           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between the FOS EOC components and the CSMS-provided LSM within the EOC.

C-ISS-01300           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between the CSMS and the SDPS components at the MSFC DAAC.

C-ISS-01310           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between CSMS components at the MSFC DAAC.

C-ISS-01320           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between CSMS and SDPS components at the MSFC DAAC.

C-ISS-01330           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between CSMS components at the LaRC DAAC.

C-ISS-01340           The ISS shall provide LAN connectivity and OSI Layer 1 through 4 (i.e., from the physical to the transport layer) services between CSMS and SDPS components at the LaRC DAAC.

### 7.1.2 ISS Performance Requirements

The ISS performance requirements are detailed in Section 4.3.3, "ISS-INHCI Performance Requirements."

### 7.1.3 RMA Requirements

- C-ISS-04000 The ISS LANs and WANs shall have an operational availability of 0.96 at a minimum and an MDT of four (4) hours or less (1.5 hour design goal) unless otherwise specified.
- C-ISS-04020 Backups of all router configuration files shall be maintained at the local DAAC and the Network Management Facility (NMF).
- C-ISS-04030 Each ESN WAN point of presence shall have a primary and backup router.
- C-ISS-04040 The EOC LAN shall have no single point of failure for critical real-time functions.
- C-ISS-04050 The EOC Operational LAN shall provide the following levels of availability and mean down time (MDT): for critical real-time data, .99980 availability, MDT < 1 minute; for non-critical real-time data, .99925, MDT < 5 minutes.
- C-ISS-04055 The EOC Support LAN shall have an operational availability of at least 0.96 and shall have an MDT of no greater than 4 hours.
- C-ISS-04060 The portion of the DAAC LAN supporting the SDPS function of receiving science data shall contribute to the function's operational availability of 0.999 at a minimum and an MDT of two (2) hours or less.
- C-ISS-04070 The portion of the DAAC LAN supporting the SDPS function of archiving and distributing data shall contribute to the function's operational availability of 0.98 at a minimum and an MDT of two (2) hours or less.
- C-ISS-04080 The portion of the DAAC LAN supporting user interfaces to SDPS Client subsystem services shall contribute to the function's operational availability of 0.993 at a minimum and an MDT of two (2) hours or less
- C-ISS-04090 The portion of the DAAC LAN supporting the SDPS function of information searches on the ECS Directory shall contribute to the function's operational availability of 0.993 at a minimum and an MDT of two (2) hours or less .
- C-ISS-04100 The portion of the DAAC LAN supporting the SDPS function of Data Acquisition Request (DAR) Submittal including TOOs shall contribute to the function's operational availability of 0.993 at a minimum and an MDT of two (2) hours or less.

- C-ISS-04110      The portion of the DAAC LAN supporting the SDPS function of metadata ingest and update shall contribute to the function's operational availability of 0.96 at a minimum and an MDT of four (4) hours or less.
- C-ISS-04120      The portion of the DAAC LAN supporting the SDPS function of information searches on local holdings shall contribute to the function's operational availability of 0.96 at a minimum and an MDT of four (4) hours or less.
- C-ISS-04130      The portion of the DAAC LAN supporting the SDPS function of local data order submission shall contribute to the function's operational availability of 0.96 at a minimum and an MDT of four (4) hours or less.
- C-ISS-04140      The portion of the DAAC LAN supporting the SDPS function of local data order submission across DAACs shall contribute to the function's operational availability of 0.96 at a minimum and an MDT of four (4) hours or less.
- C-ISS-04150      The portion of the DAAC LAN supporting the SDPS subsystems data base management and maintenance interface shall contribute to the function's operational availability of 0.96 at a minimum and an MDT of four (4) hours or less.

**7.1.4 Evolvability Requirements**

- C-ISS-06000      The ISS network architecture shall enable expansion to GByte networks including the ability to provide increased volume of data distribution/access.

**7.2 Networking Configuration Item**

- C-ISS-02000      The ISS shall provide connection oriented transport services as specified by the TCP protocol referenced in RFC 793.
- C-ISS-02010      The ISS shall provide the capability to filter packets based on the port/socket of the transport layer protocol.
- C-ISS-02020      The ISS shall provide connectionless transport services as specified by the UDP protocol referenced in RFC 768.
- C-ISS-02030      The ISS shall provide network layer services as specified by the Internet Protocol (IP) suite referenced in RFC 791.
- C-ISS-02040      The ISS shall provide the capability to filter packets based upon network layer source and/or destination addresses.
- C-ISS-02050      The ISS shall provide ICMP network layer service as specified by RFC 792.

- C-ISS-02060 The ISS shall provide network layer services in compliance with one or more of the following protocols as appropriate to the type of the physical network supported.
- a. IP over Ethernet as specified in RFCs 894, 895, 826 (ARP), 903 (RARP)
  - b. IP over FDDI as specified in RFC 1188, 1390 (ARP, RARP)
  - c. IP over HiPPI as specified in RFC 1374 (includes ARP, RARP)
  - d. IP over SMDS as specified in RFC 1209 (includes ARP, RARP)
- C-ISS-02510 The EOC LANs shall be capable of supporting multicasting.
- C-ISS-02520 The ISS shall provide services based on the Open Shortest Path First (OSPF) protocol referenced in RFC 1583 to route traffic between the source and destination nodes, maintain route databases, and exchange routing information between networks.
- C-ISS-02530 The ISS shall provide services based on the Routing Information Protocol (RIP) referenced in RFC 1058 to route network traffic between the source and destination nodes.

This page intentionally left blank.

# Abbreviations and Acronyms

---

ACL	Access Control List
ADC	Affiliated Data Center
AM-1	EOS AM Project spacecraft 1, morning spacecraft series -- ASTER, CERES, MISR, MODIS and MOPITT instruments
ANSI	American National Standards Institute
Ao	Operational Availability
API	application program (or programming) interface
ARP	Address Resolution Protocols
ARPA	Advance Research Project Agency
ASCII	American Standard Code for Information Exchange
AUI 802.3	10 Base 5 (Thick Ethernet) Interface
BBS	bulletin board system
BPS	Bits per seconds
Bps/bps	bytes per second
CCB	Change Control Board
CCR	Commitment, Concurrency, and Recovery Protocol:
CCR	configuration change request
CD-ROM	compact disk -- read only memory
CDRD	contract data requirement document
CDRL	Contract Data Requirements List
CDS	cell directory service
CEI	contract end item
CERES	Clouds and Earth's Radiant Energy System Configuration
CI	configuration item
CM	configuration management
CMAS	Configuration Management Application Service
CMS	Common Management Services
CORBA	common object request broker architecture
COTS	Commercial off-the-shelf (hardware or software)

COTS	commercial off-the-shelf (hardware or software)
CR	change request
CRC	cyclic redundancy code
CSMS	Communications and System Management Segment
CSS	Communications Subsystem
DAAC	Distributed Active Archive Center
DAT	digital audio tape
DB	database
DBA	Database Administrator
DBMS	Database Management System
DCCI	Distributed Computing Configuration Item
DCE	Distributed Computing Environment (OSF)
DCHCI	Distributed Computing Hardware Configuration Item
DCN	document change notice
DFRD	data format requirements document
DFS	distributed file system
DID	data item description
DIM	distributed information manager (SDPS)
DNS	Domain Name Services
DOF	Distributed Object Framework
DSU/CSU	Data Service Unit/Channel Service Unit
E-mail	electronic mail
ECN	engineering change notice
ECOM	EOS Communications
ECS	EOSDIS Core System
EDC	EROS Data Center
EDF	ECS Development Facility
EDOS	EOS Data and Operations System
EIA	Electronic Industries Association
EMC	Enterprise Monitoring and Coordination
EOC	Earth Observation Center; EOS Operations Center

EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
ESN	EOSDIS Science Network
F&PR	Functional and Performance Requirements
F&PRS	Functional and Performance Requirements Specification
FDDI	Fiber Distributed Data Interface
FIPS	Federal Information Processing Standard
FOS	Flight Operations Segment
FTP	File Transfer Protocol
Gbyte	gigabyte
GDS	ground data system
GFE	Government furnished equipment
GMT	Greenwich mean time
GSFC	Goddard Space Flight Center
GUI	graphic user interface
HiPPI	High Performance Parallel Interface
I/O	input/output
ICC	Instrument Control Center
ICD	interface control document
ICMP	Internet Control Management Protocol
IDL	Interactive Data Language
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
INHCI	Internetworking Hardware Configuration Item
IP	International Partner; Internet Protocol
IR	interim release
IR-1	interim release-1
IRD	Interface Requirements Document
ISO	International Standards Organization
ISS	Internetworking Subsystem
IST	Instrument Support Terminal

IST	Instrument Support Terminal
IV&V	independent verification and validation
JPL	Jet Propulsion Laboratory
KB/SEC	kilobyte per second
LAN	local area network
Landsat	Land Remote-Sensing Satellite
LaRC	Langley Research Center
LIS	Lightning Imaging Sensor
LSM	Local System Management
LTIP	long-term science plan
LTSP	long-term science plan
M&O	Maintenance and Operations
MACI	Management Agent Configuration Item
MAGIC	Multidimensional Applications and Gigabit Internetwork Consortium
MAN	Metropolitan Area Network
MB	megabyte ( $10^6$ )
MBPS/Mbps	million bits per second
Mbyte	megabyte
MCI	Management Software Configuration Item
MDT	Mean Downtime
MHCI	Management Hardware Configuration Item
MIB	Management Information Base
MIME	Multi-purpose Internet Mail Extensions
MISR	Multi-Angle Imaging SpectroRadiometer
MITI	Ministry of International Trade and Industry (Japan)
MLCI	Management Logistic Configuration Item
mm	millimeter
MO&DSD	Mission Operations and Data Systems Directorate (GSFC Code 500)
MODIS	Moderate-Resolution Imaging Spectrometer
msec	millisecond
MSFC	Marshall Space Flight Center

MSS	Mass Storage System; Multispectral Scanner (Landsat)
MUI	Management User Interface
NASA	National Aeronautics and Space Administration
Nascom	NASA Communications
NCC	Network Control Center (GSFC)
NFS	network file system
NISS	NASA Institutional Support Systems
NMCI	Network Management Configuration Item
NMF	Network Management Facility
NNTP	Network News Transfer Protocol
NOAA	National Oceanic and Atmospheric Administration
NOLAN	Nascom Operational Local Area Network
NSI	NASA Science Internet
NWCI	Networking Configuration Item
ODC	Other Data Center
OO	Object Oriented
OS	operating system
OSF	Open Systems Foundation
OSI	Open System Interconnect
OSI-RM	OSI Reference Model
OSPF	Open Shortest Path First (routing protocol)
P&S	planning and scheduling
PDB	Project Database
PDR	Preliminary Design Review
POCC	Payload Operations Control Center
POSIX	Portable Operating System Interface for Computer Environments
PSCN	Program Support Communications
RARP	Reverse Address Resolution Protocol
RBG	red, blue, green
RFA	Remote File Access
RFC	Request for Comment

RIP	Routing Information Protocol
RMA	Reliability, Maintainability, Availability
RPC	remote procedure call
SCF	Science Computing Facility
SDP	Science Data Processing
SDPF	Sensor Data Processing Facility
SDPS	Science Data Processing Segment
SMC	System Management Center
SMCI	System Management Configuration Item
SMDS	Switched Multi-megabit Data Service
SMTP	Simple Mail Transfer Protocol
SN	Space Network
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
SRS	Software Requirements Specification
TBD	to be determined
TBS	to be specified
TCP/IP	Transmission Control Protocol/Internet Protocol
TOOs	target of opportunity
TRMM	Tropical Rainfall Measuring Mission
TSDIS	TRMM Science Data and Information System
UDP	user datagram protocol
UTC	universal time code
VAC	volts AC
VO	Version O
VOM	volt ohm meter
WAN	Wide Area Network
X.400	OSI standard for mail services
X.500	OSI standard for directory services (207)